

2023

Web Application Development using PHP

Dr. Babasaheb Ambedkar Open University



Web Application Development Using PHP

Expert Committee

Prof. (Dr.) Nilesh K. Modi Professor and Director, School of Computer Science, Dr. Babasaheb Ambedkar Open University, Ahmedabad	(Chairman)
Prof. (Dr.) Ajay Parikh Professor and Head, Department of Computer Science Gujarat Vidyapith, Ahmedabad	(Member)
Prof. (Dr.) Satyen Parikh Dean, School of Computer Science and Application Ganpat University, Kherva, Mahesana	(Member)
M. T. Savaliya Associate Professor and Head Computer Engineering Department Vishwakarma Engineering College, Ahmedabad	(Member)
Mr. Nilesh Bokhani Assistant Professor, School of Computer Science, Dr. Babasaheb Ambedkar Open University, Ahmedabad	(Member)
Dr. Himanshu Patel Assistant Professor, School of Computer Science, Dr. Babasaheb Ambedkar Open University, Ahmedabad	(Member Secretary)

Course Writer

Dr. Ashish Parejiya	INDUS Institute of Information & Communication Technology
Dr. Kamallesh Salunke	Department of Computer Science, Gujarat Vidyapith
Dr. Ketan D Patel	AMPICS, Ganpat University

Content Reviewer and Editor

Prof. (Dr.) Ajay Parikh	Department of Computer Science, Gujarat Vidyapith
Prof. (Dr.) Nilesh K. Modi	Professor and Director, School of Computer Science, Dr. Babasaheb Ambedkar Open University, Ahmedabad

Copyright © Dr. Babasaheb Ambedkar Open University – Ahmedabad.

ISBN -

Printed and published by: Dr. Babasaheb Ambedkar Open University, Ahmedabad While all efforts have been made by editors to check accuracy of the content, the representation of facts, principles, descriptions and methods are that of the respective module writers. Views expressed in the publication are that of the authors, and do not necessarily reflect the views of Dr. Babasaheb Ambedkar Open University. All products and services mentioned are owned by their respective copyrights holders, and mere presentation in the publication does not mean endorsement by Dr. Babasaheb Ambedkar Open University. Every effort has been made to acknowledge and attribute all sources of information used in preparation of this learning material. Readers are requested to kindly notify missing attribution, if any.



Web Application Development using PHP

Block-1: Open Source & Linux Administration

UNIT-1

Open-Source Software - Introduction 02

UNIT-2

Open-Source Software Vs Closed Source Software 07

UNIT-3

Linux Administration 10

UNIT-4

Manage File Permissions and Processes 17

Block-2: Database Management Using MySQL

UNIT-1

Introduction to Relational Database Management System and MySQL 26

UNIT-2

Structured Query Language 39

UNIT-3

SQL Sub Languages 61

Block-3: PHP Programming

UNIT-1

Getting started with PHP 085

UNIT-2

Control and Looping Statements 115

UNIT-3

Working with Functions 142

UNIT-4

Working with Arrays 178

Block-4: Processing Web Forms and Handling Database in PHP

UNIT-1

Working with forms in PHP 00

UNIT-2

File and directory access in PHP 00

UNIT-3

Working and formatting with strings 00

UNIT-4

Handling Databases in PHP 00

Block-1

Open Source & Linux Administration

Unit 1: Open-Source Software - Introduction

1

Unit Structure

- 1.1. Learning Objectives
- 1.2. Introduction of Open-Source Software
- 1.3. Open-Source Products
- 1.4. Open-Source Software Development Philosophy
- 1.5. Pros and Cons of Open-Source Software
- 1.6. Suggested answers for check your progress

1.1 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand about open-source software and products
- Understand pros and cons of open-source software.

1.2 INTRODUCTION OF OPEN-SOURCE SOFTWARE

The term open-source means something which is publicly available and one can modify and share it. The idea behind open source is that the software should be available to everyone so that people can understand and learn it, modify it and improve it. Generally, these kinds of sources are available under GNU General Public Licence.

Open-source software is a kind of computer software whose source code is made available to all the users. It gives freedom to users to use it, modify it, add some functionality into it, improve the existing futures and fix the bugs in the earlier versions.

Open-source software is different from any other type of software in a way that their creators make the source code available to all and allow them to learn, modify and share it. Open-source software licences permits the users to use it for their intended purpose. The biggest advantage of open-source software is that they are freely available and there is no need to purchase any licence.

These open-source software are managed by the open-source community which is a worldwide community of software developers and programmers. These community is continuously working on various open-source projects to make them better.

Check your progress - 1:

1. What are open-source software?

.....
.....
.....

1.3 OPEN-SOURCE PRODUCTS

There are various of open-source products available world-wide for different purposes. Some of the popular open-source products are:

- **Operating Systems** – Linux, Android, Ubuntu, Fedora etc.
- **Browsers** – Firefox, Chromium, Brave etc.
- **Web Development Tools** – Angular JS, Node JS, ReactJS, XAMPP, Notepad++, WAMP, phpMyAdmin etc.
- **Programming Languages** –Java, PHP, Python, Ruby, R Programming etc.
- **CMS** – WordPress, Joomla, Dhrupal, Magento etc.

Check your progress - 2:

1. List out any 5 open-source software products with their use?

.....
.....
.....

1.4 OPEN-SOURCE SOFTWARE DEVELOPMENT PHILOSOPHY

Open-Source software supports the belief that the source code of Open-Source software should be available publicly to view, change, improve and share that code without paying for it

These software's are community-based projects which are built by group of developers and programmers who spent their time and expertise to build this software, modify them, improve them by adding new features and share the source code to everyone without any financial motive.

Open-source philosophy promotes:

- Sharing source code to everyone
- Collaborative works
- Free exchange and sharing of information

1.5 PROS AND CONS OF OPEN-SOURCE SOFTWARE

Pros:

- **Cost Effectiveness** – generally open-source software's is freely available. Users are not required to pay anything.
- **Flexible** – Users are free to modify it as per their requirement.
- **Reliability** – open-source software's are highly reliable as they are developed and maintained by group of expert developers and programmers.
- **Community Support** - open-source software's are managed by strong user communities and hence they are stable and one can trust on it. Users will also get modified versions on regular intervals.

Cons:

- **Not user friendly** – all the open-source software's are not user friendly and they are not easy to use. Some of their GUIs are not user friendly.
- **Security** – as the code is freely available and free to edit, there are chances of vulnerabilities due to misuse of code.
- **Compatibility issues** -Many types of proprietary hardware need specialised drivers to run open-source programs which may lead to extra cost.
- **Lack of Support** – you can not expect the level of support that you receive from any proprietor software.
- **Extra Cost** – Though open-source software's are free to use there are some hidden costs involves while using them such as – additional hardware cost, training cost, support and maintenance cost etc.

Check your progress - 3:

1. What are the pros and cons of open-source software?

.....
.....
.....

1.6 SUGGESTED ANSWERS FOR CHECK YOUR PROGRESS

Check Your Progress - 1:

refer section 1.2

Check Your Progress - 2:

refer section 1.3

Check Your Progress - 3:

refer section 1.5

Unit 2: Open-Source Software Vs Closed Source Software

2

Unit Structure

2.1 Learning Objectives

2.2 Open-Source Vs Closed Source Software

2.3 Free Software

2.4 Source Available Software

2.5 Suggested answers for check your progress

2.1 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand about open-source software and closed source software
- Understand the difference between open and closed source software
- Know about open-source development tools.

2.2 OPEN-SOURCE Vs CLOSED SOURCE SOFTWARE

There are some differences between open and closed source software.

Open-Source Software	Closed-Source Software
Open-source software refers to the computer software which source code is open means the general public can access, use and modify it.	Closed-source software refers to the computer software which source code is closed means public is not given access to the source code.
source code is publicly available.	source code is not publicly available. It is

	protected.
User can modify the source code.	The owners or creators can only modify the source code.
In most cases, open-source software is freely available.	They are not freely available and user has to pay for use.
They can be installed to any computer.	They need to have a valid license before installation into any computer.
No one is explicitly responsible for an Open-Source software.	The owner or vendor is responsible for Closed-Source software.
They are often less user friendly	Generally, more user friendly.
Updates may take longer time to arrive.	Generally, updates are available more quickly.
Examples: Linux, Firefox, PHP, Open Office etc.	Examples: Windows OS, Microsoft office, Adobe Photo Shop etc.

Check your progress - 1:

1. What is the difference between open and closed source software?

.....
.....
.....

2.3 FREE SOFTWARE

A software that respects users' freedom and community is called Free software. It means that the users have the freedom to study, run, change, copy, distribute and improve the software. Thus, free software is a matter of liberty, it is not about price. It is about freedom.

According to GPL (General Public Licence) a software is called free software if its users have following freedom:

- The freedom to run the software for any purpose.
- The freedom to study the source code and change the software for any purpose.
- The freedom to share the software with others.
- The freedom to share your own modified versions of the software with others.

2.4 SOURCE AVAILABLE SOFTWARE

Source-available software is software released through a source code distribution model that includes arrangements where the source can be viewed, and in some cases modified, but without necessarily meeting the criteria to be called open-source. [Source: Wiki]

Check your progress - 2:

1. What is the difference between free software and source available software?

.....

.....

.....

2.5 SUGGESTED ANSWERS FOR CHECK YOUR PROGRESS

Check Your Progress - 1:

refer section 2.2

Check Your Progress - 2:

refer section 2.3 and 2.4

Unit 3: Linux Administration

Unit Structure

3.1 Learning Objectives

3.2 Linux Administration

3.3 Configuring the bash shell

3.4 Finding Files

3.5 Managing Users and Groups

3.6 Suggested answers for check your progress

3.1 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand the job of Linux administration
- Configure bash shell
- Manage users and groups

3.2 LINUX ADMINISTRATION

Linux system administration is a process of setting up, configuring, and managing a computer system in a Linux environment. System administration involves various tasks such as: creating and managing user accounts, taking reports, performing backup, updating configuration files, documentation, and performing recovery actions. System administrator is the person who manages the server, fixes configuration issues, recommends new software updates, and updates the document.

In this unit we will learn some of the system administration tasks such as managing users, groups and permissions, managing processes, finding files and so on.

3.3 CONFIGURING THE BASH SHELL

A Shell is a command that provides command line interface for Linux OS. Bash shell reads the Linux commands and execute them.

If you want to check the available shell in your system you can use:

```
cat /etc/shells
```

Changing from other shell to Bash

to change from any shell to the bash, type following:

```
bash
```

it will change the shell to bash shell.

chsh (change shell) command is used to change your default shell.

```
chsh -s /bin/bash
```

the above command will change the default shell to bash shell for current user.

We can also change the default shell of other users using chsh command. Example:

```
chsh -s /bin/shkali
```

it will change the default shell of the user kali to */bin/sh*. We need to login as root user.

If you want to know the default shell for a user you can use grep command. Grep is an essential Linux command which is used to search text and strings from a given file.

```
grep kali /etc/passwd
```

the above command will find the default shell of user named kali.

Check your progress 1:

How can you change the shell in Linux?

.....
.....
.....

3.4 FINDING FILES

The find command in Linux is used to find a file. You can use the find command to search for a file or directory on your file system.

find expressions take the following form:

find options starting/path expression

- The options attribute will control the find process's behaviour and optimization method.
- The starting/path attribute will define the top-level directory where find begins filtering.
- The expression attribute controls the tests that search the directory hierarchy to produce output.

find . -name testfile.txt Find a file called testfile.txt in current and sub-directories.

find /home -name *.jpg Find all .jpg files in the /home and sub-directories.

Check your progress 2:

How can you find files in Linux?

.....
.....
.....

3.5 MANAGING USERS & GROUPS

To create a secure environment in Linux, it is essential to learn about user groups and their permissions. User permissions are like you want someone to access the file but can't modify it.

Creating User Accounts:

We can create a new user account by using the following command:

```
sudo useradd newuser
```

there are two ways to make sure that the new user account has been successfully created:

1. With the use of *id* command.

```
id newuser .
```

it will print output like this

```
uid=1000(newuser) gid=1000(newuser) groups=1000(newuser)
```

This will print details like the user id and the groups of that user, usually a new group with the same username is assigned to the user.

2. By opening the following file: `/etc/passwd`.

So, we can use `cat /etc/passwd` and we can confirm that the new user has been created.

After creating the user using the command above, you may notice that no user directories have been created inside `/home` directory, which creates problem that the user cannot log in to his account. Hence it is required to create new user directory.

To create a new user with its directories, we can use following:

```
sudo useradd -m -s /bin/bash newuser
```

If you look into the /home directory, you can see that a new directory with the name newuser is created now.

To set a new password for the newuser executes the following:

```
sudo passwd newuser
```

After creating a new user and setting a password to it, newuser can login through GUI or by terminal

Deleting a user

If you want to delete the user then you can use userdel command of linux.

```
sudo userdel newuser
```

userdel command will delete the newuser but the user directory will not be deleted. You need to delete it by yourself.

You can use following command to do all this at once:

```
sudo deluser --remove-home newuser
```

Managing User groups

A group is a collection of users. The primary purpose of creating the groups is to define a set of privileges like read, write, or execute permission for a given resource that can be shared among the users within the group.

Create a group

Before creating the group, if you want to see all the groups you can do it like:

```
cat /etc/group
```

now Let's create a new group with the name newgrp as:

```
sudo groupadd newgrp
```

it will create new group called newgrp

Adding user to a group

After creating the group, lets add our user newuser into the group newgrp:

```
sudo usermod -aG newgrp newuser
```

Deleting the user from the group

You can delete the user from the group using the following:

```
sudo groupdel newuser
```

it will delete the user newuser from the group newgrp.

Delete a group

If you want to delete a group then you can do it as:

```
sudo groupdel newgrp
```

Check your progress 3:

How can you manage users and groups in Linux?

.....
.....

3.6 SUGGESTED ANSWERS FOR CHECK YOUR PROGRESS

Check your Progress 1:

Refer section 3.3

Check your Progress 2:

Refer section 3.4

Check your Progress 3:

Refer section 3.5

Unit 4: Manage File Permissions and Processes

4

Unit Structure

4.1 Learning Objectives

4.2 Introduction

4.3 Managing File Permissions

4.4 Managing Processes

4.5 System Administration Tools

4.6 Suggested answers for check your progress

4.1 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Manage permissions
- Manage processes in Linux
- Aware about various system administration tools

4.2 INTRODUCTION

As we know that Linux is a multi-user operating system and hence it can be accessed by many users simultaneously. It is also be used in mainframes and servers without any modifications. But this may lead to security issues such that a user with malafide intention can do anything like he/she can corrupt, alter or remove important data. Hence Linux divides authorization into two levels for effective security:

1. Ownership
2. Permission

In this unit we will understand these two levels in brief.

4.3 MANAGE FILE PERMISSIONS

Let's start by discussing about the ownership in Linux files. It is as below:

1. User: the owner of the file (person who has created the file).
2. Group: the collection of users, group can contain multiple users. Therefore, all users in same group will have the same permissions.
3. Other: any person has access to that file, who has neither created the file, nor are they in any group which has access to that file.

If you want to see file permissions you can use following command:

```
ls -l
```

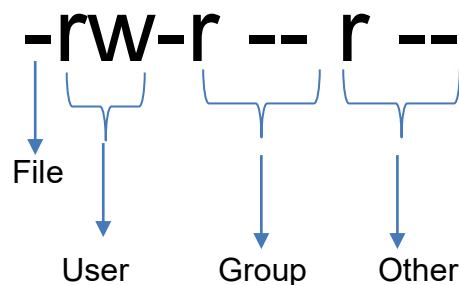
it will display file information along with permissions as below:

```
-rw-r -- r --
```

Lets discuss what it means.

These characters indicate different permissions like:

- r - read.
- w - write.
- x - execute.
- - no permission.



As you can see in the above image, the first part denotes file or directory. empty first part means it is a file. It would be the letter 'd' if it was directory. The second part means that the user 'Home' has both read and write permissions but he does not have the execute permission. The group and others have only the read permission.

If you want to change the permissions then use chmod command.

```
chmodo+w myfile.txt
```

it will add the write permission to other users on myfile.txt text file. Here 'o' means to others, 'g' means group, 'u' means user and 'a' means all. Now the other users have both read and write permissions. If you execute ls -l again then you can see the permissions like -rw-r--rw-.

Now let's add the execute permission to the user with following command:

```
chmodu+x myfile.txt
```

now the permissions look like -rwxr--rw-.

If you want to remove the permission, you can use the same command but with '-' instead of '+'.
For example, let's remove the write permission from the others:

```
chmod o-w myfile.txt
```

And the permissions now look like: -rw-r--r--.

Instead of using character symbols, you can use Symbolic Mode to modify permissions like the following:

Number	Permission
0	No permission
1	Execute

Number	Permission
2	Write
3	Execute and Write
4	Read
5	Read and Execute
6	Read and Write
7	Read, Write and Execute

For example, let's give every permission for all users as:

```
chmod 777 myfile.txt
```

here first 7 is for user, second 7 is for group and third 7 is for other. It means all users have all permissions.

Now the permissions look like

```
-rwxrwxrwx
```

Check your progress 1:

What is file permission and how can you change it in Linux?

.....

.....

.....

4.4 MANAGING PROCESSES

A process is a set of instructions loaded into memory. The Linux kernel tracks every aspect of a process by its PID under `/proc/PID`.

Listing All Processes

The `ps` command is used to view the information about process. By Default, it shows processes from the current terminal. The options that can be used with `ps` are:

- a: Shows processes from all the terminals.
- x: Shows all the processes owned by you, or shows all the processes when used together with an option (such as: `psax`) Including processes that are not controlled by a terminal Such as Daemon processes.
- u: prints process owner information.
- f: Shows process parentage
- o: Shows custom information.

If you want a list of all the processes running on your system, you can run `ps aux` command.

Tracking system activities

Top command tells the user about all the running processes on the Linux machine.

Kill command is used to terminate all the running processes.

Syntax: `kill pid`

Where `pid` is the process id that you want to terminate.

Process Status

Every process has a state property, which describes whether the process is actively using the CPU (Running), in memory but not doing anything (Sleeping), waiting for a resource to become available (Uninterruptable Sleep) or terminated but not flushed from the process list (Zombie).

We can check the state of the process by executing commands like `top` and `ps`.

Setting Priority of the Process

There are hundreds of processes running in a system at given point of time, which are created mostly by the Linux operating system and some of them are created by the logged-in user. Each running process has a priority assigned to it and this priority determines how fast the process is executed by the system. High priority processes are executed early then the low priority processes.

You can change the priority of the process with the use of *nice* and *renice* commands in Linux.

The nice value ranges from -20 to 19. Default value is 0. Where -20 denotes Highest priority and 19 denotes Lowest priority.

We can use *ps* or *top* command to check the nice value of the process. Nice value is represented under NI column header.

Example:

```
nice -5 gnome-terminal
```

to set the negative priority, use double hyphen. You need to have root privilege to assign negative priority to process like:

```
nice --5 gnome-terminal
```

you can use *renice* command to change the priority of running process.

```
renice -n 10 -p 7210
```

it will set the new priority 10 to the process having id 7210.

Check your Progress 2:

How can you change the priority of the running process in Linux?

.....
.....
.....

4.5 SYSTEM ADMINISTRATION TOOLS

The job of Linux system administrator includes OS installation, upgrade, and monitoring system performance. There are some tools that makes administrator job easy. Few of such popular tools are explained here. You can visit their official web pages to get more insights about their detailed functionality.

Webmin: it is a web-based interface for Unix system administrators. It helps sysadmin to configure and modify various system internals.

Zenmap: it is a free and open-source tool which is used to scan for network issues. It is the GUI version of Nmap Security.

Shorewall: it is a free and open-source GUI for configuring firewalls, creating and managing blacklists, gateways, VPNs, and controlling traffic.

Cockpit: it is widely used for regular server administration tasks. It is best suited for Red Hat OS.

PHPMyAdmin: it is an open-source PHP based web app that allows users to create and manage MySQL databases using a web browser. It provides user friendly GUI to manage MySQL databases.

MySQL Workbench: MySQL Workbench is a graphical tool for working with MySQL servers and databases. The main functionality includes: server administration, data migration, SQL development, data modelling etc.

Nmap: it is a network scanner widely used by network admins to monitor network and to detect vulnerabilities.

Check your Progress 3:

List out various system administration tools in Linux.

.....
.....
.....

4.6 SUGGESTED ANSWERS FOR CHECK YOUR PROGRESS

Check your Progress 1:

Refer section 4.3

Check your Progress 2:

Refer section 4.4

Check your Progress 3:

Refer section 4.5

Block-2
Database Management Using
MySQL

Unit 1: Introduction to Relational Database Management System and MySQL

1

Unit Structure

- 1.1. Learning Objectives and Outcome
- 1.2. Introduction
- 1.3. Introduction to Relational Database Management System
- 1.4. Introduction to MySQL
- 1.5. Why MySQL?
- 1.6. MySQL Installation and Configuration
- 1.7. MySQL Datatypes
- 1.8. Check Your Progress
- 1.9. Check Your Progress: Possible Answers

1.1 LEARNING OBJECTIVES AND OUTCOME

After studying this unit student will be able to:

- Introduction to Relational Database Management System and MySQL
- Understand basic concepts of database and table
- MySQL/Maria datatypes
- Concepts of unique key, primary key, foreign key and table constraints.

Outcome:

- Install, manage and work with MySQL
- Understand and explain the roles RDBMS
- MySQL datatypes and unique key, primary key, foreign key and table constraints.

1.2 INTRODUCTION

In a simple word Database is a repository where you can store your data or you can say a database is a set of data stored in a computer. This data is usually structured in a way that makes the data easily accessible and controlled redundancy. Database Management System (DBMS) is application software that allows you to create, update, and administer the database.

A relational database is a type of DBMS. RDBMS stands for Relational Database Management Systems. Most relational database management systems use the Structure Query Language (SQL) to access the database. IBM developed the SQL language in mid-1979. All communication with the clients and the RDBMS or between RDBMS is via SQL. Whether the client is a basic CUI SQL engine or a disguised GUI engine, report writer or one RDBMS talking to another, SQL statements pass from the client to the server. The server responds by processing the SQL and returning the results. The advantage of this approach is that the only network traffic is the initial query and the resulting response. The processing power of the client is reserved for running the application. SQL open standard language to work similarly with most the databases. SQL consist of three built-in sub-languages: Data definition language (DDL), Data manipulation language (DML) and Data control

language (DCL). SQL is a fourth generation language. SQL has many more features and advantages. Let us discuss the SQL in more detail in this chapter.

1.3 INTRODUCTION TO RELATIONAL DATABASE MANAGEMENT SYSTEM

RDBMS stands for Relational Database Management Systems. All modern database management systems like MySQL, MS SQL Server, IBM DB2, ORACLE, and Microsoft Access are based on RDBMS. It is called Relational Data Base Management System (RDBMS) because it is based on relational model introduced by E.F. Codd. During 1970-1972, E.F. Codd published a paper to propose the use of relational database model. RDBMS is originally based on that E.F. Codd's relational model invention.

A relational database is a type of DBMS. It uses a structure that allows us to identify and access data in relation to another piece of data in the database. In RDBMS, data is organized into tables. Each table has its own primary key. Tables can grow large and have a multitude of columns and records. Data is represented in terms of tuples (rows) in table. Due to a collection of organized set of tables, data can be accessed easily in RDBMS. In RDBMS terminology, database also known as schema.

The RDBMS database uses tables to store data. A table is a collection of related data entries and contains rows and columns to store data. The RDBMS database may contains one or more tables.

Let's examine the Employee table.

Employee_ID	Name	Designation	Date_of_Join
1	Ramesh	Manager	12-12-2018
2	Rajesh	Sr. Supervisor	02-12-2013
3	Vijay	Worker	01-03-2015

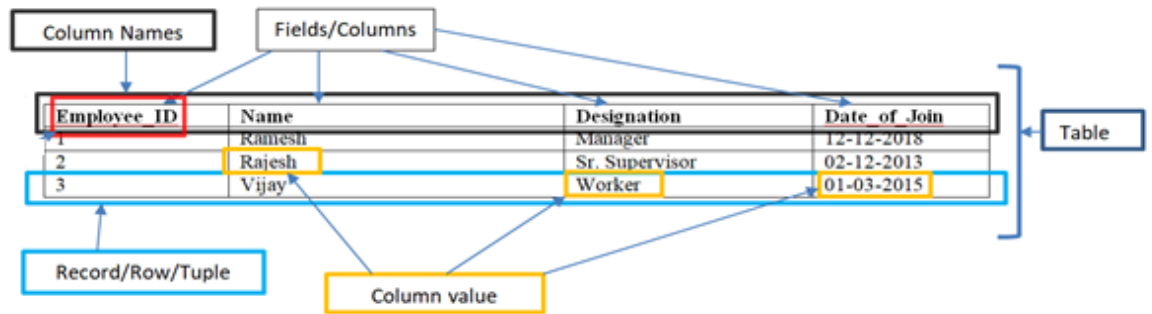


Figure 1: Table Example

Tuple/Row/Record:

In RDBMS terminology a row of a table is also known as tuple or record. It contains the specific data of each individual entry in the table. It is a horizontal entity in the table. For example: The above Employee table contains 3 records and four columns.

Field/Column:

Field is a homogenous entity of the table that contains specific data about every tuple/row in the table. A field/column is a vertical entity in the table. For example, Employee_ID, Name, Designation and Date_of_Join are known as fields or columns.

Here, column called Employee_ID may have type INTEGER (representing the type of data it is meant to hold).

The Name and Designation column store character data types, whereas Date_of_Join store date data type.

Data Integrity

There are the following categories of data integrity exist with each RDBMS:

Entity integrity: It specifies that there should be no duplicate rows in a table.

E.g. in Figure-1, Employee_ID value must be unique.

Domain: Domain means it is a pool of legal values

E.g. in Figure-1, Name is domain of employee's name. So legal values of employee name are "Ramesh", "Vijay" etc... While, "Rajesh123" is illegal employee name.

Domain integrity: It enforces valid entries for a given column by restricting the type, the format, or the range of values.

E.g. here employee **name** contain 20 character alphabets only.

Referential integrity: It specifies that rows cannot be deleted, which are used by other records.

User-defined integrity: It enforces some specific business rules that are defined by users. These rules are different from entity, domain or referential integrity.

Primary Key: A primary key is a single field or combination of fields that contains a unique record. It must be filled. None of the field of primary key can contain a null value. A table can have only one primary key.

Foreign Key: A foreign key is a field or a column that is used to establish a link between two tables. In simple words you can say that, a foreign key in one table used to point primary key in another table.

Unique Key: A unique key is a set of one or more than one fields/columns of a table that uniquely identify a record in a database table.

DBMS and RDBMS both are used to store information in physical database but there are some remarkable differences between them.

Why RDBMS?

- RDBMS applications store data in a database
- Database contains one or more tables.
- Normalization is present in RDBMS.
- RDBMS defines the integrity constraint for the purpose of ACID (Atomocity, Consistency, Isolation and Durability) property.
- RDBMS system supports a tabular structure of the data and a relationship between them to access the stored information
- In RDBMS, the tables have an identifier called primary key and the data values are stored in the form of tables.

- In RDBMS, data values are stored in the form of tables, so a relationship between these data values will be stored in the form of a table as well.
- RDBMS supports distributed database.
- RDBMS is designed to handle large amount of data. It supports multiple users.

1.4 INTRODUCTION TO MySQL

MySQL is a fast, easy to use, and open source relational database management system. MySQL Community Edition is the freely downloadable version of the open source version of MySQL database. It is available under the GPL license and is supported by a huge and active community of open source developers. It is very commonly used in conjunction with PHP scripts to create powerful and dynamic server-side applications.

1.5 WHY MySQL?

MySQL is becoming so popular because of these following reasons:

- MySQL (CE-Community Edition) is an open-source database that is free to use and you can download it from MySQL official website. **<https://www.mysql.com/products/community/>**
- It is a relational database management system.
- It is customizable because it is an open source database and the open-source GPL license facilitates programmers to modify the SQL software according to their requirement.
- It is a very powerful; it come with a large set of functionality.
- It is quicker than other databases.
- It supports many operating systems and many languages like PHP, PERL, C, C++, JAVA, etc.
- It uses a standard form of the well-known SQL data language.

- It is very friendly with PHP, the most popular language for web development.
- It supports large databases.
- It supports Transitions Management

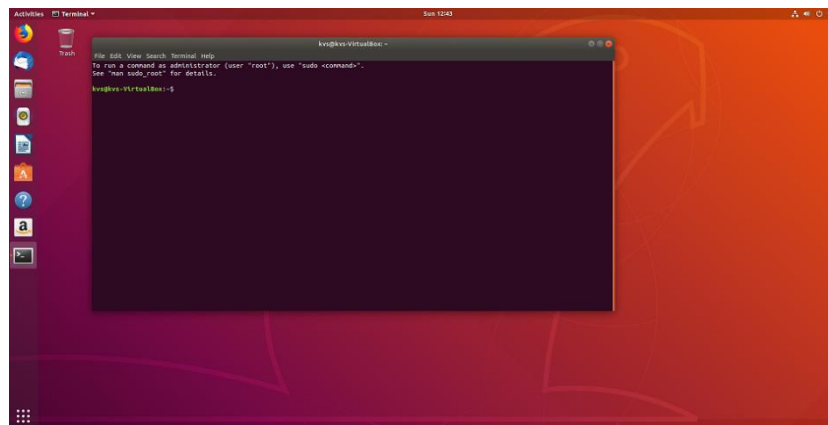
1.6 HOW TO INSTALL AND CONFIGURE MySQL

Following section describes a basic installation of a MySQL database server on Ubuntu Linux. You might need to install other packages to let applications use MySQL, like extensions for PHP. Check your application documentation for details.

MySQL Installation Steps:

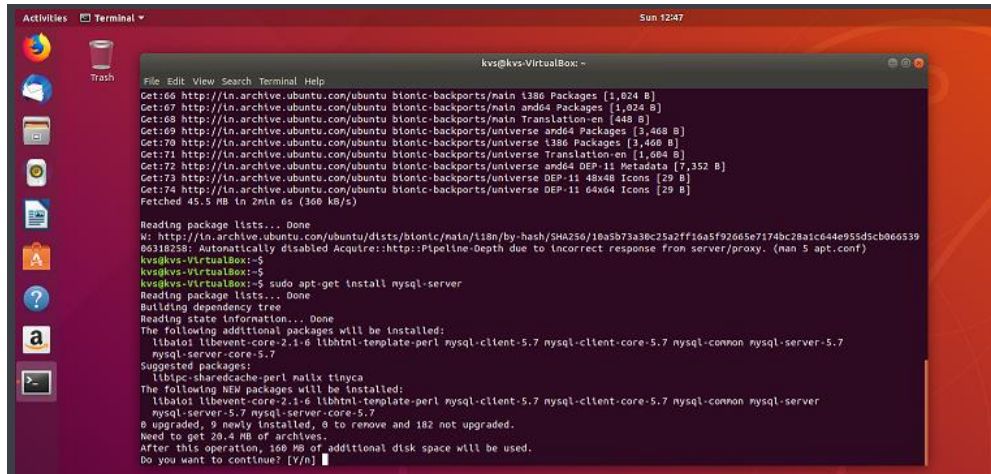
Install the MySQL server by using the Ubuntu package manager:

Step -1 to install MySQL, open terminal window by pressing ctrl + Alt + T key and run following commands



```
sudo apt-get update
```

```
sudo apt-get install mysql-server
```



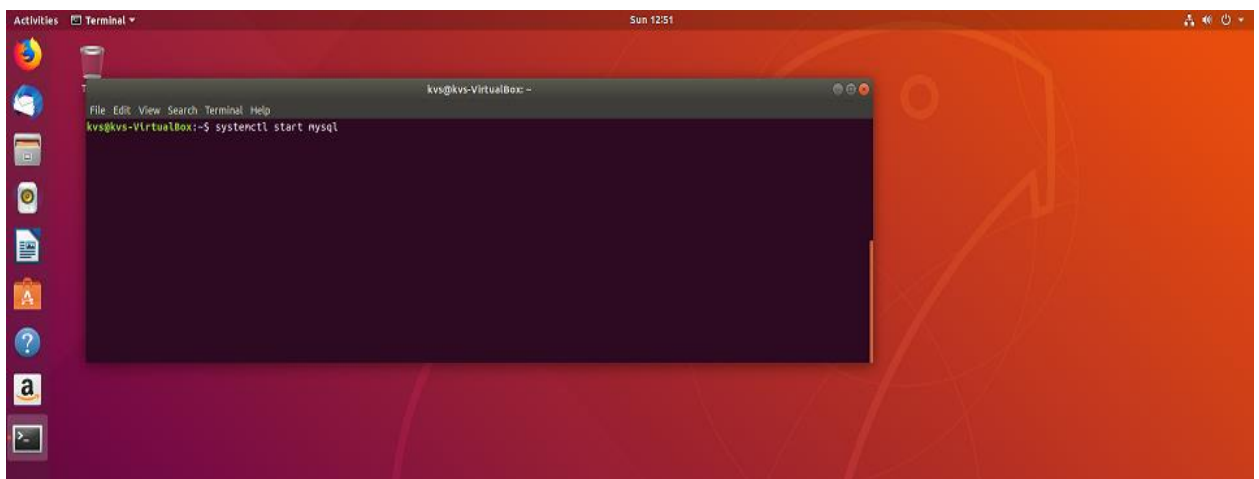
Press 'Y' to start installation. The installer installs MySQL and all dependencies.

After installation is complete, the `mysql_secure_installation` utility runs. This utility prompts you to define the mysql root password and other security related options, including removing remote access to the root user and setting the root password.

Step-2 Start the MySQL Service

After the installation is complete, you can start the database service by running the following command. If the service is already started, a message informs you that the service is already running:

```
systemctl start mysql
```



Step-3 Launch at boot/startup

To ensure that the database server launches after a reboot, run the following command:

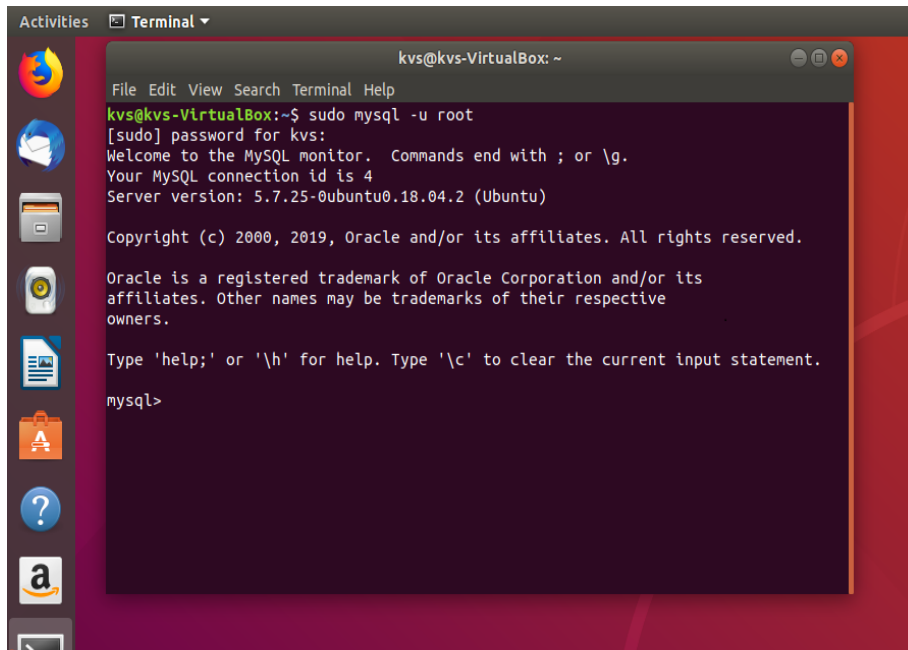
```
systemctl enable mysql
```

Step-4 Start MySQL shell

There is more than one way to work with a MySQL server, but this article focuses on the most basic and compatible approach, the `mysql` shell.

1. At the command prompt, run the following command to launch the `mysql` shell and enter it as the root user:

```
/usr/bin/mysql -u root
```



2. When you're prompted for a password, enter the one that you set at Ubuntu installation time, or if you haven't set one, press **Enter** to submit no password.

The following `MySQL` shell prompt should appear:

```
mysql>
```

At MySQL shell prompt you can run various SQL statements

1.7MySQL DATATYPES

A Data Type denoting a particular type of data, like integer, floating points, Boolean etc. It also identifies the possible values that the type can hold, the operations that can be performed on that type and the way the values of that type are stored.

MySQL data types mainly divide into three categories: numeric, date and time, and string.

Numeric Data Types:

Data type	Description
INT	A normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. You can specify a width of up to 11 digits.
TINYINT	A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify a width of up to 4 digits.
SMALLINT	A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. You can specify a width of up to 5 digits.
MEDIUMINT	A medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. You can specify a width of up to 9 digits.
BIGINT	A large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 20 digits.
FLOAT(m,d)	A floating-point number that cannot be unsigned. You can define

	the display length (m) and the number of decimals (d). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a float.
DOUBLE(m,d)	A double precision floating-point number that cannot be unsigned. You can define the display length (m) and the number of decimals (d). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a double. Real is a synonym for double.
DECIMAL(m,d)	An unpacked floating-point number that cannot be unsigned. In unpacked decimals, each decimal corresponds to one byte. Defining the display length (m) and the number of decimals (d) is required. Numeric is a synonym for decimal.

Date and Time Data Type:

Data Type	Maximum Size	Explanation
DATE	Values range from '1000-01-01' to '9999-12-31'.	Displayed as 'yyyy-mm-dd'.
DATETIME	Values range from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.	Displayed as 'yyyy-mm-dd hh:mm:ss'.
TIMESTAMP(m)	Values range from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' TC.	Displayed as 'YYYY-MM-DD HH:MM:SS'.
TIME	Values range from '-838:59:59' to '838:59:59'.	Displayed as 'HH:MM:SS'.
YEAR[(2 4)]	Year value as 2 digits or 4 digits.	

Data Type	Maximum Size	Explanation
CHAR(size)	Maximum size of 255 characters.	Where size is the number of characters to store. Fixed-length strings. Space padded on right to equal size characters.
VARCHAR(size)	Maximum size of 255 characters.	Where size is the number of characters to store. Variable-length string.
TINYTEXT(size)	Maximum size of 255 characters.	Where size is the number of characters to store.
TEXT(size)	Maximum size of 65,535 characters.	Where size is the number of characters to store.
MEDIUMTEXT(size)	Maximum size of 16,777,215 characters.	Where size is the number of characters to store.
LONGTEXT(size)	Maximum size of 4GB or 4,294,967,295 characters.	Where size is the number of characters to store.
BINARY(size)	Maximum size of 255 characters.	Where size is the number of binary characters to store. Fixed-length strings. Space padded on right to equal size characters.
VARBINARY(size)	Maximum size of 255 characters.	Where size is the number of characters to store. Variable-length string.

Large Object Data Types (LOB) Data Types:

Data Type	Maximum Size
TINYBLOB	Maximum size of 255 bytes.
BLOB(size)	Maximum size of 65,535 bytes.
MEDIUMBLOB	Maximum size of 16,777,215 bytes.
LONGTEXT	Maximum size

1.8 CHECK YOUR PROGRESS

- 1) What is RDBMS?
- 2) What is database?
- 3) Which MySQL version is free to use and open source
- 4) How to shutdown MySQL server

1.9 CHECK YOUR PROGRESS: POSSIBLE ANSWERS

- 1) A **Relational Database Management System (RDBMS)** is a software that
 - Allow you to implement a database with tables, columns and indexes.
 - Guarantees the Referential Integrity between rows of various tables.
 - Updates the indexes automatically.
 - Interprets an SQL query and combines information from various tables.
- 2) Database is a collection of tables, with related data or in other word it is repository of data.
- 3) MySQL Community Edition
- 4) `sudo mysqladmin -u root -p shutdown`

Unit 2: Structured Query Language

2

Unit Structure

2.1 Learning Objectives and Outcome

2.2 Introduction

2.3 What is SQL?

2.4 SQL Commands

2.5 Check your Progress

2.6 Check your Progress: Possible Answers

2.1 LEARNING OBJECTIVES AND OUTCOME

After studying this unit student will be able to:

- Understand SQL and SQL syntax used with MySQL
- Concepts of unique key, primary key, foreign key and table constraints.
- How to retrieve and manipulates data from one or more tables through queries
- How to filter data based upon filter conditions

Outcome:

- Install, manage and work with MySQL
- Understand and explain the roles RDBMS
- Understand and use of the Structured Query Language (SQL)
- Write DDL statement to Create, modify and delete database/ schema, tables, and indexes
- MySQL datatypes and unique key, primary key, foreign key and table constraints.
- Write DML statement to create, modify and delete tables and Retrieve data from the tables through queries and sub-queries.

2.2 INTRODUCTION

SQL is an open standard language to work similarly with most the databases. SQL consist of three built-in sub-languages: Data definition language (DDL), Data manipulation language (DML) and Data control language (DCL). SQL is a fourth generation language. SQL has many more features and advantages. Let us discuss the SQL in more detail in this chapter.

2.3 WHAT IS SQL?

SQL (Structured Query Language) is a programming language used to communicate with a relational database management system. SQL syntax is similar to the English language, which makes it relatively easy to write, read, and interpret. Many RDBMSs use SQL (and variations of SQL) to access the data in tables. For

example, SQLite is a relational database management system. SQLite contains a minimal set of SQL commands (which are the same across all RDBMSs). Other RDBMSs may use other variants. SQL is often pronounced in one of two ways. You can pronounce it by speaking each letter individually like “S-Q-L”, or pronounce it using the word “sequel”. SQL open standard language. All the relational systems support SQL, thus allowing migration of database from one DBMS to another. This feature is commonly referred to as portability. Although, SQL is portable but SQL syntax may differ slightly depending on which RDBMS you are using.

Some of the important features of SQL are:

- SQL is a non procedural language.
- SQL is open standard language.
- SQL is an English-like language.
- SQL can process a single record as well as sets of records at a time.
- SQL is different from a third generation language
- SQL is a data sub-language consisting of three built-in languages: Data definition language (DDL), Data manipulation language (DML) and Data control language (DCL).
- It insulates the user from the underlying structure and algorithm of databases.

SQL has facilities for defining database views, security, integrity constraints, transaction.

2.4 SQL COMMANDS

There are three types of built-in SQL commands: Data definition language (DDL), Data manipulation language (DML) and Data control language (DCL). SQL commands are instructions to RDBMS. It is used to communicate with the database. It is also used to perform specific tasks, functions, and queries of data. SQL can perform various operations like create a table, add data to tables, drop the table, modify the table, and set permission for users.

2.4.1 DATA DEFINITION LANGUAGE (DDL)

As discussed in the previous section, the basic storage unit of a relational database management system is a database/schema, within database data is organized into tables. Each table has its own primary key. Tables can grow large and have a multitude of columns and records. Data is represented in terms of tuples (rows) in table. Due to a collection of organized set of tables, data can be accessed easily in RDBMS. In RDBMS terminology, database also known as schema.

The Data definition language (DDL) defines a set of commands used in the creation and modification of schema and objects such as tables, indexes, views etc. These commands provide the ability to create, alter and drop these objects. These commands are related to the management and administrations of the databases. Before and after each DDL statement, the current transactions are implicitly committed, that is changes made by these commands are permanently stored in the databases. Let us discuss these commands in more detail:

Create a database in MySQL:

There is a difference between a *database server* and a *database*, even though those terms are often used interchangeably. MySQL is a database server, meaning it tracks databases and controls access to them. The database stores the data, and it is the database that applications are trying to access when they interact with MySQL.

List of DDL Statements/Commands:

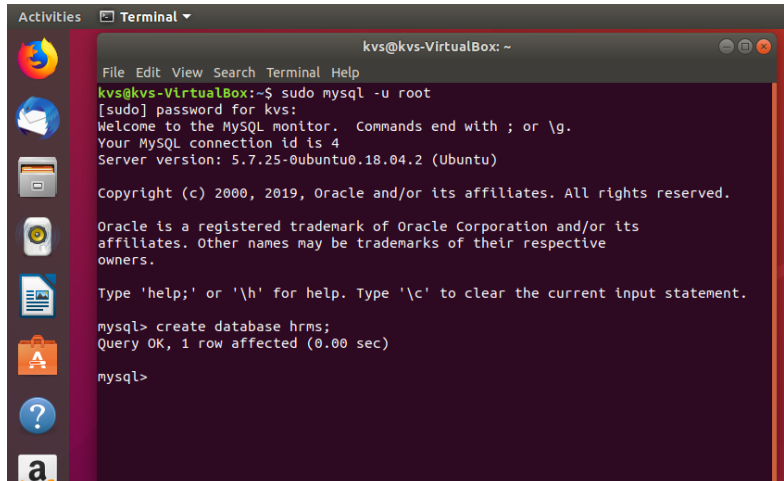
DDL Command	Description
CREATE	To create database / objects in the database
ALTER	To alters the structure of the database
DROP	To delete database / objects from the database
TRUNCATE	To remove all records from a table, including all spaces allocated for the records are removed
COMMENT	To add comments to the data dictionary
RENAME	rename an object

To create a database, log in to the `mysql` shell and run the following command, replacing `<database-name>` with actual name of the database that you want to create:

```
CREATE DATABASE <database-name>;
```

Example: Create database `hrms`;

Here, Create database is a command to create a database and `hrms` is a database name



```
Activities Terminal
kvs@kvs-VirtualBox: ~
File Edit View Search Terminal Help
kvs@kvs-VirtualBox:~$ sudo mysql -u root
[sudo] password for kvs:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.25-0ubuntu0.18.04.2 (Ubuntu)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database hrms;
Query OK, 1 row affected (0.00 sec)

mysql>
```

Naming Conventions for MySQL/MariaDB

MySQL database identifiers that you can name include databases, tables, columns, other database objects, and alias. They follow these naming conventions.

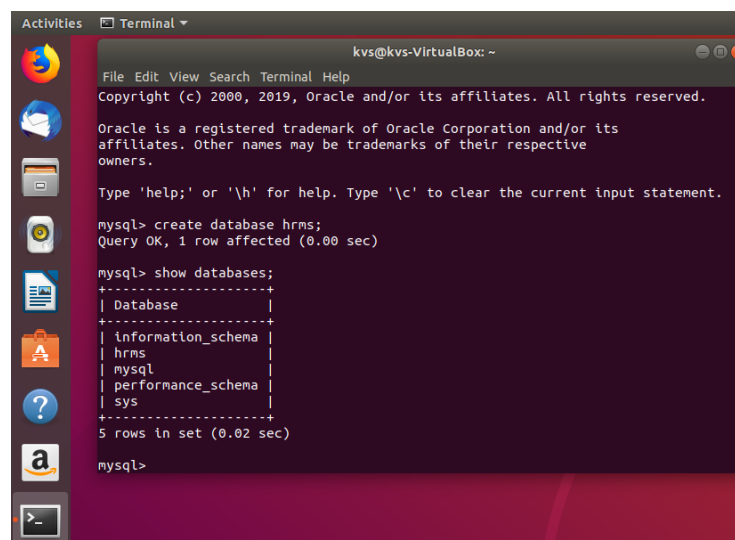
- Names must be from 1 to 255 characters long. All other identifier names must be from 1 to 64 characters long.
- Database names can use any character that is allowed in a directory name except for a period, a backward slash (`\`), or a forward slash (`/`).
- By default, MySQL encloses column names and table names in quotation marks.
- Table names can use any character that is allowed in a filename except for a period or a forward slash.
- Column names and alias names allow all characters.
- Embedded spaces and other special characters are not permitted unless you enclose the name in quotation marks.
- Embedded quotation marks are not permitted.

- Case sensitivity is set when a server is installed. By default, the names of database objects are case sensitive on UNIX and not case sensitive on Windows. For example, the names **CUSTOMER** and **Customer** are different on a case-sensitive server.
- A name cannot be a reserved word in MySQL unless you enclose the name in quotation marks. See the MySQL documentation for more information about reserved words.
- Database names must be unique. For each user within a database, names of database objects must be unique across all users. For example, if a database contains a department table that User A created, no other user can create a department table in the same database.

Note: MySQL does not recognize the notion of schema, so tables are automatically visible to all users with the appropriate privileges. Column names and index names must be unique within a table. For detailed information about naming conventions, see your MySQL documentation.

After the database is created, you can verify its creation by run a query **show databases** to list all databases. The following example shows the query and example output:

SHOW databases;



```
Activities Terminal
kvs@kvs-VirtualBox: ~
File Edit View Search Terminal Help
Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> create database hrms;
Query OK, 1 row affected (0.00 sec)
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| hrms |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.02 sec)
mysql>
```


Create Table:

The CREATE TABLE statement is used to create a new table in a database.

```
CREATE TABLE <<tablename>> (  
                                column1                datatype,  
                                column2                datatype,  
                                column3                datatype,  
                                ....  
);
```

The column parameters specify the names of the columns of the table.

The datatype parameter specifies the type of data the column can hold (e.g. varchar, integer, date, etc.).

Rules for creation of table:

- Table name should start with an alphabet.
- In table name, blank spaces and single quotes are not allowed.
- Reserve words of that DBMS cannot be used as table name.
- Proper data types and size should be specified.
- Unique column name should be specified.

Column Constraints: NOT NULL, UNIQUE, PRIMARY KEY, CHECK, DEFAULT, REFERENCES

For example,

```
CREATE TABLE hrms.employees (emp_id int PRIMARY KEY, emp_name  
varchar(15) NOT NULL, job_name varchar(10), manager_id int, hire_date date,  
salary decimal(10,2), commission float(7,2), dep_id int);
```

The command above creates a table. Primary key constraint (Discussed in data integrity section of chapter-1) ensures that emp_id is not null and unique (both are the properties of primary key). Varchar basically is variable length character type

subject to a maximum specified in the declarations. We will use them at most of the places.

```
mysql> CREATE TABLE hrms.employees (emp_id int PRIMARY KEY, emp_name varchar(15) NOT NULL, job_name varchar(10),
manager_id int, hire_date date, salary decimal(10,2), commission float(7,2), dep_id int);
Query OK, 0 rows affected (0.59 sec)

mysql> _
```

To view Structure of table:

Query for view/describe table structure as follows:

Desc <databasename>.<tablename>;

Example,

Desc hrms.employee;

```
mysql> desc hrms.employees;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| emp_id     | int(11)       | NO   | PRI | NULL    |       |
| emp_name   | varchar(15)   | NO   |     | NULL    |       |
| job_name   | varchar(10)   | YES  |     | NULL    |       |
| manager_id | int(11)       | YES  |     | NULL    |       |
| hire_date  | date          | YES  |     | NULL    |       |
| salary     | decimal(10,2) | YES  |     | NULL    |       |
| commission | float(7,2)    | YES  |     | NULL    |       |
| dep_id     | int(11)       | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
8 rows in set (0.05 sec)

mysql> _
```

Similarly, create another table department in hrmsdatabase :

```
CREATE TABLE hrms.department ( dep_id int, dep_name varchar(20),
dep_location varchar(15));
```

ALTER TABLE: This SQL statement is used for modification of existing structure of the table in the following situation:

- When a new column is to be added to the table structure.
- When the existing column definition has to be changed, i.e., changing the width of the data type or the data type itself.
- When integrity constraints have to be included or dropped.
- When a constraint has to be enabled or disabled.

Following are various syntax of it.

ALTER TABLE <table name> ADD (<column name><data type>...);

ALTER TABLE <table name> MODIFY (<column name><data type>...);

ALTER TABLE <table name> ADD CONSTRAINT <constraint name> (field name);

ALTER TABLE <table name> DROP<constraint name>;

ALTER TABLE <table name> ENABLE/DISABLE <constraint name>;

The structure of department table is:

```
mysql> desc department;
+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default |
+-----+-----+-----+-----+-----+
| dep_id         | int(11)       | NO   |     | 0        |
| dep_name       | varchar(20)   | YES  |     | NULL     |
| dep_location   | varchar(15)   | YES  |     | NULL     |
+-----+-----+-----+-----+-----+
```

Now, if we want to make primary key to dep_id then,

Alter table hrms.department Add constraint Primary Key (dep_id);

```
mysql> Alter table hrms.department Add constraint Primary Key (dep_id);
Query OK, 0 rows affected (0.56 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql> _
```

After executing above command the structure of department table changed as:

```
mysql> desc department;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| dep_id         | int(11)       | NO   | PRI | 0        |       |
| dep_name       | varchar(20)   | YES  |     | NULL     |       |
| dep_location   | varchar(15)   | YES  |     | NULL     |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

DROP TABLE:

When an existing table is not required for further use, it is always better to remove it from the database. To delete the existing table from the database the following command is used.

```
DROP TABLE <table name>;
```

```
Drop table hrms.department;
```

```
mysql> Drop table hrms.department;  
Query OK, 0 rows affected (0.11 sec)
```

2.4.2 DATA MANIPULATION LANGUAGE

Data manipulation language (DML) defines a set of commands that are used to query and modify data within existing database tables. In this case commit is not implicit that is changes are not permanent till explicitly committed. DML statements consist of SELECT, INSERT, UPDATE and DELETE statements

List of DML Statements/Commands:

DML Command	Description
SELECT	To retrieve data from the a database
INSERT	To insert data into a table
DELETE	To deletes all or specific records from a table
UPDATE	To updates existing data within a table

INSERT INTO Statement:

This SQL statement is used for inserting data into existing table.

This statement used in the following situation:

- Values can be inserted for all columns or for the selected columns.
- Values can be given through sub query.
- In place of values parameter substitution can also be used with insert.

- If data is not available for all the columns, then the column list must be included following the table name.

```
INSERT INTO <tablename>(column1, column2, column3,...columnN) VALUES
(value1, value2, value3,...valueN);
```

For example, insert data into employee table

```
mysql> Insert into hrms.employees
-> Values(66928, 'Vijay', 'MANAGER', 68319, '1991-11-11', 6000.00, 400.00, 1001);
Query OK, 1 row affected (0.05 sec)
```

OR

```
mysql> Insert into hrms.employees (emp_id, emp_name, job_name, manager_id, hire_date, salary, commission, dep_id)
-> Values(66928, 'Vijay', 'MANAGER', 68319, '1991-11-11', 6000.00, 400.00, 1001);
Query OK, 1 row affected (0.05 sec)
```

If you want to insert some fields than you must have to specify filed list.

```
mysql> Insert into hrms.employees (emp_id, emp_name, job_name, hire_date, salary, dep_id)
-> Values(68319, 'Rajesh', 'RESIDENT', '1991-11-11', 60000.00, 1001);
Query OK, 1 row affected (0.06 sec)
```

Finally, table contains following records

```
mysql> select * from hrms.employees;
+-----+-----+-----+-----+-----+-----+-----+-----+
| emp_id | emp_name | job_name | manager_id | hire_date | salary | commission | dep_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 66928 | Vijay | MANAGER | 68319 | 1991-11-11 | 6000.00 | 400.00 | 1001 |
| 68319 | Rajesh | PRESIDENT | NULL | 1991-11-11 | 6000.00 | NULL | 1001 |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Populate one table using another table

You can populate the data into a table through the select statement over another table; provided the other table has a set of fields, which are required to populate the first table.

Here is the syntax:

```
INSERT INTO first_<tablename> [(column1, column2, ... columnN)] SELECT
column1, column2, ...columnN
```

```
FROM second_<tablename> [WHERE condition];
```

UPDATE Statement:

The SQL UPDATE Query is used to update the existing records in a table. You can use the WHERE clause with the UPDATE query to update the selected rows, otherwise all the rows would be affected.

```
UPDATE <<tablename>> SET column1 = value1, column2 = value2...., columnN = valueN WHERE [condition];
```

The following query will update the salary for a employee whose emp_id is 68319 in the table.

```
mysql> Update hrms.employees SET salary = 70000.00 Where emp_id = 68319;
Query OK, 1 row affected (0.06 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Now,

```
mysql> select * from hrms.employees;
+-----+-----+-----+-----+-----+-----+-----+-----+
| emp_id | emp_name | job_name | manager_id | hire_date | salary | commission | dep_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 66928 | Vijay | MANAGER | 68319 | 1991-11-11 | 6000.00 | 400.00 | 1001 |
| 68319 | Rajesh | PRESIDENT | NULL | 1991-11-11 | 70000.00 | NULL | 1001 |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

DELETE Statement:

The SQL DELETE Query is used to delete the existing records from a table. You can use the WHERE clause with a DELETE query to delete the selected rows, otherwise all the records would be deleted.

The basic syntax of the DELETE queries with the WHERE clause is as follows:

```
DELETE FROM <tablename> WHERE [condition];
```

You can combine N number of conditions using AND or OR operators.

```
mysql> Delete from hrms.employees Where emp_id = 66928;
Query OK, 1 row affected (0.05 sec)

mysql> select * from hrms.employees;
+-----+-----+-----+-----+-----+-----+-----+-----+
| emp_id | emp_name | job_name | manager_id | hire_date | salary | commission | dep_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 68319 | Rajesh | PRESIDENT | NULL | 1991-11-11 | 70000.00 | NULL | 1001 |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

SQL Operators:

With the SQL statement you also use operators. An operator used with WHERE clause to perform operation(s) in SQL, e.g. comparisons and arithmetic operators.

SQL Operators are:

Arithmetic Operators:

Operators	Meaning
+	Addition (E.g. a + b)
-	Subtraction (E.g. a - b)
*	Multiplication (E.g. a * b)
/	Division (E.g. a / b)
%	Modulus (E.g. a % b)

Comparison operators:

Operators	Meaning
=	equal to (E.g. a = b)
!= <>	not equal to (E.g. a !=b or a <> b)
>	greater than (E.g. a > b)
<	less than (E.g. a < b)
>=	greater than or equal (E.g. a >= b)
<=	(less than or equal (E.g. a <= b)

For example, employee tables contains following data

emp_id	emp_name	job_name	manager_id	hire_date	salary	commission	dep_id
66926	Kamal	MANAGER	68320	2000-01-01	8000.00	800.00	1001
66928	Vijay	MANAGER	68319	1991-11-11	6000.00	400.00	1001
66929	Vijay	CLERK	68319	2002-10-01	5000.00	600.00	1001
66930	Ajay	CLERK	68320	2002-10-01	5000.00	200.00	1002
68319	Rajesh	PRESIDENT	NULL	1991-11-11	70000.00	NULL	1001

Now we want summation of salary and commission then,

```
mysql> select salary + commission as "Total Earning" from employees;
+-----+
| Total Earning |
+-----+
|      8800.00 |
|      7600.00 |
|      6400.00 |
|      5600.00 |
|      5200.00 |
|          NULL |
+-----+
6 rows in set (0.02 sec)
```

Now if you want to calculate 20% income tax on salary then

```
mysql> select emp_id, emp_name, (salary * 20 / 100) as "Income Tax" from employees;
+-----+-----+-----+
| emp_id | emp_name | Income Tax |
+-----+-----+-----+
| 66926 | Kamal    | 1600.000000 |
| 66927 | Amar     | 1400.000000 |
| 66928 | Vijay    | 1200.000000 |
| 66929 | Vijay    | 1000.000000 |
| 66930 | Ajay     | 1000.000000 |
| 68319 | Rajesh   | 14000.000000 |
+-----+-----+-----+
6 rows in set (0.00 sec)
```

Logical and other operators:

Operators	Meaning
ALL	This operator is used to compare a value to all values in another value set.
AND	AND – This operator allows the existence of multiple conditions in an SQL statement's WHERE clause.
ANY	ANY - This operator is used to compare a value to any applicable value in the list as per the condition
BETWEEN	This operator is used to search for values that are within a set of values, given the minimum value and the maximum value.
EXISTS	This operator is used to search for the presence of a row in a specified table that meets a certain criterion.

IN	This operator is used to compare a value to a list of literal values that have been specified.
LIKE	This operator is used to compare a value to similar values using wildcard operators. is used to compare a value to a list of literal values that have been specified.
NOT	This operator reverses the meaning of the logical operator with which it is used. Eg: NOT
NOT BETWEEN, NOT IN, etc.	This is a negate operator
OR	This operator is used to combine multiple conditions in an SQL statement's WHERE clause
IS NULL	This operator is used to compare a value with a NULL value
UNIQUE	This operator searches every row of a specified table for uniqueness (no duplicates).

SELECT Statement:

The SELECT statement is used for retrieving information from the databases tables. It can be used with many clauses.

Note: Discuss the other SQL operators in with more detail later in this chapter.

Simple SELECT statement:

SELECT * FROM<tablename>;

Here, * means all fields. To fetch all the fields available in the field, then you can use *

```
mysql> select * from hrms.employees;
+-----+-----+-----+-----+-----+-----+-----+-----+
| emp_id | emp_name | job_name | manager_id | hire_date | salary | commission | dep_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 68319 | Rajesh | PRESIDENT | NULL | 1991-11-11 | 70000.00 | NULL | 1001 |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

SELECT column1, column2....columnN FROM <tablename>;

Here, column1, column2... are the fields of a table whose values you want to fetch

```
mysql> select emp_id,emp_name from hrms.employees;
+-----+-----+
| emp_id | emp_name |
+-----+-----+
| 66928  | Vijay    |
| 68319  | Rajesh   |
+-----+-----+
2 rows in set (0.00 sec)
```

SELECT DISTINCT:

The DISTINCT keyword is used in conjunction with the SELECT statement to eliminate all the duplicate records and fetching only unique records

SELECT DISTINCT column1, column2,columnN FROM <tablename>;

```
mysql> select DISTINCT emp_name from hrms.employees;
+-----+
| emp_name |
+-----+
| Vijay    |
| Rajesh   |
+-----+
```

SELECT with WHERE clause:

WHERE clause is used to specify a condition while fetching the data from a single table or by joining with multiple tables. You should use the WHERE clause to filter the records and fetching only the necessary records.

The WHERE clause is also used with INSERT, UPDATE and DELETE statements.

SELECT column1, column2....columnN FROM<tablename> WHERE [condition];

```
mysql> select * from hrms.employees where emp_id = 68319;
+-----+-----+-----+-----+-----+-----+-----+-----+
| emp_id | emp_name | job_name | manager_id | hire_date | salary | commission | dep_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 68319  | Rajesh   | PRESIDENT | NULL       | 1991-11-11 | 70000.00 | NULL       | 1001 |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

```
mysql> select * from hrms.employees where emp_name= 'vijay';
+-----+-----+-----+-----+-----+-----+-----+-----+
| emp_id | emp_name | job_name | manager_id | hire_date | salary | commission | dep_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 66928  | Vijay    | MANAGER  | 68319     | 1991-11-11 | 6000.00 | 400.00     | 1001 |
| 66929  | Vijay    | CLERK    | 68319     | 2002-10-01 | 5000.00 | 600.00     | 1001 |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)
```

SELECT with WHERE clause and logical operators:

SELECT column1, column2....columnN FROM <tablename> WHERE CONDITION-1 {AND|OR} CONDITION-2;

The AND & OR operators are used to combine multiple conditions to narrow result data in SQL statement. These two operators are called as the conjunctive operators.

Consider the employees table having the following records:

The AND operator allows the existence of multiple conditions with WHERE clause.

Following is an example, which would fetch the all the fields from the employee table, where the salary is greater than 6000 and the department id (dept_id) is equals to 1001

```
mysql> select * from hrms.employees where dep_id = 1001 and salary > 6000;
+-----+-----+-----+-----+-----+-----+-----+-----+
| emp_id | emp_name | job_name | manager_id | hire_date | salary | commission | dep_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 66926 | Kamal | MANAGER | 68320 | 2000-01-01 | 8000.00 | 800.00 | 1001 |
| 68319 | Rajesh | PRESIDENT | NULL | 1991-11-11 | 70000.00 | NULL | 1001 |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

The following example has a query, which would fetch the all fields from the employees table, where the department id (dept_id) is equals to 1001 OR the salary is greater than 6000

```
mysql> select * from hrms.employees where dep_id = 1001 or salary > 6000;
+-----+-----+-----+-----+-----+-----+-----+-----+
| emp_id | emp_name | job_name | manager_id | hire_date | salary | commission | dep_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 66926 | Kamal | MANAGER | 68320 | 2000-01-01 | 8000.00 | 800.00 | 1001 |
| 66928 | Vijay | MANAGER | 68319 | 1991-11-11 | 6000.00 | 400.00 | 1001 |
| 66929 | Vijay | CLERK | 68319 | 2002-10-01 | 5000.00 | 600.00 | 1001 |
| 68319 | Rajesh | PRESIDENT | NULL | 1991-11-11 | 70000.00 | NULL | 1001 |
+-----+-----+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

SELECT with WHERE IN clause:

SELECT column1, column2....columnN FROM<tablename> WHERE column_name IN (val-1, val-2,...val-N);

The following example has a query, which would fetch the record of employees who are the part of the department (1001, 1003)

```
mysql> select * from hrms.employees Where dep_id IN (1001,1003);
```

emp_id	emp_name	job_name	manager_id	hire_date	salary	commission	dep_id
66926	Kamal	MANAGER	68320	2000-01-01	8000.00	800.00	1001
66927	Amar	SALESMAN	68326	2000-02-10	7000.00	600.00	1003
66928	Vijay	MANAGER	68319	1991-11-11	6000.00	400.00	1001
66929	Vijay	CLERK	68319	2002-10-01	5000.00	600.00	1001
68319	Rajesh	PRESIDENT	NULL	1991-11-11	70000.00	NULL	1001

```
5 rows in set (0.00 sec)
```

LIKE clause:

SELECT column1, column2....columnN FROM<tablename> WHERE column_name LIKE { PATTERN };

The **LIKE** clause is used to compare a value to similar values using wildcard operators. There are two wildcards used in conjunction with the LIKE operator.

- The percent sign (%)
- The underscore (_)

The percent sign represents zero, one or multiple characters. The underscore represents a single number or character. These symbols can be used in combinations.

Syntax of % and _ is as follows

SELECT FROM <tablename>WHERE column LIKE 'XXXX%'

SELECT FROM <tablename>WHERE column LIKE '%XXXX%'

SELECT FROM <tablename>WHERE column LIKE 'XXXX_'

SELECT FROM <tablename> WHERE column LIKE '_XXXX'

SELECT FROM <tablename> WHERE column LIKE '_XXXX_'

```
mysql> select * from hrms.employees Where emp_name like '_ijay';
```

emp_id	emp_name	job_name	manager_id	hire_date	salary	commission	dep_id
66928	Vijay	MANAGER	68319	1991-11-11	6000.00	400.00	1001
66929	Vijay	CLERK	68319	2002-10-01	5000.00	600.00	1001

```
2 rows in set (0.00 sec)
```

ORDER BY clause:

The ORDER BY clause is used to sort the data in ascending or descending order, based on one or more columns. Most of databases sort the query results in an ascending order by default.

SELECT column1, column2....columnN FROM<tablename> WHERE CONDITION ORDER BY column_name {ASC|DESC};

The following example display all the record in ascending order of employee name of employees table

```
mysql> select * from hrms.employees order by emp_name;
+-----+-----+-----+-----+-----+-----+-----+-----+
| emp_id | emp_name | job_name | manager_id | hire_date | salary | commission | dep_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 66930 | Ajay | CLERK | 68320 | 2002-10-01 | 5000.00 | 200.00 | 1002 |
| 66927 | Amar | SALESMAN | 68326 | 2000-02-10 | 7000.00 | 600.00 | 1003 |
| 66926 | Kamal | MANAGER | 68320 | 2000-01-01 | 8000.00 | 800.00 | 1001 |
| 68319 | Rajesh | PRESIDENT | NULL | 1991-11-11 | 70000.00 | NULL | 1001 |
| 66928 | Vijay | MANAGER | 68319 | 1991-11-11 | 6000.00 | 400.00 | 1001 |
| 66929 | Vijay | CLERK | 68319 | 2002-10-01 | 5000.00 | 600.00 | 1001 |
+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.12 sec)
```

The following query display all the record in descending order of employee name of employees table

```
mysql> select * from hrms.employees order by emp_name desc;
+-----+-----+-----+-----+-----+-----+-----+-----+
| emp_id | emp_name | job_name | manager_id | hire_date | salary | commission | dep_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 66928 | Vijay | MANAGER | 68319 | 1991-11-11 | 6000.00 | 400.00 | 1001 |
| 66929 | Vijay | CLERK | 68319 | 2002-10-01 | 5000.00 | 600.00 | 1001 |
| 68319 | Rajesh | PRESIDENT | NULL | 1991-11-11 | 70000.00 | NULL | 1001 |
| 66926 | Kamal | MANAGER | 68320 | 2000-01-01 | 8000.00 | 800.00 | 1001 |
| 66927 | Amar | SALESMAN | 68326 | 2000-02-10 | 7000.00 | 600.00 | 1003 |
| 66930 | Ajay | CLERK | 68320 | 2002-10-01 | 5000.00 | 200.00 | 1002 |
+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

GROUP BY clause:

GROUP BY clause is used in association with the SELECT statement to arrange identical data into groups. This GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.

SELECT SUM(column_name) FROM <tablename> WHERE CONDITION GROUP BY column_name;

For example, if you want to know the total salary pay to department then the GROUP BY query would be as follows

```
mysql> select dep_id,sum(salary) from hrms.employees group by dep_id order by dep_id;
+-----+-----+
| dep_id | sum(salary) |
+-----+-----+
| 1001   | 89000.00    |
| 1002   | 5000.00     |
| 1003   | 7000.00     |
+-----+-----+
3 rows in set (0.00 sec)
```

HAVING Clause:

The HAVING Clause enables you to specify conditions that filter which group results appear in the results.

The WHERE clause places conditions on the selected columns, whereas the HAVING clause places conditions on groups created by the GROUP BY clause.

SELECT COUNT(column_name) FROM <tablename> WHERE CONDITION; SQL HAVING Clause

SELECT SUM(column_name) FROM <tablename> WHERE CONDITION GROUP BY column_name HAVING (arithmetic function condition);

Following is an example, which would display total no of employees are working in it

```
mysql> select dep_id, count(dep_id) as 'Total Employee in Department' from hrms.employees
-> group by dep_id having count(dep_id) ;
+-----+-----+
| dep_id | Total Employee in Department |
+-----+-----+
| 1001   | 4                             |
| 1002   | 1                             |
| 1003   | 1                             |
+-----+-----+
3 rows in set (0.00 sec)
```

2.5 CHECK YOUR PROGRESS

- 1) What is SQL?
- 2) What are the usages of SQL?
- 3) Create a department tables table in hrms with following fields
- 4) In department table declare dep_id as primary key

- 5) What is the difference between DELETE and TRUNCATE statement in SQL?
- 6) What are the advantages of SQL?

2.6 CHECK YOUR PROGRESS: POSSIBLE ANSWERS

1) SQL stands for the Structured Query Language. SQL is a open standard query language used for maintaining the relational database and perform many different operations, like data manipulation on tables. It is a database language used for database creation, deletion, fetching rows and modifying rows, etc. sometimes it is pronounced as 'sequel.'

2) Usages of SQL:

SQL performs following operation on database

- execute queries against a database
- retrieve data from a database
- inserts, updates and/or delete records from a database
- create new databases
- create new tables, views etc. in a database
- Perform complex operations on the database.

3) Create a department tables :

```
Create table hrms.department (dep_id int, dep_name varchar(20),  
dep_location varchar(20));
```

4) Alter table hrms.department Add Add constraint Primary Key (dep_id);

5) DELETE is a DML statement while TRUNCATE is DDL statement.

In DELETE we can user WHERE clause while in TRUNCATE we cannot use WHERE clause.

DELETE statement use to delete one or more row from table While in TRUNCATE remove all rows from table.

DELETE is slower than TRUNCATE

You can rollback/undo data after using DELETED statement while rollback is not possible after using TRUNCATE

6) Advantages of SQL:

- A standard for database query languages
- Easy to learn
- Portability
- SQL standard exists
- Both interactive and embedded access
- Can be used by specialist and non-specialist

Unit 3: SQL Sub Languages

3

Unit Structure

- 3.1 Learning Objectives and outcome
- 3.2 Introduction
- 3.3 SQL Sub Languages
- 3.4 MySQL Functions
- 3.5 Check your Progress
- 3.6 Check your Progress: Possible Answers

3.1 LEARNING OBJECTIVES AND OUTCOME

After studying this unit student will be able to:

- How to retrieve and manipulates data from one or more tables through queries and sub-queries
- How to Grant and Revoke privileges on database objects
- Built-in MySQL functions

Outcome:

- Write DCL statement to undo work or permanently save work to database.
- Control Transactions
- Write DML statement to create, modify and delete tables and Retrieve data from the tables through queries and sub-queries
- Work with MySQL built-in functions.

3.2 INTRODUCTION

SQL open standard language to work similarly with most the databases. SQL consist of three built-in sub-languages: Data definition language (DDL), Data manipulation language (DML) and Data control language (DCL). SQL is a fourth generation language. Sometimes we need to generate a result by combining two or more tables. In this condition join and sub-queries help us. DCL commands are used to establish user access to the database. SQL commands directly affect the base tables, which contain the raw data, or they may affect database view, which has been created. TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only. Sometimes Transaction Control Language (TCL) is consideras fourth type of SQL command.

3.3SQL SUB LANGUAGES

3.3.1 DATA CONTROL LANGUAGE(DCL)

DCL commands are used to establish user access to the database. SQL commands directly affect the base tables, which contain the raw data, or they may affect

database view, which has been created. DCL commands are used to grant and take back authority from any database user.

There are two DCL commands:

- Grant
- Revoke

Grant Statement:

```
GRANT SELECT, UPDATE... ON <tablename>|<database>.*  
TO <someuser>, <anotheruser>;
```

For example,

```
Grant Select, Update, Delete ON hrms.employees To hrmanager
```

Revoke Statement:

It is used to take back permissions from the user.

```
REVOKE SELECT, UPDATE,... ON <tablename>|<database>.*  
TO <someuser>, <anotheruser>;
```

For example,

```
Revoke Select, Update, Delete ON hrms.employees To hrmanager
```

3.3.2 TRANSACTION CONTROL LANGUAGE (TCL)

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE, While DCL command's operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

What is Transaction?

A transaction is a unit of work that is performed with a database. Transactions are units or sequences of work accomplished in a logical order, whether in a manual fashion by a user or automatically by application program. For example, if you are

creating a record or updating a record or deleting a record from the table, then you are performing a transaction on that table. It is important to control these transactions to ensure the data integrity and to handle database errors.

Transaction Control Language statements/commands (TCL) are:

TCL Statement	Description
COMMIT	to save the changes parentally in database
ROLLBACK	to roll back the changes
SAVEPOINT	creates points within the groups of transactions in which to ROLLBACK.
SET TRANSACTION	Places a name on a transaction
RELEASE SAVEPOINT	The RELEASE SAVEPOINT command is used to remove a SAVEPOINT that you have created.

COMMIT Statement:

The COMMIT command is the transactional command used to save changes made by a transaction to the database. By default in MySQL, data are saved permanently, that means autocommit is always true.

Syntax of commit statement is as follows:

```
mysql> COMMIT;
```

For examples, employees table contains following data

```
mysql> select * from hrms.employees order by emp_name desc;
+-----+-----+-----+-----+-----+-----+-----+-----+
| emp_id | emp_name | job_name | manager_id | hire_date | salary | commission | dep_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 66928 | Vijay | MANAGER | 68319 | 1991-11-11 | 6000.00 | 400.00 | 1001 |
| 66929 | Vijay | CLERK | 68319 | 2002-10-01 | 5000.00 | 600.00 | 1001 |
| 68319 | Rajesh | PRESIDENT | NULL | 1991-11-11 | 70000.00 | NULL | 1001 |
| 66926 | Kamal | MANAGER | 68320 | 2000-01-01 | 8000.00 | 800.00 | 1001 |
| 66927 | Amar | SALESMAN | 68326 | 2000-02-10 | 7000.00 | 600.00 | 1003 |
| 66930 | Ajay | CLERK | 68320 | 2002-10-01 | 5000.00 | 200.00 | 1002 |
+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

Following is an example which would insert records into the table and then COMMIT the changes in the database.

```
Mysql> Insert into employees values (66931,'Ramesh','CLERK',68320,'2003-05-05',3000.00,000.00,1002);
```

```
Mysql> Commit;
```

```
MariaDB [hrms]> insert into employees values(66931,'Ramesh','CLERK',68320,'2003-05-05',3000.00,000.00,1002);
Query OK, 1 row affected (0.02 sec)

MariaDB [hrms]> commit;
Query OK, 0 rows affected (0.00 sec)
```

Thus, the new row would be inserted and the SELECT statement would produce the following result.

```
MariaDB [hrms]> select * from employees;
+-----+-----+-----+-----+-----+-----+-----+-----+
| emp_id | emp_name | job_name | manager_id | hire_date | salary | commission | dep_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 66926 | Kamal | MANAGER | 68320 | 2000-01-01 | 8000.00 | 800.00 | 1001 |
| 66927 | Amar | SALESMAN | 68326 | 2000-02-10 | 7000.00 | 600.00 | 1003 |
| 66928 | Vijay | MANAGER | 68319 | 1991-11-11 | 6000.00 | 400.00 | 1001 |
| 66929 | Vijay | CLERK | 68319 | 2002-10-01 | 5000.00 | 600.00 | 1001 |
| 66930 | Ajay | CLERK | 68320 | 2002-10-01 | 5000.00 | 200.00 | 1002 |
| 68319 | Rajesh | PRESIDENT | NULL | 1991-11-11 | 70000.00 | NULL | 1001 |
| 66931 | Ramesh | CLERK | 68320 | 2003-05-05 | 3000.00 | 0.00 | 1002 |
+-----+-----+-----+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)
```

ROLLBACK Statement:

Syntax of commit statement is as follows:

```
Mysql> ROLLBACK;
```

The ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database. This command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.

Following is an example, which would delete records from the table which have the emp_id = 66931

```
MariaDB [hrms]> delete from employees where emp_id = 66931;
Query OK, 1 row affected (0.03 sec)

MariaDB [hrms]> select * from employees;
+-----+-----+-----+-----+-----+-----+-----+-----+
| emp_id | emp_name | job_name | manager_id | hire_date | salary | commission | dep_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 66926 | Kamal | MANAGER | 68320 | 2000-01-01 | 8000.00 | 800.00 | 1001 |
| 66927 | Amar | SALESMAN | 68326 | 2000-02-10 | 7000.00 | 600.00 | 1003 |
| 66928 | Vijay | MANAGER | 68319 | 1991-11-11 | 6000.00 | 400.00 | 1001 |
| 66929 | Vijay | CLERK | 68319 | 2002-10-01 | 5000.00 | 600.00 | 1001 |
| 66930 | Ajay | CLERK | 68320 | 2002-10-01 | 5000.00 | 200.00 | 1002 |
| 68319 | Rajesh | PRESIDENT | NULL | 1991-11-11 | 70000.00 | NULL | 1001 |
+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

It deletes a record, now we execute ROLLBACK command then it will undo changes in the database.

```
MariaDB [hrms]> select * from employees;
```

emp_id	emp_name	job_name	manager_id	hire_date	salary	commission	dep_id
66926	Kamal	MANAGER	68320	2000-01-01	8000.00	800.00	1001
66927	Amar	SALESMAN	68326	2000-02-10	7000.00	600.00	1003
66928	Vijay	MANAGER	68319	1991-11-11	6000.00	400.00	1001
66929	Vijay	CLERK	68319	2002-10-01	5000.00	600.00	1001
66930	Ajay	CLERK	68320	2002-10-01	5000.00	200.00	1002
68319	Rajesh	PRESIDENT	NULL	1991-11-11	70000.00	NULL	1001
66931	Ramesh	CLERK	68320	2003-05-05	3000.00	0.00	1002

```
7 rows in set (0.00 sec)
```

SAVEPOINT statement:

A SAVEPOINT is a marking point in a transaction where you can rollback the transaction to a point without rolling back the entire transaction.

The syntax for a SAVEPOINT statement is as follows:

```
SAVEPOINT SAVEPOINT_NAME;
```

Example: SAVEPOINT A2;

If you want to rollback the transaction then, use following command:

```
Rollback To SAVEPOINT_NAME;
```

Example: ROLLBACK To A2;

RELEASE SAVEPOINT Statement;

The syntax for a RELEASE SAVEPOINT statement is as follows:

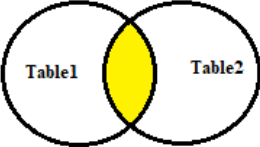
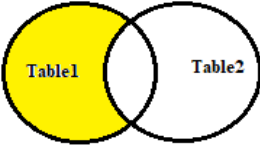
```
RELEASE SAVEPOINT SAVEPOINT_NAME;
```

Example: RELEASE SAVEPOINT A2;

3.3.3 JOIN QUERY

The Joins is used to combine records from two or more tables in a database. A JOIN is a means combine two or more tables is based on common field between them.

There are different types of joins available in SQL

Join Type	Description																																		
Inner Join	<p>Returns records that have matching values in both tables.</p>  <p>Class table</p> <table border="1" data-bbox="435 732 652 1014"> <thead> <tr> <th>ID</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Abhi</td> </tr> <tr> <td>2</td> <td>Rajesh</td> </tr> <tr> <td>3</td> <td>Ramesh</td> </tr> <tr> <td>4</td> <td>Vijay</td> </tr> </tbody> </table> <p>Class_info table</p> <table border="1" data-bbox="435 1131 707 1357"> <thead> <tr> <th>ID</th> <th>Address</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Ahmedabad</td> </tr> <tr> <td>2</td> <td>Baroda</td> </tr> <tr> <td>3</td> <td>Bhavnagar</td> </tr> </tbody> </table> <p>Inner Join</p> <table border="1" data-bbox="435 1471 943 1697"> <thead> <tr> <th>ID</th> <th>Name</th> <th>ID</th> <th>Address</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Abhi</td> <td>1</td> <td>Ahmedabad</td> </tr> <tr> <td>2</td> <td>Rajesh</td> <td>2</td> <td>Baroda</td> </tr> <tr> <td>3</td> <td>Ramesh</td> <td>3</td> <td>Bhavnagar</td> </tr> </tbody> </table>	ID	Name	1	Abhi	2	Rajesh	3	Ramesh	4	Vijay	ID	Address	1	Ahmedabad	2	Baroda	3	Bhavnagar	ID	Name	ID	Address	1	Abhi	1	Ahmedabad	2	Rajesh	2	Baroda	3	Ramesh	3	Bhavnagar
ID	Name																																		
1	Abhi																																		
2	Rajesh																																		
3	Ramesh																																		
4	Vijay																																		
ID	Address																																		
1	Ahmedabad																																		
2	Baroda																																		
3	Bhavnagar																																		
ID	Name	ID	Address																																
1	Abhi	1	Ahmedabad																																
2	Rajesh	2	Baroda																																
3	Ramesh	3	Bhavnagar																																
Left Join	<p>Returns all rows from the left table, even if there are no matches in the right table.</p> 																																		

Class table

ID	Name
1	Abhi
2	Rajesh
3	Ramesh
4	Vijay

Class_info table

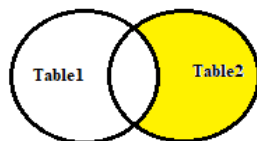
ID	Address
1	Ahmedabad
2	Baroda
3	Bhavnagar

Left Join

ID	Name	ID	Address
1	Abhi	1	Ahmedabad
2	Rajesh	2	Baroda
3	Ramesh	3	Bhavnagar
4	Vijay	Null	Null

Right
Join

Returns all rows from the right table, even if there are no matches in the left table.

**Class table**

ID	Name
1	Abhi
2	Rajesh
3	Ramesh
4	Vijay
5	Anu

Class_info table

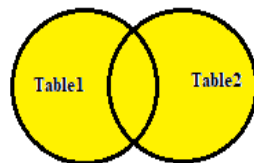
ID	Address
1	Ahmedabad
2	Baroda
3	Bhavnagar
7	Amreli
8	Rajkot

Right Join

ID		ID	Address
1	Abhi	1	Ahmedabad
2	Rajesh	2	Baroda
3	Ramesh	3	Bhavnagar
Null	Null	7	Amreli
Null	Null	8	Rajkot

Full
Join

Returns rows when there is a match in one of the tables. In other word,
Return all records when there is a match in either left or right table

**Class table**

ID	Name
1	Abhi
2	Rajesh
3	Ramesh
4	Vijay
5	Anu

Class_info table	
ID	Address
1	Ahmedabad
2	Baroda
3	Bhavnagar
7	Amreli
8	Rajkot

Full Join			
ID	Name	ID	Address
1	Abhi	1	Ahmedabad
2	Rajesh	2	Baroda
3	Ramesh	3	Bhavnagar
4	Vijay	Null	Null
5	Anu	Null	Null
Null	Null	7	Amreli
Null	Null	8	Rajkot

Self-Join	Used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.
-----------	---

Consider following table and data for further study

```
mysql> select * from employees;
+-----+-----+-----+-----+-----+-----+-----+
| emp_id | emp_name | job_name | manager_id | hire_date | salary | commission | dep_id |
+-----+-----+-----+-----+-----+-----+-----+
| 66926 | Kamal | MANAGER | 68320 | 2000-01-01 | 8000.00 | 800.00 | 1001 |
| 66927 | Amar | SALESMAN | 68326 | 2000-02-10 | 7000.00 | 600.00 | 1003 |
| 66928 | Vijay | MANAGER | 68319 | 1991-11-11 | 6000.00 | 400.00 | 1001 |
| 66929 | Vijay | CLERK | 68319 | 2002-10-01 | 5000.00 | 600.00 | 1001 |
| 66930 | Ajay | CLERK | 68320 | 2002-10-01 | 5000.00 | 200.00 | 1002 |
| 68319 | Rajesh | PRESIDENT | NULL | 1991-11-11 | 70000.00 | NULL | 1001 |
+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> select * from department;
+-----+-----+-----+
| dep_id | dep_name | dep_location |
+-----+-----+-----+
| 1001 | Computer | Ahmedabad |
| 1002 | Electronics | Vadodara |
| 1003 | Electrical | Ahmedabad |
| 1004 | Physics | Vadodara |
| 1005 | Chemistry | Surat |
+-----+-----+-----+
```

INNER JOIN/EQUIJOIN

The most important and frequently used of the joins is the INNER JOIN. It is also known as an EQUIJOIN. The INNER JOIN produce a new result table by combining column values of two tables (table1 and table2) based upon the join-predicate.

```
SELECT table1.column1, table2.column2.... FROM <table1>, <table2> WHERE  
table1.column_field = table2.column_field;
```

The following SQL statement selects all employees and department they belong.

```
Select department.*,employees.* from department, employees where  
department.dep_id = employees.dep_id;
```

```
mysql> Select department.*,employees.* from department, employees where department.dep_id = employees.dep_id;
```

dep_id	dep_name	dep_location	emp_id	emp_name	job_name	manager_id	hire_date	salary	commission	dep_id
1001	Computer	Ahmedabad	66926	Kamal	MANAGER	68320	2000-01-01	8000.00	800.00	1001
1001	Computer	Ahmedabad	66928	Vijay	MANAGER	68319	1991-11-11	6000.00	400.00	1001
1001	Computer	Ahmedabad	66929	Vijay	CLERK	68319	2002-10-01	5000.00	600.00	1001
1001	Computer	Ahmedabad	68319	Rajesh	PRESIDENT	NULL	1991-11-11	70000.00	NULL	1001
1002	Electronics	Vadodara	66930	Ajay	CLERK	68320	2002-10-01	5000.00	200.00	1002
1003	Electrical	Ahmedabad	66927	Amar	SALESMAN	68326	2000-02-10	7000.00	600.00	1003

```
6 rows in set (0.00 sec)
```

Outer Join:

In the SQL outer JOIN all the content of the both tables are integrated together either they are matched or not. There are two types of Outer join - **Left Join and Right Join.**

Left Join: It returns all rows from the left table, even if there are no matches in the right table.

```
SELECT table1.column1, table2.column2.... FROM <table1> LEFT JOIN  
<table2> ON table1.column_field = table2.column_field;
```

```
Select department.*,employees.* from department LEFT JOIN employees ON  
department.dep_id = employees.dep_id;
```

```
mysql> Select department.*,employees.* from department left join employees on department.dep_id = employees.dep_id;
```

dep_id	dep_name	dep_location	emp_id	emp_name	job_name	manager_id	hire_date	salary	commission	dep_id
1001	Computer	Ahmedabad	66926	Kamal	MANAGER	68320	2000-01-01	8000.00	800.00	1001
1003	Electrical	Ahmedabad	66927	Amar	SALESMAN	68326	2000-02-10	7000.00	600.00	1003
1001	Computer	Ahmedabad	66928	Vijay	MANAGER	68319	1991-11-11	6000.00	400.00	1001
1001	Computer	Ahmedabad	66929	Vijay	CLERK	68319	2002-10-01	5000.00	600.00	1001
1002	Electronics	Vadodara	66930	Ajay	CLERK	68320	2002-10-01	5000.00	200.00	1002
1001	Computer	Ahmedabad	68319	Rajesh	PRESIDENT	NULL	1991-11-11	70000.00	NULL	1001
1004	Phyisics	Vadodara	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
1005	Chemistry	Surat	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
8 rows in set (0.00 sec)
```

Right Join: It returns all rows from the right table, even if there are no matches in the left table.

```
SELECT table1.column1, table2.column2.... FROM <table1> RIGHT JOIN <table2> ON table1.column_field = table2.column_field;
```

Select department.*,employees.* from department RIGHT JOIN employees ON department.dep_id = employees.dep_id;

```
mysql> Select department.*,employees.* from department RIGHT join employees on department.dep_id = employees.dep_id;
```

dep_id	dep_name	dep_location	emp_id	emp_name	job_name	manager_id	hire_date	salary	commission	dep_id
1001	Computer	Ahmedabad	66926	Kamal	MANAGER	68320	2000-01-01	8000.00	800.00	1001
1001	Computer	Ahmedabad	66928	Vijay	MANAGER	68319	1991-11-11	6000.00	400.00	1001
1001	Computer	Ahmedabad	66929	Vijay	CLERK	68319	2002-10-01	5000.00	600.00	1001
1001	Computer	Ahmedabad	68319	Rajesh	PRESIDENT	NULL	1991-11-11	70000.00	NULL	1001
1002	Electronics	Vadodara	66930	Ajay	CLERK	68320	2002-10-01	5000.00	200.00	1002
1003	Electrical	Ahmedabad	66927	Amar	SALESMAN	68326	2000-02-10	7000.00	600.00	1003

```
6 rows in set (0.00 sec)
```

Full Join: The SQL full join is the result of combination of both left and right outer join and the join tables have all the records from both tables. It puts NULL on the place of matches not found.

```
SELECT table1.column1, table2.column2.... FROM <table1> LEFT JOIN <table2> ON table1.column_field = table2.column_field;
```

UNION

```
SELECT table1.column1, table2.column2.... FROM <table1> RIGHT JOIN <table2> ON table1.column_field = table2.column_field;
```

Select department.*,employees.* from department LEFT JOIN employees ON department.dep_id = employees.dep_id;

UNION

Select department.*,employees.* from department RIGHT JOIN employees ON department.dep_id = employees.dep_id;

```
mysql> Select department.*,employees.* from department left join employees on department.dep_id = employees.dep_id union Select depart
.*,employees.* from department RIGHT JOIN employees ON department.dep_id = employees.dep_id;
```

dep_id	dep_name	dep_location	emp_id	emp_name	job_name	manager_id	hire_date	salary	commission	dep_id
1001	Computer	Ahmedabad	66926	Kamal	MANAGER	68320	2000-01-01	8000.00	800.00	1001
1003	Electrical	Ahmedabad	66927	Amar	SALESMAN	68326	2000-02-10	7000.00	600.00	1003
1001	Computer	Ahmedabad	66928	Vijay	MANAGER	68319	1991-11-11	6000.00	400.00	1001
1001	Computer	Ahmedabad	66929	Vijay	CLERK	68319	2002-10-01	5000.00	600.00	1001
1002	Electronics	Vadodara	66930	Ajay	CLERK	68320	2002-10-01	5000.00	200.00	1002
1001	Computer	Ahmedabad	68319	Rajesh	PRESIDENT	NULL	1991-11-11	70000.00	NULL	1001
1004	Physiscs	Vadodara	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
1005	Chemistry	Surat	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

8 rows in set (0.00 sec)

3.3.4 SUBQUERY/INNER QUERY/NESTED QUERY:

A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause. A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved. Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

```
SELECT column_name [, column_name ] FROM <tablename1> [,<  
tablename2> ] WHERE column_name OPERATOR (SELECT column_name  
[, column_name ] FROM <tablename1> [, <tablename2 >] [WHERE  
condition])
```

```
mysql> select * from employees where emp_id in (select emp_id from employees where salary > 6000);
```

emp_id	emp_name	job_name	manager_id	hire_date	salary	commission	dep_id
66926	Kamal	MANAGER	68320	2000-01-01	8000.00	800.00	1001
66927	Amar	SALESMAN	68326	2000-02-10	7000.00	600.00	1003
68319	Rajesh	PRESIDENT	NULL	1991-11-11	70000.00	NULL	1001

3 rows in set (0.00 sec)

Subqueries with the INSERT Statement:

Subqueries also can be used with INSERT, UPDATE and DELETE statements.

The syntax is as follows.

```
INSERT INTO <tablename> [ (column1 [, column2 ]) ] SELECT [ *|column1 [, column2 ] FROM <tablename1> [, <tablename2> ] [ WHERE condition]
```

```
UPDATE table SET column_name = new_value [ WHERE OPERATOR [ VALUE ] (SELECT COLUMN_NAME FROM <tablename>) [ WHERE condition) ]
```

```
DELETE FROM TABLE_NAME [ WHERE OPERATOR [ VALUE ] (SELECT COLUMN_NAME FROM <tablename>) [ WHERE condition) ]
```

3.4 MYSQL FUNCTIONS

Here is the list of all important MySQL functions and describe their peruse.

Numeric Functions:

MySQL numeric functions are used primarily for numeric manipulation and/or mathematical calculations. The following table details the numeric functions

Function	Description
ABS	Returns the absolute value of a number
ACOS	Returns the arc cosine of a number
ASIN	Returns the arc sine of a number
ATAN	Returns the arc tangent of one or two numbers
ATAN2	Returns the arc tangent of two numbers
AVG	Returns the average value of an expression
CEIL	Returns the smallest integer value that is >= to a number
CEILING	Returns the smallest integer value that is >= to a number

COS	Returns the cosine of a number
COT	Returns the cotangent of a number
COUNT	Returns the number of records returned by a select query
DEGREES	Converts a value in radians to degrees
DIV	Used for integer division
EXP	Returns e raised to the power of a specified number
FLOOR	Returns the largest integer value that is <= to a number
GREATEST	Returns the greatest value of the list of arguments
LEAST	Returns the smallest value of the list of arguments
LN	Returns the natural logarithm of a number
LOG	Returns the natural logarithm of a number, or the logarithm of a number to a specified base
LOG10	Returns the natural logarithm of a number to base 10
LOG2	Returns the natural logarithm of a number to base 2
MAX	Returns the maximum value in a set of values
MIN	Returns the minimum value in a set of values
MOD	Returns the remainder of a number divided by another number
PI	Returns the value of PI
POW	Returns the value of a number raised to the power of another number

POWER	Returns the value of a number raised to the power of another number
RADIANS	Converts a degree value into radians
RAND	Returns a random number
ROUND	Rounds a number to a specified number of decimal places
SIGN	Returns the sign of a number
SIN	Returns the sine of a number
SQRT	Returns the square root of a number
SUM	Calculates the sum of a set of values
TAN	Returns the tangent of a number
TRUNCATE	Truncates a number to the specified number of decimal places

Examples:

The sqrt() function return square root of a number

```
Mysql> SELECT SQRT(49);
```

```
+-----+
|          SQRT(49)          |
+-----+
|              7              |
+-----+
```


The power() or pow() functions return the value of X raised to the power of Y.

```
mysql> SELECT POWER(3,3);
```

```
+-----+
|          POWER(3,3)          |
+-----+
|                27            |
+-----+
```

Date & Time Functions:

MySQL date & time functions are used primarily for create, play and manipulating date and time value. The following table details the date & time functions

Function	Description
ADDDATE	Adds a time/date interval to a date and then returns the date
ADDTIME	Adds a time interval to a time/datetime and then returns the time/datetime
CURDATE	Returns the current date
CURRENT_DATE	Returns the current date
CURRENT_TIME	Returns the current time
CURRENT_TIMESTAMP	Returns the current date and time
CURTIME	Returns the current time
DATE	Extracts the date part from a datetime expression
DATEDIFF	Returns the number of days between two date values
DATE_ADD	Adds a time/date interval to a date and then returns the date

DATE_FORMAT	Formats a date
DATE_SUB	Subtracts a time/date interval from a date and then returns the date
DAY	Returns the day of the month for a given date
DAYNAME	Returns the weekday name for a given date
DAYOFMONTH	Returns the day of the month for a given date
DAYOFWEEK	Returns the weekday index for a given date
DAYOFYEAR	Returns the day of the year for a given date
EXTRACT	Extracts a part from a given date
FROM_DAYS	Returns a date from a numeric datevalue
HOUR	Returns the hour part for a given date
LAST_DAY	Extracts the last day of the month for a given date
LOCALTIME	Returns the current date and time
LOCALTIMESTAMP	Returns the current date and time
MAKEDATE	Creates and returns a date based on a year and a number of days value
MAKETIME	Creates and returns a time based on an hour, minute, and second value
MICROSECOND	Returns the microsecond part of a time/datetime
MINUTE	Returns the minute part of a time/datetime
MONTH	Returns the month part for a given date
MONTHNAME	Returns the name of the month for a given date

NOW	Returns the current date and time
PERIOD_ADD	Adds a specified number of months to a period
PERIOD_DIFF	Returns the difference between two periods
QUARTER	Returns the quarter of the year for a given date value
SECOND	Returns the seconds part of a time/datettime
SEC_TO_TIME	Returns a time value based on the specified seconds
STR_TO_DATE	Returns a date based on a string and a format
SUBDATE	Subtracts a time/date interval from a date and then returns the date
SUBTIME	Subtracts a time interval from a datetime and then returns the time/datettime
SYSDATE	Returns the current date and time
TIME	Extracts the time part from a given time/datettime
TIME_FORMAT	Formats a time by a specified format
TIME_TO_SEC	Converts a time value into seconds
TIMEDIFF	Returns the difference between two time/datettime expressions
TIMESTAMP	Returns a datetime value based on a date or datetime value
TO_DAYS	Returns the number of days between a date and date "0000-00-00"
WEEK	Returns the week number for a given date

WEEKDAY	Returns the weekday number for a given date
WEEKOFYEAR	Returns the week number for a given date
YEAR	Returns the year part for a given date
YEARWEEK	Returns the year and week number for a given date

Examples of Date and Time functions:

The CURDATE() function Returns the current date as a value in 'YYYY-MM-DD' or YYYYMMDD format, depending on whether the function is used in a string or numeric context.

```
mysql> SELECT CURDATE();
```

```
+-----+
|          CURDATE()          |
+-----+
|          1997-12-15         |
+-----+
```

The CURTIME() function returns the current time as a value in 'HH:MM:SS' or HHMMSS format, depending on whether the function is used in a string or numeric context. The value is expressed in the current time zone.

```
mysql> SELECT CURTIME();
```

```
+-----+
|          CURTIME()          |
+-----+
|          23:50:26           |
+-----+
```

Advanced Functions:

Function	Description
BIN	Returns a binary representation of a number
BINARY	Converts a value to a binary string
CASE	Goes through conditions and return a value when the first condition is met
CAST	Converts a value (of any type) into a specified datatype
COALESCE	Returns the first non-null value in a list
CONNECTION_ID	Returns the unique connection ID for the current connection
CONV	Converts a number from one numeric base system to another
CONVERT	Converts a value into the specified datatype or character set
CURRENT_USER	Returns the user name and host name for the MySQL account that the server used to authenticate the current client
DATABASE	Returns the name of the current database
IF	Returns a value if a condition is TRUE, or another value if a condition is FALSE
IFNULL	Return a specified value if the expression is NULL, otherwise return the expression
ISNULL	Returns 1 or 0 depending on whether an expression is NULL
LAST_INSERT_ID	Returns the AUTO_INCREMENT id of the last row that has been inserted or updated in a table
MD5	Returns digest of the given string
NULLIF	Compares two expressions and returns NULL if they are

	equal. Otherwise, the first expression is returned
SESSION_USER	Returns the current MySQL user name and host name
SYSTEM_USER	Returns the current MySQL user name and host name
USER	Returns the current MySQL user name and host name
VERSION	Returns the current version of the MySQL database

For example:

```
Mysql> Select md5("password");
```

```
+-----+
| md5("password")          |
+-----+
| 5f4dcc3b5aa765d61d8327deb882cf99 |
+-----+
```

Similarly, there many functions are available in MySQL like, string function, aggregate functions etc....

3.4 CHECK YOUR PROGRESS

- 1) What is the difference between a sub-query and a join? Under what circumstances would you not be able to use a sub-query?
- 2) Consider the following Relational database.

Employees (eno, ename, address, basic_salary)

Projects (pno, Pname, enos-of-staff-alotted)

Workin (pno, eno, pjob)

- i) Write a query to find names of employees who are working on all projects

- ii) Write a query to find names of the projects which are currently not being worked upon.
- 3) Write a query to find total number of records stored in employee table

3.5 CHECK YOUR PROGRESS: POSSIBLE ANSWERS

1) With a sub-query, the columns specified in the SELECT list are restricted to only one table. Thus, cannot use a sub-query if the SELECT list contains columns from more than one table. But with a join operation SELECT list contains columns from more than two tables.

2) Related to Relational database:

i) Query to find names of employees who are working on all projects

```
SELECT ename FROM employees WHERE eno IN ( SELECT eno
FROM workin

GROUP BY eno HAVING COUNT (*) = (SELECT COUNT (*) FROM
projects));
```

ii) Query to find names of the projects which are currently not being worked upon.

```
SELECT Pname FROM projects WHERE Pno IN ( SELECT Pno FROM
projects

MINUS GROUP BY eno (SELECT DISTINCT Pno FROM workin));
```

3) Query to find total number of records stored in employee table

```
Select Count(*) from employee;
```

Block-3

PHP Programming

Unit 1: Getting started with PHP

Unit Structure

- 1.7. Learning Objectives
- 1.8. Introduction
- 1.9. Getting with PHP
- 1.10. PHP Comments
- 1.11. Working with variables
- 1.12. Working with constants
- 1.13. Working with simple expression
- 1.14. Working with operators

1.1 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand basics of PHP
- Understand declaration, initialization and implementing variable in PHP
- Understand various operators and expression in PHP

1.2 INTRODUCTION

PHP: Hypertext Preprocessor, it is originally stood for "Personal Home Page"

PHP started out as a open source project. It was founded by Rasmus Lerdorf unleashed the first version of PHP way back in 1994.

- PHP is a server-side scripting language, it is embedded to HTML. It is used to manage content dynamically, databases, session management etc., even you can build entire e-commerce sites.
- Using PHP you can connect and manipulate various databases likes MariaDB/MySQL, PostgreSQL, Oracle, Sybase, Informix, IBM-DB2 and Microsoft SQL Server 2008 V2.
- PHP is enjoyably lively in its execution, especially when compiled as an Apache module on the LINUX side also call LAMP Technology. The MySQL/MariaDB server, once started, executes even very complex queries with huge result sets in record-setting time.
- It's also known as:
 - LAMP: Linux Apache MariaDB/MySQL PHP
 - WAMP: Windows Apache MariaDB/MySQL PHP
 - XAMPP: X-AnyPlatform Apache MariaDB/MySQL PHP Perl
- PHP supports major protocols such as POP3, IMAP, LDAP and PHP6 etc. (Added support for Java and distributed object architectures and making n-tier or MVC development is possible.)
- PHP is forgiving: PHP scripting language tries to be as forgiving as possible.
- PHP Syntax similar like C Programming

- The standard PHP interpreter, powered by the Zend Engine
- PHP is free software released under the PHP License and PHP has been widely ported and can be deployed on most web servers on almost every operating system and platform.(you can download form <http://www.php.net>).

PHP code may be executed with a command line interface (CLI), embedded into HTML code, or it can be used in combination with various web template systems, Web CMS.content management systems, and web frameworks.

PHP code is usually processed by a PHP interpreter implemented as a module in a web server or as a Common Gateway Interface (CGI) executable.

The main characteristics of PHP are:

- PHP is web-specific and open source
- Scripts are embedded into static HTML files
- Fast execution of scripts
- Fast access to the database tier of applications
- Supported by most web servers and operating systems
- Supports many standard network protocols libraries available for IMAP, NNTP, SMTP, POP3
- Supports many database management systems libraries available for UNIX DBM, MySQL, Oracle
- Dynamically generated output like text, HTML, XHTML and any other XML fileAlso Dynamic generate images as a response, PDF files and even Flash animation movies.
- Text processing features, from the POSIX Extended or Perl regular expressions to parsing XML documents
- A fully featured programming language suitable for complex systems development

Three main uses of PHP

Server-side scripting: This is the most traditional and main use for PHP. You need three things to make this work:

- The PHP parser (CGI or server module),
- A web server: needs a connected PHP installation

- A web browser: access PHP page through URL

Command Line Scripting: You can make a PHP script to run without any server or browser. You only need the PHP parser to use as Command line Scripting.

Client-side GUI applications: PHP is probably not the very best language to write windowing(GUI) applications, but PHP-GTK (PHP Graphics Tool Kit) can be used to write such programs.

PHP Usage

PHP is a general-purpose scripting language that is especially suited to server-side web development where PHP generally runs on a web server. Any PHP code in a requested file is executed by the PHP runtime, usually to create dynamic web page content or dynamic images used on Web sites or elsewhere. It can also be used for command-line scripting and client-side graphical user interface (GUI) applications. PHP can be deployed on most Web servers, many operating systems and platforms, and can be used with many relational database management systems (RDBMS). It is available free of charge, and the PHP Group provides the complete source code for users to build, customize and extend for their own use.

PHP acts primarily as a filter, taking input from a file or stream containing text and/or PHP instructions and outputting another stream of data; most commonly the output will be HTML. Since PHP 4, the PHP parser compiles input to produce bytecode for processing by the Zend Engine, giving improved performance over its interpreter predecessor. PHP current version is 7.1.29 and it is released on dated 02-May-2019.

Originally designed to create dynamic Web pages, PHP now focuses mainly on server-side scripting, and it is similar to other server-side scripting languages that provide dynamic content from a Web server to a client, such as Microsoft's ASP.NET, Sun Microsystems' JavaServer Pages, and mod_perl. PHP has also attracted the development of many frameworks that provide building blocks and a design structure to promote rapid application development (RAD). Some of these include CakePHP, Symfony, CodeIgniter, Yii Framework, and Zend Framework, offering features similar to other web application frameworks.

The LAMP architecture has become popular in the Web industry as a way of deploying Web applications. PHP is commonly used as the P in this bundle alongside Linux, Apache and MySQL. Similar packages are also available for

Windows and OS X, then called WAMP and MAMP, with the first letter standing for the respective operating system.

Popular Implementation of PHP

As of April 2007, over 20 million Internet domains had Web services hosted on servers with PHP installed and mod_php was recorded as the most popular Apache HTTP Server module. PHP is used as the server-side programming language on 75% of all Web sites. Web content management systems written in PHP include MediaWiki, Joomla, eZ Publish, SilverStripe, WordPress, Drupal and Moodle. All Web sites created using these tools are written in PHP, including the user-facing portion of Wikipedia, Facebook, and Digg.

Security in PHP

About 30% of all vulnerabilities listed on the National Vulnerability Database are linked to PHP. These vulnerabilities are caused mostly by not following best practice programming rules: technical security flaws of the language itself or of its core libraries are not frequent (23 in 2008, about 1% of the total). Recognizing that programmers make mistakes, some languages include taint checking to detect automatically the lack of input validation which induces many issues. Such a feature is being developed for PHP, but its inclusion in a release has been rejected several times in the past.

There are advanced protection patches such as Suhosin and Hardening-Patch, especially designed for Web hosting environments.

PHPIDS adds security to any PHP application to defend against intrusions. PHPIDS detects attacks based on cross-site scripting (XSS), SQL injection, header injection, directory traversal, remote file execution, remote file inclusion, and denial-of-service (DoS).

1.3 GETTING STATED WITH PHP

PHP development was began in 1994 when Rasmus Lerdorf wrote several Common Gateway Interface (CGI) programs in C.

(As we have already setup PHP execution environment in the Block no 1.)

Characteristics of PHP:

- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

The PHP differentiate PHP scripting code from other elements in the page. The mechanism for doing so is known as 'escaping to PHP.'

Canonical PHP tags: The most universally effective PHP tag style is:

```
<?php..... ?>
```

If you use this style, you can be positive that your tags will always be correctly interpreted as PHP Script code and file extension is <<file_Name>>.php or .php, .php3, php4, php5, .phps, .phpt, .phtml

Now it's time to write your first PHP script. PHP is a server-side scripting language. To create a PHP file, following the below steps for creating or repeat a PHP File:

1. Open Notepad/Notepad++ or any editor
2. Write content in the file as below

```
<?php echo "Welcome BAOU"; ?>
```

3. Save file into <<Apps_Name>>/index.php into www folder of lamp or wamp server (Assume <<Apps_Name>> = BAOU)
4. Let's run our first php code, open any browser
5. Type : <http://localhost/baou/index.php>
6. Output will generated as below

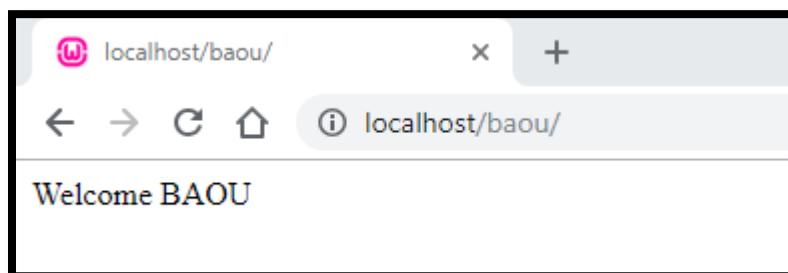


Figure 2 file name: index.php, Hello BAOU output

PHP Syntax

The above PHP script run at only server side, it will response only in client script format. e.g. HTML, Plain text, pdf, etc. and the plain HTML result is sent back to the browser(Client system). The PHP syntax and semantics are the format (syntax) and the related meanings (semantics) of the text and symbols in the PHP programming language. They form a set of rules that define how a PHP program can be written and interpreted. The PHP processor only parses code within its delimiters (the trigger symbols). Anything outside its delimiters is sent directly to the output and not parsed by PHP. There are four ways of including PHP in a web page.

Method 1:

```
<?phpecho("Hello world"); ?>
```

This method is clear and unambiguous.

Method 2:

```
<script language = "php">
    echo("Hello world");
</script>
```

This method is useful in environments supporting mixed scripting languages in the same HTML file (most do not).

Method 3:

```
<? echo("Hello world"); ?>
```

Another short opening tags (<?=>) are also available for use. This method depends on the server configuration.

Method 4:

```
<% echo("Hello world"); %>
```

Another short opening tags (<%=>) are also available for use. This method depends on the server configuration.

We can also use print instead of echo in the methods both functions are nearly identical. The major difference between them is print is slower than echo because the

former will return a status indicating if it was successful or not in addition to text to output, whereas the latter does not return a status and only returns the text for output.

1.4 PHP COMMENTS

A comment is the part of a program that exists only for the developer reader and exclude for displaying/executing the programs result. There are two types of commenting formats in PHP.

Single line comment: it is generally used for short explanations or notes relevant to the local code. below code is examples of single line comments

Example of Single line Comment:

```
<?php
```

```
// This is a First line comment using double slash ()
```

```
# This is a second line comment using hash,
```

```
    echo "Welcome BAOU";
```

```
?>
```

Multi-line comment: It is generally used for multiple line code remarks or notes, below code is examples of multi-line comments

Example of Multi-line Comment:

```
<?php
```

```
/* This is a multi-line comment or remark note, .....
```

```
This is a second line comment or remark note, apply comment using star and slash symbol combination */
```

```
    echo "Welcome BAOU";
```

```
?>
```

1.5 WORKING WITH VARIABLES IN PHP

The main way to store information in the middle of a PHP program is by using a variable.

PHP Variables: A variable can have a short name (like a and b) or a more descriptive name like age, surname, total_volume. PHP is a loosely typed scripting programming language

```
$_1name="BAOU"; // Valid Variable in PHP
```

```
$1_name="BAOU"; // Invalid Variable in PHP – Start with number not allow
```

Basic rules for PHP variables:

- A variable always starts with the \$ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (\$age and \$AGE are two different variables)

Here are the most important things to know about variables in PHP:

- Variables are assigned with the = operator (It is called as assignment operator), with the variable on the left-hand side and the expression to be evaluated on the right.
 - \$name="M.Sc-IT";
 - \$sem=3;
- Variables can, but do not need, to be declared before assignment.
- Variables in PHP do not have inherent types - a variable does not know in advance whether it will be used to store a number or a string of characters.
- Variables used before they are assigned have default values.
- PHP automatically converting data types from one to another when necessary.

- In other languages such as C, C++, and Java, the programmer must declare the name and type of the variable before using it.

Datatype PHP has a total of eight data types which we use to construct declare our variables:

Sr. No	Datatype	Description
1	Integers	It is whole numbers, without a decimal point, like 4195. Example : \$a=10; \$b=2104;
2	Double	It is floating-point numbers, like 3.14159 or 49.1. Example : \$a=5.3; \$b=21.04;
3	Booleans	It has only two possible values either true or false. Example : \$a=true; \$b=false;
4	NULL	It is a special type that only has one value: NULL. Example : \$a=null; \$b=NULL;
5	Strings	It is a sequences of characters, like PHP supports string operations. Example : \$a='BAOU'; \$b="BAOU";
6	Arrays	It is a namedarray() and indexed collections of other values. Example :\$courses = array("M.Sc.-IT", "BCA", "MCA", "PGDCA")
7	Objects	It is instances of user-defined classes, which can pack up both other kinds of values and functions that are specific to the class. Example : \$program = new Course();
8	Resources	It is special variables that hold references to resources external to PHP (such as database connections). Example :\$open_todo_file = fopen("todo_list.txt", "r");

The first five are simple types, and the next two (arrays and objects) are compound type. The compound types can package up other uninformed values of subjective type, whereas the simple types cannot.

Kindly remember that PHP variable names are case-sensitive!

Example: `$age="BAOU", $AGE="BAOU";` `$age` and `$AGE` are two different variables

Note: When you assign a text value to a variable, put quotes around the text

Note: Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

PHP has a useful function named `var_dump()` that prints the current type and value for one or more variables. Arrays and objects are printed recursively with their values indented to show structure.

Integers

They are whole numbers, without a decimal point, like 3011. They are the simple type. They correspond to simple whole numbers, both positive and negative. Integers can be assigned to variables, or they can be used in expressions, like

```
$int_var = 12345;
```

```
$another_int = -12345 + 12345;
```

Integer can be in decimal (base 10), octal (base 8), and hexadecimal (base 16) format. Decimal format is the default, octal integers are specified with a leading 0, and hexadecimals have a leading 0x. (as per mention in Table-1)

Doubles

They like 3.14159 or 49.1. By default, doubles print with the minimum number of decimal places needed. For example, the code:

```
$many = 2.2888800;
```

```
$many_2 = 2.2111200;  
$few = $many + $many_2;  
print(.$many + $many_2 = $few<br>.);
```

It produces the following browser output:

```
2.28888 + 2.21112 = 4.5
```

Boolean

They have only two possible values either true or false. PHP provides a couple of constants especially for use as Booleans: TRUE and FALSE, which can be used like so:

```
if (TRUE)  
    print("This will always print<br>");  
else  
    print("This will never print<br>");
```

Interpreting other types as Booleans

Rules for determine the "truth" of any value not already of the Boolean type:

- If the value is a number, it is false if exactly equal to zero and true otherwise.
- If the value is a string, it is false if the string is empty (has zero characters) or is the string "0", and is true otherwise.
- Values of type NULL are always false.
- If the value is an array, it is false if it contains no other values, and it is true otherwise. For an object, containing a value means having a member variable that has been assigned a value.
- Valid resources are true (although some functions that return resources when they are successful will return FALSE when unsuccessful).
- Don't use double as Booleans.

Each of the following variables has the truth value embedded in its name when it is used in a Boolean context.

```
$true_num = 3 + 0.14159;  
$true_str = "Tried and true"  
$true_array[49] = "An array element";
```

```
$false_array = array();  
$false_null = NULL;  
$false_num = 999 - 999;  
$false_str = "";
```

NULL

NULL is a special type that only has one value "NULL". To assign a NULL value to the variable, simply assign it like this:

```
$my_var = NULL;
```

The special constant NULL is capitalized by convention, but actually it is case insensitive, you could just as well have typed:

```
$my_var = null;
```

A variable that has been assigned NULL has the following properties:

- It evaluates to FALSE in a Boolean context.
- It returns FALSE when tested with Isset() function.

Strings

They are sequences of characters, like "PHP supports string operations". Following are valid examples of string

```
$string_1 = "This is a string in double quotes";  
$string_2 = "This is a somewhat longer, singly quoted string";  
$string_21 = "This string has twenty-one characters";  
$string_0 = ""; //a string with zero characters
```

Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values as well as specially interpreting certain character sequences.

```
<?  
$variable = "BAOU";  
$literally = 'My $variable will not print!\n';  
print($literally);  
$literally = "My $variable will print!\n";  
print($literally);
```

?>

This will produce following result:

My \$variable will not print!\n

My name will print

There are no artificial limits on string length - within the bounds of available memory, you ought to be able to make arbitrarily long strings.

Strings that are delimited by double quotes (as in "this") are preprocessed in both the following two ways by PHP:

- Certain character sequences beginning with backslash (\) are replaced with special characters
- Variable names (starting with \$) are replaced with string representations of their values.

The escape-sequence replacements are:

- \n is replaced by the newline character
- \r is replaced by the carriage-return character
- \t is replaced by the tab character
- \\$ is replaced by the dollar sign itself (\$)
- \" is replaced by a single double-quote (")
- \\ is replaced by a single backslash (\)

String as Document

You can assign multiple lines to a single string variable using here document:

```
<?php
// Save code as multiline.php
$channel =<<<_XML_
<channel>
<title>What is for lunch</title>
<link>http://www.baou.edu.in/ </link><br>
<description>Choose what to eat today in BAOU Canteen.</description>
```

```
</channel>
```

```
_XML_;
```

```
echo "<br>";
```

```
echo <<<END
```

This uses the "here document" syntax to output multiple lines with variable interpolation. Note that the here document terminator must appear on a line with just a semicolon. no extra whitespace!

```
END;
```

```
echo "<br>";
```

```
print $channel;
```

```
?>
```

This will produce following result:

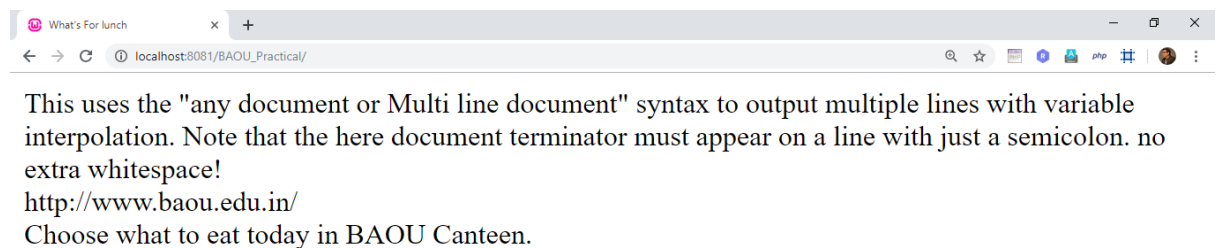


Figure 3 : (multiline.php) Multiline output using generate using XML also page title change

Arrays

PHP has essentially one type of array – the associative array (i.e., hash table). Each element in the array has a key and a corresponding value. Standard arrays (i.e., indexed arrays) can be used in PHP as well; they are simply associative arrays with integer-indexed keys.

There are three ways to populate an array. The first method is to use the array() function:

First Method :

```
$courses = array("M.Sc.-IT", "BCA", "MCA", "PGDCA")
```

The second method is to access the elements directly using the array operator []:

```
$courses[0] = "M.Sc.-IT";  
$courses[1] = "BCA";  
$courses[2] = "MCA";  
$courses[3] = "PGDCA";
```

The third method is to use the array operator with no key provided:

```
for ($i=0; $i< 10; $i++)  
    $nums[] = $i+1;
```

This syntax is used less frequently, and has the effect of appending the given value to the array.

Arrays are heterogeneous, meaning that the data stored in the array does not need to be of the same type. For example, `$mixedbag` contains a string, floating-point value, and a boolean value. To get the number of elements in an array, use the `count()` function.

Associative Arrays

Associative arrays work much like their indexed counterparts, except that associative arrays can have non-numeric keys (e.g., strings). The same three methods for populating indexed arrays apply, except for the `array()` function:

```
$file_ext = array(".c" => "C",  
    ".cpp" => "C++",  
    ".php" => "PHP",  
    ".java" => "Java");
```

The value to the left of the `=>` operator is the key, and right of `=>` operator in the corresponding value of key.

Multidimensional Arrays

Multidimensional arrays in PHP can be indexed or associative, and are heterogeneous. Consider the following code which constructs a multidimensional array.

```
$proglangs['scripted'] = array("Perl", "Python", "PHP");  
$proglangs['compiled'] = array("C", "Java", "FORTRAN");  
$proglangs['fun'] = array("PHP" => "Web programming", "Java" => "Nice for  
object-oriented code.");
```

The variable `$proglangs` is a multidimensional associative array. The first two statements create indexed arrays of strings, and the last statement creates another associative array.

Objects

Object Oriented Programming (OOP) promotes clean modular design, simplifies debugging, maintenance and assists with code reuse.

Class is the unit of object-oriented design. A class is a definition of a structure that contains properties (variables) and methods (functions). Classes are defined with the class keyword.

Example (Object file) : PHP_Object.php

```
<?php
class Person
{
    var $name = "";
    function name ($newname = NULL)
    {
        if (! is_null($newname))
        {
            $this->name = $newname;
        }
        return $this->name;
    }
}
?>
```

Once a class is defined, any number of objects can be made from it with the new keyword, the properties and methods can be accessed with the -> construct:

```
<?php
$ed = new Person;
$ed->name("Riya");
printf("Hello, %s\n", $ed->name);
$tc = new Person;
$tc->name('Happy Boy');
printf("<br>Look out below %s\n", $tc->name); ?>
```



Figure 4 Output of Object variable code

Use the `is_object()` function to test whether a value is an object:

```
if (is_object($x))
{
    // $x is an object
}
```

Resources

Many modules provide several functions for dealing with the outside world. For example, every database extension has at least a function to connect to the database, a function to send a query to the database, and a function to close the connection to the database, because you can have multiple database connections open at once, the connect function gives you something by which to identify that connection when you call the query and close functions: a resource.

Resources are really integers under the surface. Their main benefit is that they are garbage collected when no longer in use. When the last reference to a resource value goes away, the extension that created the resource is called to free any memory, close any connection, etc. for that resource:

```
$res = database_connect(); // fictitious function

database_query($res);

$res = "boo"; // database connection automatically closed
```

The benefit of this automatic cleanup is best seen within functions, when the resource is assigned to a local variable. When the function ends, the variable's value is reclaimed by PHP:

```
function search()
{
    $res = database_connect();
    $database_query($res);
}
```

When there are no more references to the resource, it is automatically shut down.

That said, most extensions provide a specific shutdown or close function, and it's considered good style to call that function explicitly when needed rather than to rely on variable scoping to trigger resource cleanup.

Use the `is_resource()` function to test whether a value is a resource:

```
if (is_resource($x))
{
    // $x is a resource
}
```

Variable Variables

You can reference the value of a variable whose name is stored in another variable.

For example:

```
$foo = 'bar';
$$foo = 'baz';
```

After the second statement executes, the variable \$bar has the value "baz".

Variable References

In PHP, references are how you create variable aliases. To make \$black an alias for the variable

```
$white, use
$black =& $white;
```

The old value of \$black is lost.

Functions can return values by reference (for example, to avoid copying large strings or arrays):

```
function &ret_ref( ) // note the &
{
    $var = "PHP";
    return $var;
}
$v =&ret_ref( ); // note the &
```

PHP Variables Scope:

In PHP script writing, the variables can be declared anywhere in the script.

The scope of a variable is the part of the script where the variable can be referenced or used.

PHP has four different variable scopes:

1. Local variables
2. Global variables
3. Static variables

4. Function parameters

1. Local Variables

A variable declared in a function is considered local; that is, it can be referenced solely in that function. Any assignment outside of that function will be considered to be an entirely different variable from the one contained in the function –

```
<?php
    $a = 5; // global scope
        function myTest()
        {
            echo $a; // local scope
        }
    myTest();
    echo $a; // Inside or outside scope of variable
?>
```

The script above will not produce any output because the echo statement refers to the local scope variable \$a, which has not been assigned a value within this scope.

You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.

Local variables are deleted as soon as the function is completed.

2. Global variables

Global scope refers to any variable that is defined outside of any function.

Global variables can be accessed from any part of the script that is not inside a function.

To access a global variable from within a function, use the global keyword:

```
<?php // Script save as global_var.php
    $a = 15;
    $b = 6;
        function myTest()
        {
            global $a, $b;
            $b = $a + $b;
        }
    myTest();
    echo $b;
?>
```

```
myTest();
echo "Global Variable output is" . $b;
?>
```

The script above will output 21.

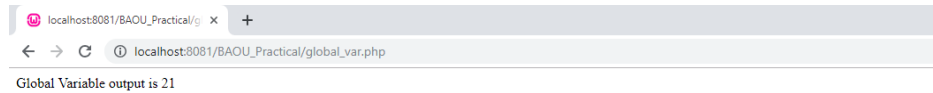


Figure 5 Global Variable output

PHP also stores all global variables in an array called `$GLOBALS[index]`. Its index is the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.

The example above can be rewritten as this:

```
<?php // Script save as global_var_2.php
    $a = 6;
    $b = 15;
    function myTest()
    {
        $GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b'];
    }
    myTest();
    echo "Global Variable output is" . $b;
?>
```

3. Static variables

When a function is completed, all of its variables are normally deleted. However, sometimes if you want to delete a local variable, it cannot be deleting.

To do this, use the static keyword when you first declare the variable.

```
static $rememberMe;
```

Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.

Note: The variable is still local to the function.

4. Function Parameters

Function parameters are declared after the function name and inside parentheses. They are declared much like a typical variable would be, A parameter is a local variable whose value is passed to the function by the calling code.

Parameters are declared in a parameter list as part of the function declaration:

```
function myTest($para1,$para2,...)
{
// function code
}
```

Parameters are also called arguments.

1.6WORKING WITH CONSTANTS IN PHP

A constant is a name or an identifier for a simple value. A constant value cannot change during the execution of the script. By default, a constant is case-sensitive. By convention, constant identifiers are always uppercase. A constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. If you have defined a constant, it can never be changed or undefined.

To define a constant you have to use `define()` function and to retrieve the value of a constant, you have to simply specifying its name. Unlike with variables, you do not need to have a constant with a `$`. You can also use the function `constant()` to read a constant's value if you wish to obtain the constant's name dynamically.

1.7WORKING WITH SIMPLE EXPRESSIONS IN PHP

Regular expressions are powerful pattern matching algorithm that can be performed in a single expression.

Regular expressions use arithmetic operators such as `(+,-,^)` to create complex expressions.

Regular expressions help you accomplish tasks such as validating email addresses, IP address etc.

Why to use regular expressions

- Regular expressions simplify identifying patterns in string data by calling a single function. This saves us coding time.
- When validating user input such as email address, domain names, telephone numbers, IP addresses, etc... Highlighting keywords in search results

When creating a custom HTML template. Regular expressions can be used to identify the template tags and replace them with actual data.

In this unit, you will learn-

- Regular expressions in PHP
- Preg_match
- Preg_split
- Preg_replace
- Meta characters
- Explaining the pattern

Regular expressions in PHP

PHP has built in functions that allow us to work with regular functions. Let's now look at the commonly used regular expression functions in PHP.

- preg_match – this function is used to perform a pattern match on a string. It returns true if a match is found and false if a match is not found.
- preg_split – this function is used to perform a pattern match on a string and then split the results into a numeric array
- preg_replace – this function is used to perform a pattern match on a string and then replace the match with the specified text.

Below is the syntax for a regular expression function such as preg_match, preg_split or preg_replace.

```
<?php
function_name('/pattern/',subject);
?>
```

HERE,

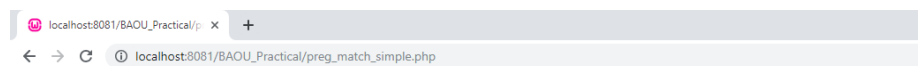
- "function_name(...)" is either preg_match, preg_split or preg_replace.
- "/.../" The forward slashes denote the beginning and end of our regular expression
- "/pattern/" is the pattern that we need to matched
- "subject" is the text string to be matched against

PHP Preg_match

The first example uses the preg_match function to perform a simple pattern match for the word baou in a given URL.

The code below shows the implementation for the above example.

```
<?php // script save as preg_match_simple.php
    $my_url = "www.baou.edu.in";
    if (preg_match("/baou/", $my_url))
    {
        echo "the url $my_url contains baou";
    }
    else
    {
        echo "the url $my_url does not contain baou";
    }
?>
```



the url www.baou.edu.in contains baou

Figure 6 PHP Preg_match script output

- "preg_match(...)" is the PHP regular expression function
- "/baou/" is the regular expression pattern to be matched
- "\$my_url" is the variable containing the text to be matched against.

PHP Preg_split

Let's now look at another example that uses the preg_split function.

We will take a string phrase and explode it into an array; the pattern to be matched is a single space.

The text string to be used in this example is "I Love Regular Expressions".

The code below illustrates the implementation of the above example.

```
<?php // script save as preg_split.php

    $my_text="I am pursuing M.Sc-IT in BAOU, This is Regular Expressions
    example";

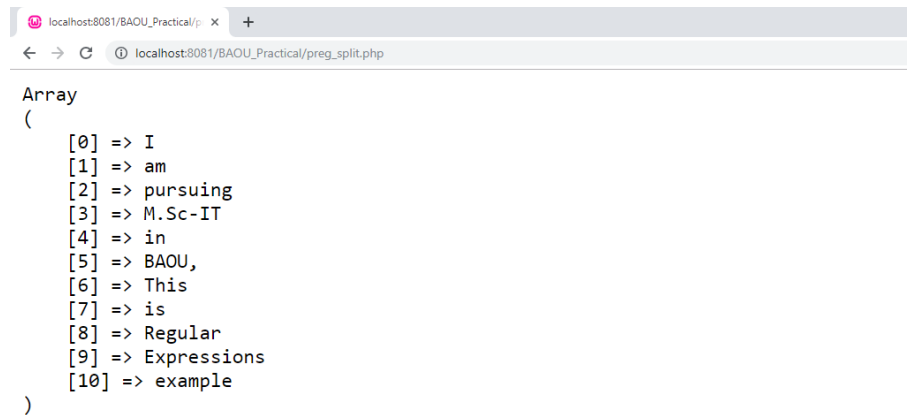
    $my_array =preg_split("/ /", $my_text);

    echo "<pre>";

    print_r($my_array );

    echo "</pre>";

?>
```

```
localhost:8081/BAOU_Practical/p: x +
localhost:8081/BAOU_Practical/preg_split.php
Array
(
    [0] => I
    [1] => am
    [2] => pursuing
    [3] => M.Sc-IT
    [4] => in
    [5] => BAOU,
    [6] => This
    [7] => is
    [8] => Regular
    [9] => Expressions
    [10] => example
)
```

Figure 7Preg_split script output

PHP Preg_replace

Let's now look at the preg_replace function that performs a pattern match and then replaces the pattern with something else.

The code below searches for the word baou in a string.

It replaces the word baou with the word baou surrounded by css code that highlights the background colour.

```
<?php // script save as Preg_replace.php

$text = "You at BAOU for education for all. www.baou.edu.com";

$text= preg_replace("/BAOU/", '<span
style="background:yellow">BAOU</span>', $text);

echo $text;

?>
```



```
localhost:8081/BAOU_Practical/p: x +
localhost:8081/BAOU_Practical/preg_replace.php
You at BAOU for education for all. www.baou.edu.com
```

Figure 8Preg_replace script output

Meta characters

The above examples used very basic patterns; metacharacters simply allow us to perform more complex pattern matches such as test the validity of an email address. Let's now look at the commonly used metacharacters.

Meta_Char	Description	Example
.	Matches any single character except a new line	./ matches anything that has a single character
^	Matches the beginning of or string / excludes characters	^PH/ matches any string that starts with PH
\$	Matches pattern at the end of the string	/com\$/ matches google.com,yahoo.com Etc.
*	Matches any zero (0) or more characters	/com*/ matches computer, communication etc.
+	Requires preceding character(s) appear at least once	/yah+oo/ matches yahoo
\	Used to escape meta characters	/yahoo+\.com/ treats the dot as a literal value
[...]	Character class	/[abc]/ matches abc
a-z	Matches lower case letters	/a-z/ matches cool, happy etc.
A-Z	Matches upper case letters	/A-Z/ matches WHAT, HOW, WHY etc.
0-9	Matches any number between 0 and 9	/0-4/ matches 0,1,2,3,4

The above list only gives the most commonly used metacharacters in regular expressions.

```
<?php // Script save as email_pattern.php
    $my_email = "drashishparejiya@gmail.com";
    if(preg_match("/^[a-zA-Z0-9._-]+@[a-zA-Z0-9]+\.[a-zA-Z]{2,5}$/",$my_email))
    {
        echo "$my_email is a valid email address";
    }
    else
    {
        echo "$my_email is NOT a valid email address";
    }
?>
```

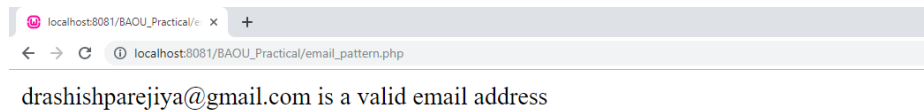


Figure 9 email pattern script output

lets understand the pattern `"[/^[a-zA-Z0-9._-]+@[a-zA-Z0-9-]+\.[a-zA-Z.]{2,5}$/]"` as below

- `"/.../"` starts and ends the regular expression
- `"^[a-zA-Z0-9._-]"` matches any lower or upper case letters, numbers between 0 and 9 and dots, underscores or dashes.
- `"+@[a-zA-Z0-9-]"` matches the @ symbol followed by lower or upper case letters, numbers between 0 and 9 or dashes.
- `"+\.[a-zA-Z.]{2,5}$/"` escapes the dot using the backslash then matches any lower or upper case letters with a character length between 2 and 5 at the end of the string.

As you can see from the above example breakdown, metacharacters are very powerful when it comes to matching patterns.

- A regular expression is a pattern match algorithm
- Regular expressions are very useful when performing validation checks, creating HTML template systems that recognize tags etc.
- PHP has built in functions namely `preg_match`, `preg_split` and `preg_replace` that support regular expressions.
- Metacharacters allow us to create complex patterns

1.8 Working with operators in PHP

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

1. Arithmetic operators
2. Assignment operators
3. Comparison operators
4. Increment/Decrement operators
5. Logical operators
6. String operators

- 7. Array operators
- 8. PHP Type Operator

1. Arithmetic Operators

The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	Name	Description
a+b	Addition	Sum of variables a and b, for example 2+3=5
a-b	Subtraction	Difference of a and b, for example 5-2=3
a*b	Multiplication	Product of a and b, for example 5*2=10
a/b	Division	Quotient of a and b, for example 10/2=5
a%b	Modulus	Remainder of a divided by b, for example 3%2=1
-a	Negation	Opposite of x, for example -5
a.b	Concatenation	Used to concat, for example \$a="Ashish"; \$b="Parejiya"; echo \$a.\$b; // output is AshishParejiya

2. Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable.

The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Operator	Same as	Description
a=b	a=b	The left operand gets set to the value of the expression on the right \$b=10; \$a=\$b; \$a get value 10 from \$b.
a=a+b	a+=b	Addition
a=a-b	a-=b	Subtraction
a=a*b	a*=b	Multiplication
a=a/b	a/=b	Division
a=a%b	a%=b	Modulus
a=a.b	a.=b	Concatenate

3. Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

Operator	Name	Example	Result
==	Equal	<code>\$x == \$y</code>	Returns true if \$x is equal to \$y
===	Identical	<code>\$x === \$y</code>	Returns true if \$x is equal to \$y, and they are of the same type
!=	Not equal	<code>\$x != \$y</code>	Returns true if \$x is not equal to \$y
<>	Not equal	<code>\$x <> \$y</code>	Returns true if \$x is not equal to \$y
!==	Not identical	<code>\$x !== \$y</code>	Returns true if \$x is not equal to \$y, or they are not of the same type
>	Greater than	<code>\$x > \$y</code>	Returns true if \$x is greater than \$y
<	Less than	<code>\$x < \$y</code>	Returns true if \$x is less than \$y
>=	Greater than or equal to	<code>\$x >= \$y</code>	Returns true if \$x is greater than or equal to \$y
<=	Less than or equal to	<code>\$x <= \$y</code>	Returns true if \$x is less than or equal to \$y

4. Increment / Decrement Operators

The PHP increment operators (++) are used to increment a variable's value.

The PHP decrement operators (--) are used to decrement a variable's value.

Operator	Name	Description
<code>++\$x</code>	Pre-increment	Increments \$x by one, then returns \$x
<code>\$x++</code>	Post-increment	Returns \$x, then increments \$x by one
<code>--\$x</code>	Pre-decrement	Decrements \$x by one, then returns \$x
<code>\$x--</code>	Post-decrement	Returns \$x, then decrements \$x by one

5. Logical Operators

The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result
and	And	<code>\$x and \$y</code>	True if both \$x and \$y are true
or	Or	<code>\$x or \$y</code>	True if either \$x or \$y is true
xor	Xor	<code>\$x xor \$y</code>	True if either \$x or \$y is true, but not both
&&	And	<code>\$x && \$y</code>	True if both \$x and \$y are true
	Or	<code>\$x \$y</code>	True if either \$x or \$y is true
!	Not	<code>!\$x</code>	True if \$x is not true

6. String Operators

PHP has two operators that are specially designed for strings.

Operator	Name	Example	Result
.	Concatenation	\$txt1 . \$txt2	Concatenation of \$txt1 and \$txt2
.=	Concatenation assignment	\$txt1 .= \$txt2	Appends \$txt2 to \$txt1

7. Array Operators

The PHP array operators are used to compare arrays.

Operator	Name	Example	Result
+	Union	\$x + \$y	Union of \$x and \$y
==	Equality	\$x == \$y	Returns true if \$x and \$y have the same key/value pairs
===	Identity	\$x === \$y	Returns true if \$x and \$y have the same key/value pairs in the same order and of the same types
!=	Inequality	\$x != \$y	Returns true if \$x is not equal to \$y
<>	Inequality	\$x <> \$y	Returns true if \$x is not equal to \$y
!==	Non-identity	\$x !== \$y	Returns true if \$x is not identical to \$y

8. PHP Type Operator

instanceof is the type operator used to determine if a PHP variable is an instantiated object of a class or not.

```
<?php // script save as PHP_Type_Operator.php
    class MyClass{}

    class OtherClass{}

    $var = new MyClass;

    var_dump($var instanceof MyClass); // prints bool(true)
    var_dump($var instanceof OtherClass); // prints bool(false)

?>
```

Unit 2: Control and Looping Statements

2

Unit Structure

- 2.1. Learning Objectives
- 2.2. Introduction
- 2.3. Control Statement: IF, IF_ELSE
- 2.4. Control Statement: IF...ELSEIF...ELSE, Nested IF
- 2.5. Control Statement: Switch Statement
- 2.6. Looping Statement: For Loop, While, Do ... While, Foreach
- 2.7. PHP break, continue statements
- 2.8. Breaking Out of Nested Loops
- 2.9. Advance Program Flow statement

2.1 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand various control Structure statements
- Understand various Looping statements
- Understand advance program flow statements

2.2 INTRODUCTION

PHP Control, looping statements and Advance ProgramFlowstatement

One of the main reasons for using scripting languages such as PHP is to build logic and intelligence into the creation and deployment of web-based data. In order to be able to build logic into PHP based scripts, it is necessary for the script to be able to make decisions and repeat tasks based on specified criteria.

For example, it may be necessary to repeat a section of a script a specified number of times, or perform a task only if one or more conditions are found to be true (i.e. only let the user log in if a valid password has been provided).

The if, if...else and if...elseif...else construct is one of the most important features of many languages. These conditional statements provide us different actions for the different conditions. When we write code, we perform different actions for different decisions. Like any other languages, PHP is built out of a series of control statements. The control statement can be an assignment, a function call, a loop, a conditional statement or even a statement that does nothing or an empty statement.

In programming terms this is known as flow control and looping. In the simplest terms this involves some standard scripting structures provided by languages such as PHP to control the logic and overall behaviour of a script. Each of these structures provides a simple and intuitive way to build intelligence into scripts. These PHP structures can be broken down into a number of categories as follows:

Conditional Statements

- if statements: it is a control statement, we execute some code only if a specified condition is true.

- if ... else ... statements: We use this control statement to execute some code if a condition is true and another code if the condition is false.
- if...elseif.... else statement – We use this control statement to select one of the several blocks of code to be executed
- switch statement – We use this control statement to select one of many blocks of code to be executed

Looping Statements

- For loops
- while loops
- do ... while loops

In this unit we will explore each of these conditional statements and create some examples of that, demonstrate how to implement these control statements.

2.3 CONTROL STATEMENT: IF, IF_ELSE

What Is a Control Statement?

In simple terms, a control Statement allows you to control the flow of code execution in your application. Generally, a program is executed sequentially, line by line, and a control structure allows you to alter that flow, usually depending on certain conditions.

Control structures are core features of the PHP language that allow your script to respond differently to different inputs or situations. This could allow your script to give different responses based on user input, file contents, or some other data.

The following flowchart explains how a control structure works in PHP.

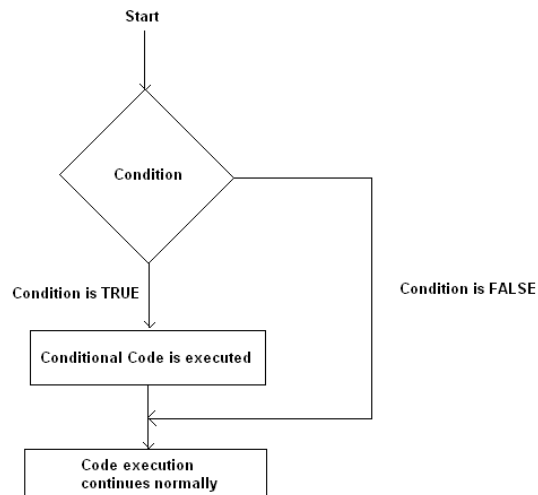


Figure 10 Simple Control Structure with single action

As you can see in the above diagram, first a condition is checked. If the condition is true, the conditional code will be executed. The important thing to note here is that code execution continues normally after conditional code execution.

Let's consider the following example.

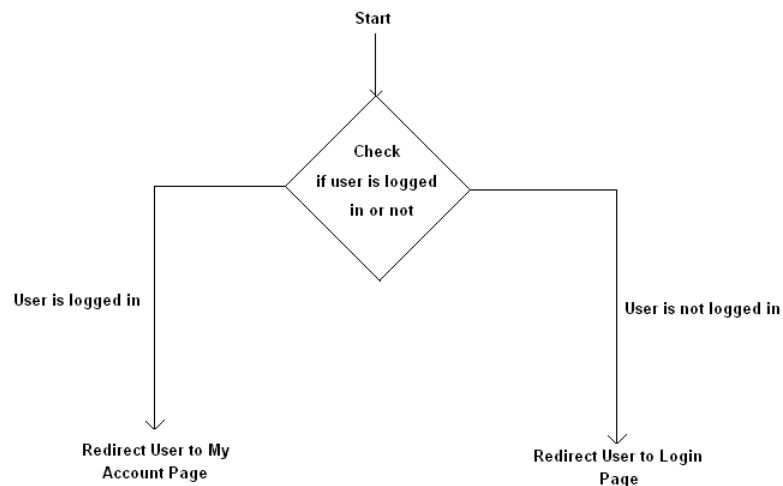


Figure 11 Control Structure with two action redirect

In the above example, the program checks whether or not the user is logged in. Based on the user's login status, they will be redirected to either the Login page or the My Account page. In this case, a control structure ends code execution by redirecting users to a different page. This is a crucial ability of the PHP language.

The if Statement(also called as control statement)

Use the if statement to execute some code only if a specified condition is true.

The expression is evaluated to its Boolean value. If expression evaluates to TRUE, PHP will execute statement, and if it evaluates to FALSE – it'll ignore if part

Syntax of If:

```
if (condition) {  
    code to be executed if condition is true;  
}
```

The following example would display” A is bigger than B” if \$a is bigger than \$b:

```
<?php // Script save as if_condi.php
```

```
    $a=10; $b=5;  
    if ($a > $b)  
    {  
        echo "A is bigger than B";  
    }  
?>
```

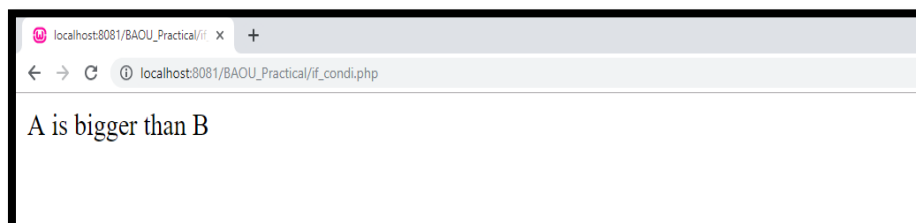


Figure 12 Output of if condition

The if...else Statement:

If...else, as its name suggests, is a combination of if and else. Like else, it extends an if statement to execute a different statement in case the original if mathematical expression evaluates to FALSE. However, unlike else, it will execute that alternative expression only if the elseif conditional expression evaluates to TRUE.

Syntax of If-else:

```
if (condition) {  
    code to be executed if condition is true;  
}
```

```
else {  
    code to be executed if condition is false;  
}
```

For example, the following code would display a is bigger than b or a is smaller than b:

```
<?php // Script save as if_else_codition.php  
    $a=21;  
    $b=12;  
    if ($a > $b) {  
        echo "a is bigger than b";  
    }  
    else {  
        echo " a is smaller than b ";  
    }  
?>
```

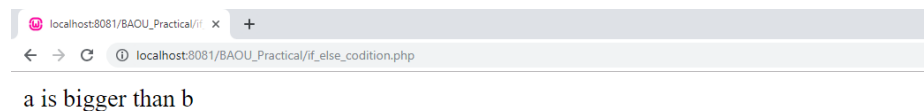


Figure 13 Output of If - else Condition

2.4 IF...ELSEIF...ELSE, NESTED IF

The if...elseif.... else Statement

Use the if.... elseif...else statement to select one of several blocks of code to be executed. We can create multiple branches using the elseif keyword. The elseif keyword tests for another condition if and only if the previous condition was not met. Note that we can use multiple elseifkeywords in our tests.

Syntaxif...elseif....else:

```
if (condition) {  
    code to be executed if this condition is true;  
} elseif (condition) {  
    code to be executed if this condition is true;  
} else {  
    code to be executed if all conditions are false;  
}
```

For example, the following code would display a is bigger than b, a equal to b or a is smaller than b:

```
<?php // Script save as if_elseif_condition.php
    $a=21;
    $b=12;
    if ($a > $b) {
        echo "a is bigger than b";
    } elseif ($a == $b) {
        echo "a is equal to b";
    } else {
        echo "a is smaller than b";
    }
?>
```

Note: Note that elseif and else if will only be considered exactly the same when using curly brackets as in the above example. When using a colon to define your if/elseif conditions, you must not separate else if into two words, or PHP will fail with a parse error.

Nested if statements in PHP

Nested if statements mean an if block inside another if block. Shortly a control structure inside another control structure.

Syntax of Nested If:

```
if (expression 1)
{
    if (expression 2)
    {
        // statements 1
    }
    else
    {
        // Statements 2
    }
}
else
{
    if (expression 2)
    {
```

```

        // Statements 3
    }
    else
    {
        // Statements 4
    }
}

```

Here we can see another if ... else structure inside the if block and else block. Like this we can add any number of nested if else statements.

Nested if statements will make the PHP codes more complex and lengthy. To avoid Nested statements, we can opt for elseif statements

To write the code let's assume that any gender type person is mature or not for wedding as per Indian law, usually laws says a man must be 21 and a woman 18.

Below is the code for this situation-

```

<?php // Script save as if_else_condition_Nested.php
    $age=18;
    $gen="F";
    echo "Age = $age, Gender=$gen <br>";
    if($age>=21)
    {
        $msg="Mature";
    }
    Else
    {
        if($age>=18)
        {
            if($gen=="m"||$gen=="M")
                $msg="Male Not Mature, He's age below <21";
            ($gen=="f"||$gen=="F")
                $msg="Female Mature";
        }
    }
}

```

```

else{
    if($gen=="m"||$gen=="M")
        $msg="Male Not Mature, He's age below <21";
    if($gen=="f"||$gen=="F")
        $msg="Female Mature, She's age below <18";
    }
}
echo "$msg"
?>

```

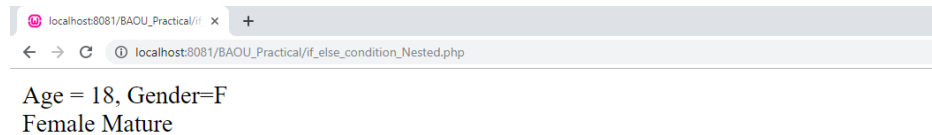


Figure 14if_else_condition_Nested output

2.5 CONTROL STATEMENT: SWITCH STATEMENT

The Switch Statement:

The switch statement is a selection control flow statement. It allows the value of a variable or expression to control the flow of program execution via a multiway branch. It creates multiple branches in a simpler way than using the if, elseif statements.

The switch statement works with two other keywords: case and break. The case keyword is used to test a label against a value from the round brackets. If the label equals to the value, the statement following the case is executed. The break keyword is used to jump out of the switch statement. There is an optional default statement. If none of the labels equals the value, the default statement is executed.

In example script, we have a \$domains_country_extension variable. It has the 'in' string. We use the switch statement to test for the value of the variable. There are several options. If the value equals to 'au' as a string, the 'Australia' string is printed to the console.

```

<?php // script save as PHP_Switch_domain.php
$domain_Name = 'www.baou.edu.in';
$domain = 'in';

```

```

echo "<b>$domain_Name</b> domain extension is <b>$domain</b><br>Switch
Case output is : ";
switch ($domain) {
    case 'in':
        echo "India\n";
        break;
    case 'us':
        echo "United States\n";
        break;
    case 'de':
        echo "Germany\n";
        break;
    case 'au':
        echo "Australia\n";
        break;
    case 'ca':
        echo "Canada\n";
        break;
    default:
        echo "Unknown\n";
        break;
}
?>

```



Figure 15PHP_Switch-Case_domain output

If we changed the *\$domains* variable to 'ca' then we get 'Canada'. If we changed the *\$domains* variable to 'ms', we would get 'Unknown'.

2.6 LOOPING STATEMENT: WHILE, DO ... WHILE, FOR, FOREACH

Loops:

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal code-lines in a script, we can use loops to perform a task like this.

In PHP, we have the following looping statements:

- while: loops through a block of code as long as the specified condition is true
- do...while: loops through a block of code once, and then repeats the loop as long as the specified condition is true
- for: loops through a block of code a specified number of times
- foreach: loops through a block of code for each element in an array

Loop statements are the primary mechanism for telling a computer to perform the same task over and over again until a set of criteria are met. This is where for, while and do ... while loops are of use.

When we write the code, some time we require to write the same code many times. For example, if we have an array with 10 elements and we want to print a message for each element. Then we will have to write the same code 10 times. **Loops** are the replacement of it. By using loops, we don't require the same code again and again. Which is already mentioned above.

The while Loop

The while is a control flow statement that allows code to be executed repeatedly based on a given boolean condition. The while loop executes a block of code as long as the specified condition is true.

Syntax of while loop:

```
while (condition is true) {  
    code to be executed;  
}
```

The while loop executes the statement when the expression is evaluated to true. The statement is a simple statement terminated by a semicolon or a compound statement enclosed in curly brackets. See below example :

```
<?php // Script save as PHP_While_loop.php
    $a = 1;
    While($a <= 4) {
        echo "BAOU: M.Sc. - IT Semester: $a <br>";
        $a++;
    }
?>
```

The example above first sets a variable \$a to 1 (\$a = 1). Then, the while loop will continue to run as long as \$a is less than, or equal to 4 (\$a<= 4). \$x will increase by 1 each time the loop runs (\$a++) output generate as below.

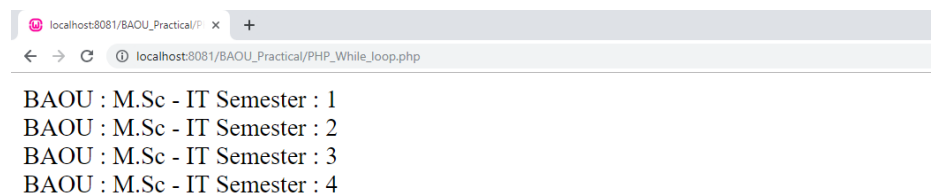


Figure 16PHP_While_loop Output

The do...while Loop

The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true. Do While Loop is very similar to while loop. The difference is, it check the condition at the end of the block and while loop check the condition at the beginning of the block. It means do-while will execute the condition at least once, even the condition is true or false.

Syntax of do...while Loop

```
do {
    code to be executed;
} while (condition is true);
```

The do while loop is a version of the while loop. The difference is that this version is guaranteed to run at least once.

See below example-

```

<?php // Script save as PHP_Do_While_Example.php
    $a = 1;
    do {
        echo "The number is: $a <br>";
        $a++;
    } while ($a <= 5);
?>

```

The example above first sets a variable \$a to 1 (\$a = 1). Then, the do while loop will write some output, and then increment the variable \$a with 1. Then the condition is checked (is \$a less than, or equal to 5?), and the loop will continue to run as long as \$a is less than, or equal to 5:

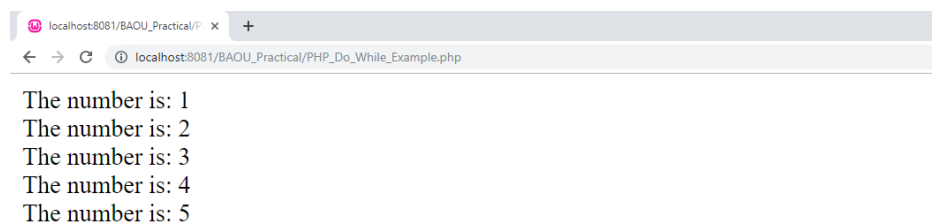


Figure 17 PHP Do ... While loop example output

Note: Notice that in a do while loop the condition is tested AFTER executing the statements within the loop. This means that the do while loop would execute its statements at least once, even if the condition is false the first time.

The for Loop

For loop in PHP is the most complicated loop. We use this loop when we know how many times the code should run.

Syntax of for Loop

```

for (Init counter; Test counter; Increment/Decrement counter) {
    code to be executed;
}

```

Parameters:

- Init counter: Initialize the loop counter value (\$i=1)

- Test counter: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends. ($\$i \leq 5$)
- Increment/Decrement counter: Increases/Decreases the loop counter value ($\$i++$)

The example below displays the numbers from 1 to 5:

```
<?php //Script save as PHP_For_Loop_Example.php
    echo "for loop output as below:<br>";
    for ( $\$i=1$ ;  $\$i \leq 5$ ;  $\$i++$ )
    {
        echo " $\$i$  ";
    }
?>
```

In the above code value of $\$i$ was set to 1 and printed. Each time it was printed, increased by 1 and checked whether it is ≤ 5 .

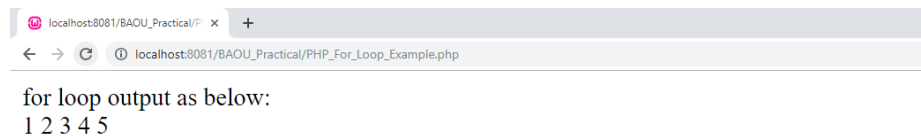


Figure 18 PHP For Loop Example Output

The Foreach Loop

The Foreach loop is used for an array and objects or the foreach loop works only on arrays, and is used to loop through each key/value pair in an array. Also it is used when you don't know the no of iteration in advance of an array at that time foreach loop help for print all element of array without length calculation. (Majority it is used in print select query result set operation)

Syntax of foreach loop is as below:

```
foreach (array_expression as $value)
{
    Statement
}
```

At each iteration the value of current element is assigned to the $\$value$ and internal array pointer is increased by one.

For every loop iteration, the value of the current array element is assigned to $\$value$ and an array pointer is moved by one, until it reaches the last array element.

The following example demonstrates a loop that will output the values of the given array (\$course):

```
<?php // script save as PHP_foreach_example.php
    $course = array("BCA", "M.Sc-IT", "PGDCA", "PGDMAD");
    foreach ($course as $value) {
        echo "BAOU Offer $value Course <br>";
    }
?>
```

The usage of the foreach statement is straightforward. The \$course is the array that we iterate through. The \$value is the temporary variable that has the current value from the array. The foreach statement goes through all the course and prints them to the response output as below.

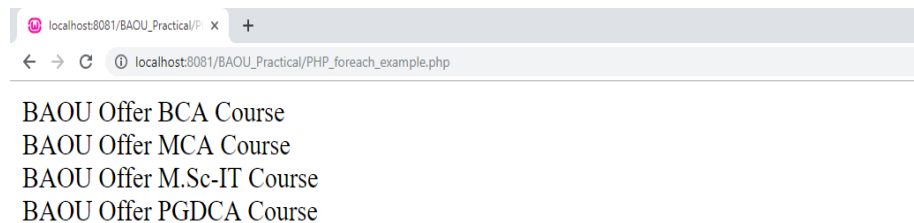


Figure 19 PHP foreach example output

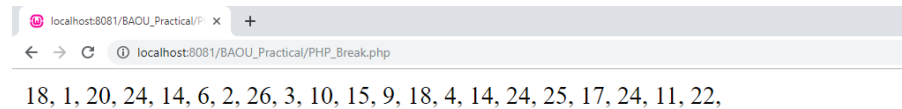
2.7 PHP BREAK, CONTINUE STATEMENTS

break statements

The break statement is used to terminate the loop. The continue statement is used to skip a part of the loop and continue with the next iteration of the loop.

```
<?php // Script save as PHP_Break.php
while (true) {
    $val = rand(1, 30);
    echo $val." ";
    if ($val == 22) {
        break;
    }
}
echo "\n";?>
```

We define an endless while loop. There is only one way to jump out of a such loop—using the break statement. We choose a random value from 1 to 30 and print it. If the value equals to 22, we finish the endless while loop.



```
localhost:8081/BAOU_Practical/PHP_Break.php
18, 1, 20, 24, 14, 6, 2, 26, 3, 10, 15, 9, 18, 4, 14, 24, 25, 17, 24, 11, 22,
```

Figure 20 PHP Break statement example output (No. 22 is end point as per condition)

We might get something like this. (Output may different because of random number function used in code)

PHP continue Statement

Sometimes a situation arises where we want to take the control to the beginning of the loop (for example for, while, do while etc.) skipping the rest statements inside the loop which have not yet been executed.

The keyword continue allow us to do this. When the keyword continue executed inside a loop the control automatically passes to the beginning of loop. Continue is usually associated with the if.

Example:

In the following example, the list of odd numbers between 1 to 10 have printed. In the while loop we test the remainder (here $\$x\%2$) of every number, if the remainder is 0 then it becomes an even number and to avoid printing of even numbers continue statement is immediately used and the control passes to the beginning of the loop.

```
<?php // Script Save as PHP_Continue_Example.php
    $x=1;
    echo 'List of odd numbers between 1 to 10 <br />';
    while ($x<=10)
    {
        if (($x % 2)==0)
        {
            $x++;
            continue;
        }
        else
        {
            echo $x.'<br />';
        }
    }
}
```


This problem can be resolved using the break statement followed by the number of loops you from which you wish to break out. The syntax for this type of break is:

```
break n;
```

where n represents the number of loop levels from which to break. With this knowledge we can modify our previous example to break out of both loops using a break 2; statement:

```
<?php
    for ($i = 0; $i < 1000; $i++)
    {
        for ($x = 0; $x < 100; $x++)
        {
            if ($x == 10)
            {
                break 2;
            }
        }
    }
?>
```

Skipping Statements in Current Loop Iteration

The break statement, when encountered in a loop breaks skips all remaining statements in the loop body and breaks the loop. The continue statement also skips all remaining statements in the loop for the current iteration, but returns to the top of the loop and allows it to continue running.

2.9 ADVANCE PROGRAM FLOW STATEMENT

Try Catch Example: Exception & Error Handling

What is an Exception?

Error is an unexpected program result that cannot be handled by the program itself. Errors are resolved by fixing the program. An example of an error would be an infinite loop that never stops executing.

Exception is unexpected program result that can be handled by the program itself. Examples of exception include trying to open a file that does not exist.

This exception can be handled by either creating the file or presenting the user with an option of searching for the file.

In this chapter, you will learn-

- Why handle exception?
- PHP Error handling
- Error handling examples
- Difference between Errors and Exception
- Multiple Exceptions
- Testing the code

Why handle exception?

Avoid unexpected results on our pages which can be very annoying or irritating to our end users

Improve the security of our applications by not exposing information which malicious users may use to attack our applications

Php Exceptions are used to change the normal flow of a program if any predictable error occurs.

PHP Error handling

When an error occurs, depending on your configuration settings, PHP displays the error message in the web browser with information relating to the error that occurred.

PHP offers a number of ways to handle errors.

We are going to look at three (3) commonly used methods:

- **Die statements:**The die statement combines the echo and exit function in one. It is very useful when we want to output a message and stop the script execution when an error occurs.
- **Custom error handlers:**These are user defined functions that are called whenever an error occurs.

- **Error reporting:** The error message depending on your PHP error reporting settings. This method is very useful in development environment when you have no idea what caused the error. The information displayed can help you debug your application.

Error handling examples

Let's now look at some simple examples with error handling routines.

Let's suppose that we have developed an application that uses text files to store data. We might want to check for the file's existence before we attempt to read data from it.

The code below implements the above example.

```
<?php // Script save as PHP_simple_exception_error.php
    $denominator = 0;
    echo 2 / $denominator;
?>
```

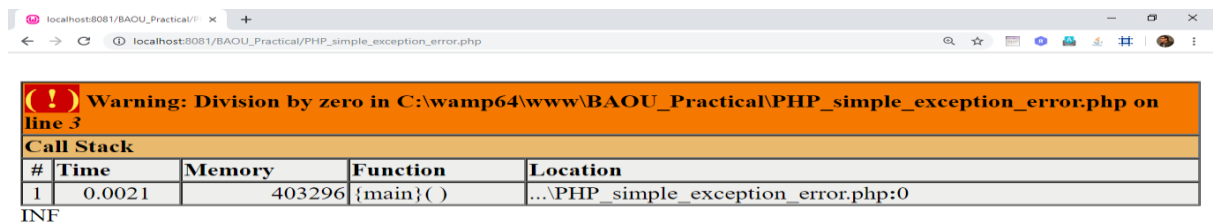


Figure 22 PHP simple exception error output as warning

As you can see from the above results, it makes our application look unprofessional and can be annoying to the user.

let's, we will modify the above code and write an error handler for the application

```
<?php // Script save as PHP_simple_exception_error_handling.php
    $denominator = 0;
    if ($denominator != 0) {
        echo 2 / $denominator;
    } else {
        echo "cannot divide by zero (0)";
    }
?>
```

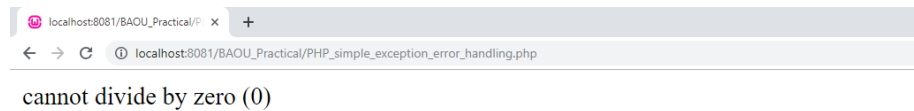


Figure 23 PHP simple exception error handling output

Custom error handling examples

Let's discuss at another example that uses a custom error handler. The custom error handler will be set as the default PHP error handling function and will basically display an error number and message.

The code below illustrates the implementation of the above example

```
<?php // Script save as PHP_exception_custom_error_handler.php
function my_error_handler($error_no, $error_msg)
{
    echo "Opps, something went wrong:";
    echo "Error number: [$error_no]";
    echo "Error Description: [$error_msg]";
}
set_error_handler("my_error_handler");
echo (5 / 0);
?>
```

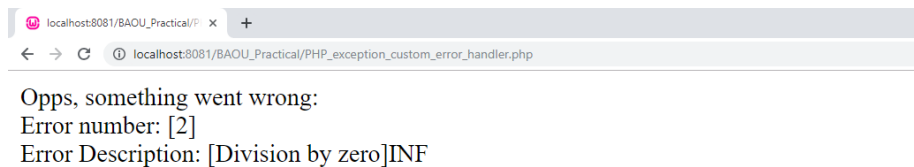


Figure 24 PHP exception custom error handler message output

As you can see from the above example, custom error handlers are powerful in the sense that

- They allow us to customize the error messages.
- The custom error handler can also include error logging in a file/database, emailing the developer etc.

PHP Error reporting

Let's now look at the third type of error handling. We will be using the PHP built in function `error_reporting` function. It has the following basic syntax

```
<?php
    error_reporting($reporting_level);
?>
```

- “error_reporting” is the PHP error reporting function
- “\$reporting_level” is optional, can be used to set the reporting level. If no reporting level has been specified, PHP will use the default error reporting level as specified in the php.ini file.

Reporting Level	Description	Example
E_WARNING	Displays warning messages only. Does not halt the execution of the script	error_reporting(E_WARNING);
E_NOTICE	Displays notices that can occur during normal execution of a program or could be an error.	error_reporting(E_NOTICE);
E_USER_ERROR	Displays user generated errors i.e. custom error handler	error_reporting(E_USER_ERROR);
E_USER_WARNING	Displays user generated warning messages	error_reporting(E_USER_WARNING);
E_USER_NOTICE	Displays user generated notices	error_reporting(E_USER_NOTICE);
E_RECOVERABLE_ERROR	Displays error that are not fatal and can be handled using custom error handlers	error_reporting(E_RECOVERABLE_ERROR);
E_ALL	Displays all errors and warnings	error_reporting(E_ALL);

Difference between Errors and Exception

- Exceptions are thrown and intended to be caught while errors are generally irrecoverable.
- Exceptions are handled in an object oriented way.

This means when an exception is thrown; an exception object is created that contains the exception details.

The table below shows the exception object methods

Method	Description	Example
getMessage()	Displays the exception's message	<pre><?php echo \$e->getMessage(); ?></pre>
getCode()	Displays the numeric code that represents the exception	<pre><?php echo \$e->getCode(); ?></pre>
getFile()	Displays the file name and path where the exception occurred	<pre><?php echo \$e->getFile(); ?></pre>
getLine()	Displays the line number where the exception occurred	<pre><?php echo \$e->getLine(); ?></pre>
getTrace()	Displays an array of the backtrace before the exception	<pre><?php print_r(\$e->getTrace()); ?></pre>
getPrevious()	Displays the previous exception before the current one	<pre><?php echo \$e->getPrevious(); ?></pre>
getTraceAsString()	Displays the backtrace of the exception as a string instead of an array	<pre><?php echo \$e->getTraceAsString(); ?></pre>
__toString()	Displays the entire exception as a string	<pre><?php echo \$e->__toString(); ?></pre>

Below is the basic syntax for throwing an exception.

```
<?php // Script save as PHP_Throw_exception.php
    throw new Exception("This is an exception example");
```

?>

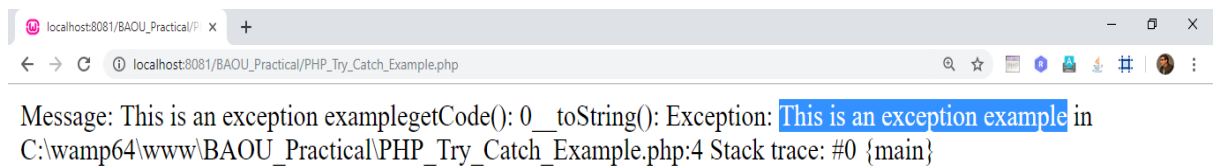


Figure 25 PHP Throw exception output

We are now going to look at an example that implements the throw and catch exceptions.

We will modify the above example and include the try, throw and catch.

Syntax of try – catch:

```
<?php
    try {
        //code goes here that could potentially throw an exception
    }
    catch (Exception $e) {
        //exception handling code goes here
    }
?>
```

- “**try{...}**” is the block of code to be executed that could potentially raise an exception
- “**catch(Exception \$e){...}**” is the block of code that catches the thrown exception and assigns the exception object to the variable \$e.

The code below shows the basic exception example with the try, throw and catch exception implemented.

The program deliberately throws an exception which it then catches.

```

<?php // Script save as PHP_Try_Catch_Example.php

    try {
        $var_msg = "This is an exception example";
        throw new Exception($var_msg);
    }
    catch (Exception $e) {
        echo "Message: " . $e->getMessage();
        echo "\n";
        echo "getCode(): " . $e->getCode();
        echo "\n";
        echo "__toString(): " . $e->__toString();
    }?>

```

It's also possible to create multiple exceptions for one php try statement depending on the type of exception thrown.

Multiple Exceptions

Multiple exception uses multiple try catch blocks to handle the thrown exceptions. Multiple exceptions are useful when,

- You want to display a customized message depending on the exception thrown.
 - You want to perform a unique operation depending on the exception thrown.
- The flowchart below illustrates the how multiple exceptions work

PHP Exception Handle in PHP

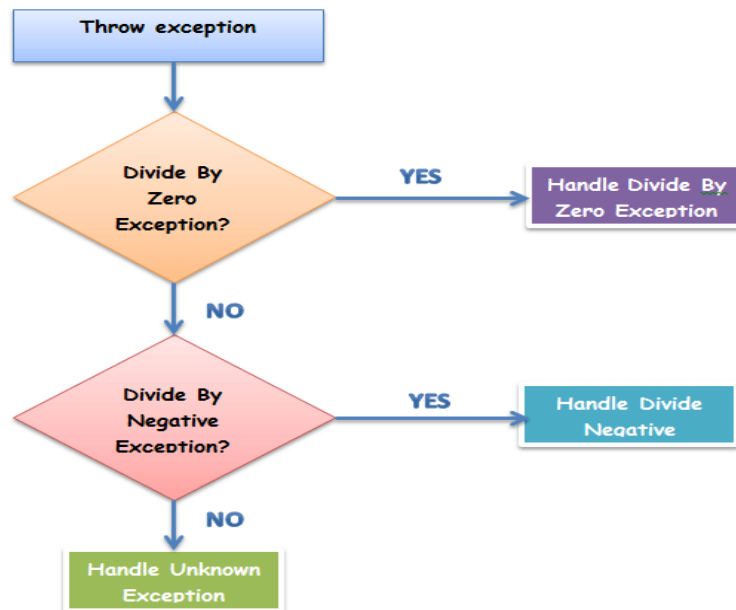


Figure 26 PHP Multiple Exception Handling Flow

Let's discuss at an example that uses multiple exceptions.

We will modify the code that divides a number by the passed in denominator.

We expect two types of exceptions to occur;

- Division by zero
- Division by a negative number

For simplicity's sake, we will only display the exception type in our catch blocks.

The PHP built in Exception class is used to throw exceptions. We will create two classes that extend the exception class and use them to throw exceptions.

The code below shows the implementation of multiple exception.

<?php // Script Save as PHP_Multiple_Exception_Handling.php

```

class DivideByZeroException extends Exception {};
class DivideByNegativeException extends Exception {};
function process($denominator)
{
    try
    {
        if ($denominator == 0)
        {
            throw new DivideByZeroException();
        }
    }
}
  
```



```

    }
    else if ($denominator < 0)
    {
        throw new DivideByNegativeException();
    }
    else
    {
        echo 25 / $denominator;
    }
}
catch (DivideByZeroException $ex)
{
    echo "DIVIDE BY ZERO EXCEPTION!";
}
catch (DivideByNegativeException $ex)
{
    echo "DIVIDE BY NEGATIVE NUMBER EXCEPTION!";
}
catch (Exception $x)
{
    echo "UNKNOWN EXCEPTION!";
}
}
process(0);

```

?>

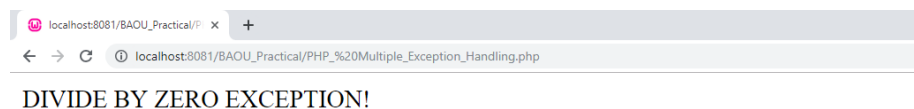


Figure 27 PHP Multiple Exception Handling output

Errors are unexpected results produced by PHP code

- Error handling improves the application performance
- PHP has built in functions that can be used to customize the way PHP reports errors
- Exceptions are like errors, but they can be caught using the catch block when thrown.
- Displaying error messages that show error information is considered a bad security practice.

Unit 3: Working with Functions

3

Unit Structure

- 3.1. Learning Objectives
- 3.2. Introduction
- 3.3. PHP Function
- 3.4. User Defined function
- 3.5. Built in function
- 3.6. Math/Numeric function
- 3.7. String function
- 3.8. Date function
- 3.9. File Inclusion function
- 3.10. File I/O operation function

3.1 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand various user defined and built in functions
- Understand mathematical and string function
- Perform I/O operation on file

3.2 INTRODUCTION

PHP Functions

A function is a block of code written in a program to perform some specific task. We can relate functions in programs to employees in a office in real life for a better understanding of how functions work. Suppose the boss wants his employee to calculate the annual budget. So how will this process complete? The employee will take information about the statics from the boss, performs calculations and calculate the budget and shows the result to his boss. Functions works in a similar manner. They take information as parameter, executes a block of statements or perform operations on these parameters and returns the result.

Types of functions:

- **Built-in functions:** PHP provides us with huge collection of built-in library functions. These functions are already coded and stored in form of functions. To use those we just need to call them as per our requirement like, `var_dump`, `fopen()`, `print_r()`, `gettype()` and so on.
- **User Defined Functions:** Apart from the built-in functions, PHP allows us to create our own customised functions called the user-defined functions.

Using this we can create our own packages of code and use it wherever necessary by simply calling it.

Why should we use functions?

- **Reusability:** If we have a common code that we would like to use at various parts of a program, we can simply contain it within a function and call it whenever required. This reduces the time and effort of repetition of a single

code. This can be done both within a program and also by importing the PHP file, containing the function, in some other program

- **Easier error detection:** Since, our code is divided into functions, we can easily detect in which function, the error could lie and fix them fast and easily.
- **Easily maintained:** As we have used functions in our program, so if anything or any line of code needs to be changed, we can easily change it inside the function and the change will be reflected everywhere, where the function is called. Hence, easy to maintain.

While creating a user defined function we need to keep few things in mind:

1. Any name ending with an open and closed parenthesis is a function.
2. A function name always begins with the keyword function.
3. To call a function we just need to write its name followed by the parenthesis
4. A function name cannot start with a number. It can start with an alphabet or underscore.
5. A function name is not case-sensitive.

In fact, you hardly need to create your own PHP function because there are already more than 1000 of built-in library functions created for different area and you just need to call them according to your requirement.

PHP is very rich in terms of Built-in functions. Here is the list of various important function categories. There are various other function categories which are not covered here. (<https://www.php.net/manual/en/language.functions.php>)

- Array Functions
- Calendar Functions
- Class/Object Functions
- Character Functions
- Date & Time Functions
- Directory Functions
- Error Handling Functions

- File System Functions
- MySQL Functions
- Network Functions
- ODBC Functions
- String Functions
- SimpleXML Functions
- XML Parsing Functions

3.3 PHP FUNCTION

What is a Function?

A function is a reusable piece or block of code that performs a specific action.

Functions can either return values when called or can simply perform an operation without returning any value.

In this Chapter, you will learn:

- Why use Functions?
- Built in Functions
- String Functions
- Numeric Functions
- Date Function
- Why use User Defined Functions?

Why use Functions?

- **Better code organization:** functions allow us to group blocks of related code that perform a specific task together.
- **Reusability:** once defined, a function can be called by a number of scripts in our PHP files. This saves us time of reinventing the wheel when we want to perform some routine tasks such as connecting to the database
- **Easy maintenance:** updates to the system only need to be made in one place.

Built in Functions

Built in functions are functions that exist in PHP installation package.

These built in functions are what make PHP a very efficient and productive scripting language.

The built-in functions can be classified into many categories. Below is the list of the categories.

3.4 USER DEFINED FUNCTION

About user Defined Functions:

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute immediately when a page loads.
- A function will be executed by a call to the function.

Why use User Defined Functions?

General purpose of use define function.

- you have routine tasks in your application such as adding data to the database
- performing validation checks on the data
- Authenticating users in the system etc.

These activities will be spread across a number of pages.

Creating a function that all these pages can be calling is one of the features that make PHP a powerful scripting language. Before we create our first user defined function, let's discuss at the rules that we must follow when creating user defining functions.

Rules for creating user defining function:

- Function names must start with a letter or an underscore but not a number
- The function name must be unique across then entire project development
- The function name must not contain spaces
- A function name can start with a letter or underscore (not a number).
- Functions can optionally accept parametrics and non-parametrics function and return values too.

Let's now discuss How to create a User Defined Function in PHP?

A user-defined function declaration starts with the word **"function"**:

Syntax of User Define Package:

```
functionFunction_Name () {  
    code to be executed;  
}
```

Note: Give the function a name that reflects what the function does!,Function names are NOT case-sensitive.

How to create User Define function:

It's very easy to create your own PHP function. Suppose if you want to create a PHP function which will simply write a simple message on your browser. Following example creates a function called PrintBAOU() and then calls it just after creating it.

Note that while creating a function its name should start with keyword function and all the PHP code should be put inside { Business Logic } braces as shown in the following example below:

Example of User Define function:

```
<?php // Script save as PHP_User_Define_Function_Example.php  
    function PrintBAOU () {  
        echo "Welcome to BAOU, User Define function demo code";  
    }  
    PrintBAOU(); // call the function  
?>
```

In the above example, we create a function named "PrintBAOU()". The opening curly brace ({) indicates the beginning of the function code and the closing curly brace (}) indicates the end of the function. The function outputs "Welcome to BAOU, User Define function demo code ". To call the function, just write its name:



Figure 28 PHP User Define Function Example Output

User Define function type:

- No parameter No return value
- No parameter Get return value
- Pass parameter No return value
- Pass parameter Get return value

PHP Function No parameter, No return value

Function can be display value as per business logical code, let's discuss the example of above type.

Example of NPNR - No parameter No return value as below:

```
<?php // Script save as PHP_User_Define_Function_NPNR.php
    function NPNR() {
        echo "Function Call :- No parameter No return value as below ";
    }
    NPNR(); // call the function
?>
```

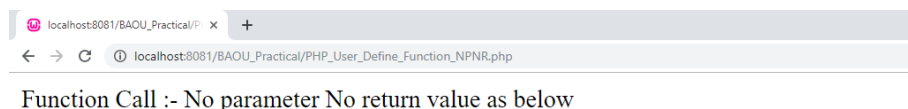


Figure 29 User Define Function NPNR output

PHP Function No parameter Get return value

Function can be return value after execute business logical code using return keyword, let's discuss the example of above user define function type.

Example of NPGR - No parameter Get return value code as below:


```

<?php // Script save as PHP_User_Define_Function_NPGR.php
function NPGR() {
    /* Business Logic Code 1
    .....
    .....
    Business Logic Code 2 */
    return ("Function Call :- No parameter No return value as below");
}
$return_value=NPGR(); // call the function
echo "Display function return value: ".$return_value;
?>

```

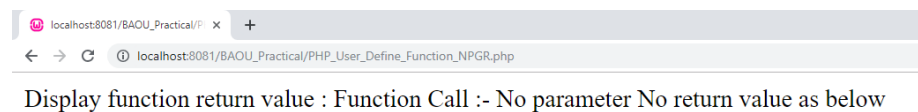


Figure 30 PHP User Define Function NPGR output

PHP Function Arguments/Parameter (PANR - Pass parameter No return value)

Information can be passed to functions through arguments. An argument is just like a variable.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument (\$FirstName). When the DisplayName() function is called, we also pass along a name (e.g. Riya), and the name is used inside the function, which outputs several different first names, but an equal last name:

Example of PPNR - Pass parameter No return value code as below:

```

<?php // Script save as PHP_User_Define_Function_PPNR.php
function DisplayName ($FirstName) {
    echo "Your first name is : $FirstName<br>";
}
DisplayName("Riya");
DisplayName("Lina");
DisplayName("Jiyan");
DisplayName("Jills");

```

?>

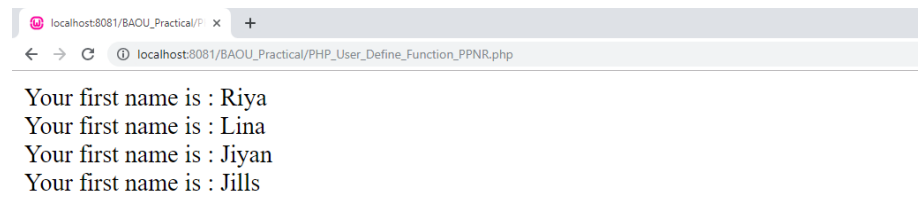


Figure 31 PHP User Define Function PPNR.php

PHP Function PAGR - Pass parameter Get Return value

A function can return a value using the return statement in conjunction with a value or object. return stops the execution of the function and sends the value back to the calling code.

Following example takes two string parameters and concatenate them together and then returns their full name to the calling program. Note that return keyword is used to return a value from a function.

```
<?php // Script save as PHP_User_Define_Function_PPGR.php
function Disp_Full_Name($First_Name, $Last_Name) {
    $Full_Name = $First_Name." ".$Last_Name;
    return $Full_Name;
}
$Get_Ret_Full_Name = Disp_Full_Name("Riya", "Parejiya");
echo "Returned value from the Disp_Full_Name function:
<b>$Get_Ret_Full_Name</b>";
```

?>



Figure 32 PHP User Define Function PPGR.php

Setting Default Values for Function Parameters

You can set a parameter to have a default value if the function's caller doesn't pass it.

Following function prints "print Default Param value as text" in case use does not pass any value to this function.

```
<?php // Script save as PHP_User_Define_Function_Default_Param.php
```

```

function printMe($param = "print Default Param value as text") {
    echo $param."<br>";
}
printMe("Print : Test argument from function as text");
printMe();
?>

```

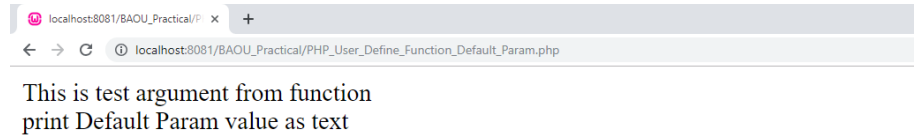


Figure 33 PHP User Define Function Default Parameter output

3.5 BUILT IN FUNCTION

The PHP Internal (Built-in) functions

Built in functions are functions that exist in PHP installation package. PHP comes standard with many functions and constructs. There are also functions that require specific PHP extensions compiled in, otherwise fatal "undefined function" errors will appear.

These built in functions are what make PHP a very efficient and productive scripting language.

The built-in functions can be classified into many categories. Let's we discuss internal (Built-In) function is the next section along with various categories.

3.6 MATH/NUMERIC FUNCTION

Numeric functions are function that return numeric results.

Numeric php function can be used to format numbers, return constants, perform mathematical computations etc.

Let's we understand common PHP numeric/math functions

Function	Description	Example	Output
is_number	Accepts an argument and returns true if its numeric and false if it's not	<pre><?php if(is_numeric("BAOU")) { echo "true"; } else { echo "false"; } ?></pre>	FALSE
		<pre><?php if(is_numeric (123)) { echo "true"; } else { echo "false"; } ?></pre>	TRUE
number_format	Used to format a numeric value using digit separators and decimal points	<pre><?php echo number_format(2509663) ; ?></pre>	2,509,663
rand	Used to generate a random number.	<pre><?php echo rand(); ?></pre>	Random number
round	Round off a number with decimal points to the nearest whole number.	<pre><?php echo rand(); ?></pre>	3
sqrt	Returns the square root of a number	<pre><?php echo sqrt(100); ?></pre>	10
cos	Returns the cosine	<pre><?php echo cos(45);</pre>	0.52532199

sin	Returns the sine	<pre>?> <?php echo sin(45); ?></pre>	0.85090352
tan	Returns the tangent	<pre><?php echo tan(45); ?></pre>	1.61977519
pi	Constant that returns the value of PI	<pre><?php echo pi(); ?></pre>	3.14159265

3.7 STRING FUNCTION

What is a string?

A string is a collection of characters. String is one of the data types supported by PHP.

The string variables can contain alphanumeric characters. Strings are created when;

- You declare variable and assign string characters to it
- You can directly use them with echo statement.
- String are language construct; it helps capture words.
- Learning how strings work in PHP and how to manipulate them will make you a very effective and productive developer.

Let's Define PHP strings variable with value:

Creating Strings Using Single quotes: The simplest way to create a string is to use single quotes.

Let's look at an example that creates a simple string in PHP.

Using Single Quotes:

```
<?php $str='M.Sc-IT'; ?>
```

Using Double Quotes:

```
<?php $str= "M.Sc-IT"; ?>
```

```

<?php
// Script Save as PHP_String_Datatype.php
    var_dump("M.Sc-IT"); //

    var_dump(21);

    var_dump(21.5);

?>

```

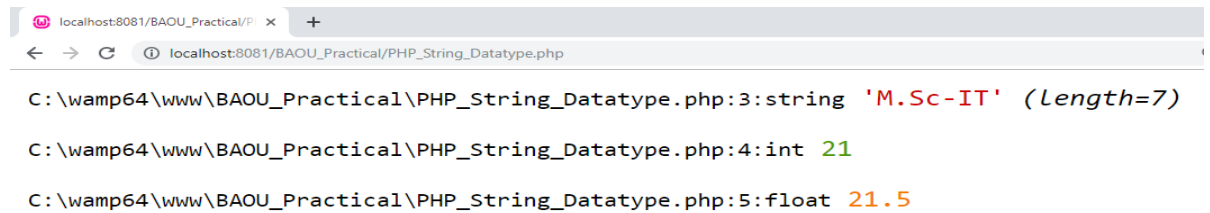


Figure 34 Output of PHP_String_Datatype script

PHP Create Strings Using Double quotes:

The double quotes are used to create relatively complex strings compared to single quotes. Variable names can be used inside double quotes and their values will be displayed. Let's look at an example.

```

<?php

    // Script save as PHP_String_DoubleQuotes.php

    $Uni_Name='BAOU';

    echo "$Uni_Name :Dr. Babasaheb Ambedkar Open University";

?>

```

The above example we create a simple string with the value of “BAOU”. The variable name is then used in the string created using double quotes and its value is interpolated at run time.

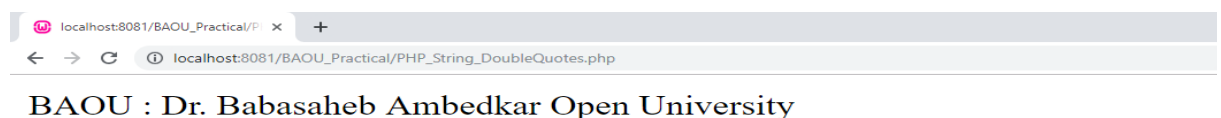


Figure 35 PHP_String_DoubleQuotes output

In addition to variable interpolations, the double quote string can also escape more special characters such as “\n for a linefeed, \\$ dollar for the dollar sign” etc.

PHP Heredoc:

- This heredoc methodology is used to create fairly complex strings as compared to double quotes.
- The heredoc supports all the features of double quotes and allows creating string values with more than one line without php string concatenation.
- Using double quotes to create strings that have multiple lines generates an error.
- You can also use double quotes inside without escaping them.
- The example below illustrates how the Heredoc method is used to create string values.

```
<?php // Script save as PHP_Heredoc.php
    $baby_name = "Riya";
    echo <<<EOT
        My name is $baby_name,
        I like to eat "APPLE" every day.
    EOT;
?>
```

<<<EOT is the string delimiter. EOT is the acronym for end of text. It should be defined in its on line at the beginning of the string and at the end.

Note: you can use anything you like in place of EOT

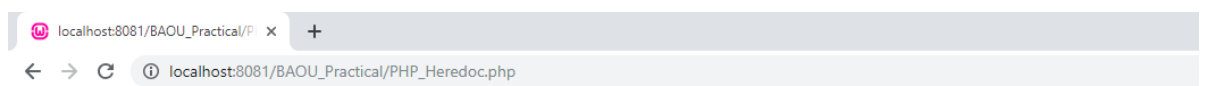


Figure 36 Output of PHP_Heredoc script

PHP Nowdoc

- The Nowdoc string creation method is similar to the heredoc method but works like the way single quotes work.
- No parsing takes place inside the Nowdoc.
- Nowdoc is ideal when working with raw data that do not need to be parsed.
- The code below shows the Nowdoc implementation

It's never printed any variable values as part of code as double quotes,

PHP string functions: all are used to manipulate string values.

We are now going to look at some of the commonly used string functions in PHP

Function	Description	Example	Output
strtolower	Used to convert all string characters to lower case letters	<code>echo strtolower('BaOU');</code>	baou
strtoupper	Used to convert all string characters to upper case letters	<code>echo strtoupper('Ahmedabad');</code>	AHMEDABAD
strlen	The string length function is used to count the number of character in a string. Spaces in between characters are also counted	<code>echo strlen('Riya');</code>	4
explode	Used to convert strings into an array variable	<code>\$settings = explode(';', "host=localhost; db=baou; uid=root; pwd=demo"); print_r(\$settings);</code>	Array ([0] => host=localhost [1] =>db=sales [2] =>uid=root [3] =>pwd=demo)
substr	Used to return part of the string. It accepts three (3) basic parameters. The first one is the string to be shortened, the second parameter is the position of the starting point, and the third parameter is the number of characters to be returned.	<code>\$my_var = 'This is a really long sentence that I wish to cut short'; echo substr(\$my_var,0, 12).!...!;</code>	This is a re...

str_replace	Used to locate and replace specified string values in a given string. The function accepts three arguments. The first argument is the text to be replaced, the second argument is the replacement text and the third argument is the text that is analyzed.	echo str_replace ('the', 'that', 'the mobile is very expensive');	that mobile is very expensive
strpos	Used to locate the and return the position of a character(s) within a string. This function accepts two arguments	echo strpos('PHP Programing','Pro');	4
sha1	Used to calculate the SHA-1 hash of a string value	echo sha1('password');	5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8
md5	Used to calculate the md5 hash of a string value	echo md5('password');	9f961034ee4de758baf4de09ceeb1a75
str_word_count	Used to count the number of words in a string.	echo str_word_count ('This is a really long sentence that I wish to cut short');	12
ucfirst	Make the first character of a string value upper case	echo ucfirst('ashish');	Ashish
lcfirst	Make the first character of a string value lower case	echo lcfirst('UNIVERSITY');	Outputs uNIVERSITY

Function list with Description for self-Study:

Function	Description
addslashes()	Returns a string with backslashes in front of the specified characters
addslashes()	Returns a string with backslashes in front of predefined characters
bin2hex()	Converts a string of ASCII characters to hexadecimal values

<code>chop()</code>	Removes whitespace or other characters from the right end of a string
<code>chr()</code>	Returns a character from a specified ASCII value
<code>chunk_split()</code>	Splits a string into a series of smaller parts
<code>convert_cyr_string()</code>	Converts a string from one Cyrillic character-set to another
<code>convert_uudecode()</code>	Decodes a uuencoded string
<code>convert_uencode()</code>	Encodes a string using the uuencode algorithm
<code>count_chars()</code>	Returns information about characters used in a string
<code>crc32()</code>	Calculates a 32-bit CRC for a string
<code>crypt()</code>	One-way string hashing
<code>echo()</code>	Outputs one or more strings
<code>explode()</code>	Breaks a string into an array
<code>fprintf()</code>	Writes a formatted string to a specified output stream
<code>get_html_translation_table()</code>	Returns the translation table used by <code>htmlspecialchars()</code> and <code>htmlentities()</code>
<code>hebrew()</code>	Converts Hebrew text to visual text
<code>hebrevc()</code>	Converts Hebrew text to visual text and new lines (<code>\n</code>) into <code>
</code>
<code>hex2bin()</code>	Converts a string of hexadecimal values to ASCII characters
<code>html_entity_decode()</code>	Converts HTML entities to characters
<code>htmlentities()</code>	Converts characters to HTML entities
<code>htmlspecialchars_decode()</code>	Converts some predefined HTML entities to characters
<code>htmlspecialchars()</code>	Converts some predefined characters to HTML entities
<code>implode()</code>	Returns a string from the elements of an array
<code>join()</code>	Alias of <code>implode()</code>

lcfirst()	Converts the first character of a string to lowercase
levenshtein()	Returns the Levenshtein distance between two strings
localeconv()	Returns locale numeric and monetary formatting information
ltrim()	Removes whitespace or other characters from the left side of a string
md5()	Calculates the MD5 hash of a string
md5_file()	Calculates the MD5 hash of a file
metaphone()	Calculates the metaphone key of a string
money_format()	Returns a string formatted as a currency string
nl_langinfo()	Returns specific local information
nl2br()	Inserts HTML line breaks in front of each newline in a string
number_format()	Formats a number with grouped thousands
ord()	Returns the ASCII value of the first character of a string
parse_str()	Parses a query string into variables
print()	Outputs one or more strings
printf()	Outputs a formatted string
quoted_printable_decode()	Converts a quoted-printable string to an 8-bit string
quoted_printable_encode()	Converts an 8-bit string to a quoted printable string
quotemeta()	Quotes meta characters
rtrim()	Removes whitespace or other characters from the right side of a string
setlocale()	Sets locale information
sha1()	Calculates the SHA-1 hash of a string
sha1_file()	Calculates the SHA-1 hash of a file
similar_text()	Calculates the similarity between two strings

soundex()	Calculates the soundex key of a string
sprintf()	Writes a formatted string to a variable
sscanf()	Parses input from a string according to a format
str_getcsv()	Parses a CSV string into an array
str_ireplace()	Replaces some characters in a string (case-insensitive)
str_pad()	Pads a string to a new length
str_repeat()	Repeats a string a specified number of times
str_replace()	Replaces some characters in a string (case-sensitive)
str_rot13()	Performs the ROT13 encoding on a string
str_shuffle()	Randomly shuffles all characters in a string
str_split()	Splits a string into an array
str_word_count()	Count the number of words in a string
strcasecmp()	Compares two strings (case-insensitive)
strchr()	Finds the first occurrence of a string inside another string (alias of strstr())
strcmp()	Compares two strings (case-sensitive)
strcoll()	Compares two strings (locale based string comparison)
strcspn()	Returns the number of characters found in a string before any part of some specified characters are found
strip_tags()	Strips HTML and PHP tags from a string
stripslashes()	Unquotes a string quoted with addslashes()
stripslashes()	Unquotes a string quoted with addslashes()
stripos()	Returns the position of the first occurrence of a string inside another string (case-insensitive)
strstr()	Finds the first occurrence of a string inside another string (case-insensitive)

<code>strlen()</code>	Returns the length of a string
<code>strnatcasecmp()</code>	Compares two strings using a "natural order" algorithm (case-insensitive)
<code>strnatcmp()</code>	Compares two strings using a "natural order" algorithm (case-sensitive)
<code>strncasecmp()</code>	String comparison of the first n characters (case-insensitive)
<code>strncmp()</code>	String comparison of the first n characters (case-sensitive)
<code>strpbrk()</code>	Searches a string for any of a set of characters
<code>strpos()</code>	Returns the position of the first occurrence of a string inside another string (case-sensitive)
<code>strrchr()</code>	Finds the last occurrence of a string inside another string
<code>strrev()</code>	Reverses a string
<code>strrpos()</code>	Finds the position of the last occurrence of a string inside another string (case-insensitive)
<code>strrpos()</code>	Finds the position of the last occurrence of a string inside another string (case-sensitive)
<code>strspn()</code>	Returns the number of characters found in a string that contains only characters from a specified charlist
<code>strstr()</code>	Finds the first occurrence of a string inside another string (case-sensitive)
<code>strtok()</code>	Splits a string into smaller strings
<code>strtolower()</code>	Converts a string to lowercase letters
<code>strtoupper()</code>	Converts a string to uppercase letters
<code>strtr()</code>	Translates certain characters in a string
<code>substr()</code>	Returns a part of a string
<code>substr_compare()</code>	Compares two strings from a specified start position (binary safe and optionally case-sensitive)
<code>substr_count()</code>	Counts the number of times a substring occurs in a string

substr_replace()	Replaces a part of a string with another string
trim()	Removes whitespace or other characters from both sides of a string
ucfirst()	Converts the first character of a string to uppercase
ucwords()	Converts the first character of each word in a string to uppercase
fprintf()	Writes a formatted string to a specified output stream
vprintf()	Outputs a formatted string
vsprintf()	Writes a formatted string to a variable
wordwrap()	Wraps a string to a given number of characters

- A string is a set of characters
- single quotes are used to specify simple strings
- double quotes are used to create fairly complex strings
- heredoc is used to create complex strings
- Nowdoc is used to create strings that cannot be parsed.

3.8 DATE FUNCTION

What is PHP Date Function?

PHP date function is an in-built function that simplify working with date data types. The PHP date function is used to format a date or time into a human readable format. It can be used to display the date of article, news, blogs and update was published. record the last updated date and time or timestamp a data in a database.

PHP Date Syntax & Example

```
<?php
    date (format, [timestamp]);
?>
```

Let's understand the syntax and parameters

- “date(.....)” is the function that returns the current time on the server.
- “format” is the general format which we want our output to be i.e.;
- “Y-m-d” for PHP date format YYYY-MM-DD
- “Y” to display the current year
- “[timestamp]” is optional. If no timestamp has been provided, PHP will get the use the php current date time on the server.

Let’s look at a basic example that displays the current year.

```
<?php
    echo date("Y");
?>
```

Output:2018

What is a TimeStamp?

A timestamp is a numeric value in seconds between the current time and value as at 1st January, 1970 00:00:00 Greenwich Mean Time (GMT).

- The value returned by the time function depends on the default time zone.
- The default time zone is set in the php.ini file.
- It can also be set programmatically using date_default_timezone_set function.
- The code below displays the current time stamp

```
<?php // Script save as PHP_TimeStamp.php
    echo time();
?>
```

OUT PUT: 1556108567

Note: the value of the timestamp is not a constant. It changes every second.

Getting a list of available time zone identifiers :

Before we look at how to set the default time zone programmatically, let’s look at how to get a list of supported time zones.

```

<?php // Script save as PHP_TimeZone.php

    $timezone_identifiers = DateTimeZone::listIdentifiers();

    foreach($timezone_identifiers as $key => $list){

        echo $list . "<br>";

    }

?>

```

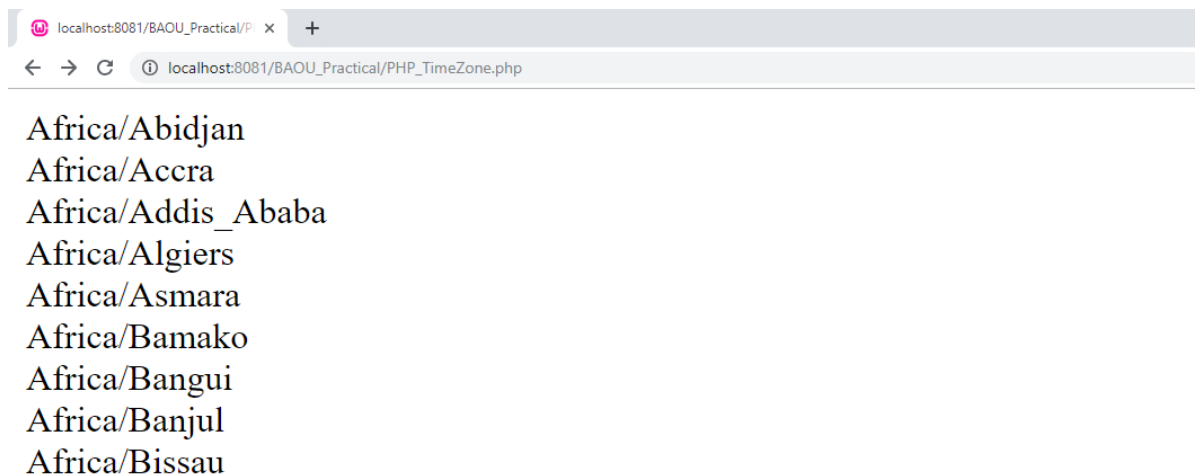


Figure 37 : output of PHP_TimeZone world time zone list, scroll down and find India time zone as (Asia/Kolkata)

- “\$timezone_identifiers = DateTimeZone::listIdentifiers();” calls the listIdentifiers static method of the DateTimeZone built in class.
- The listIdentifiers method returns a list of constants that are assigned to the variable \$timezone_identifiers.
- “foreach{...}” iterates through the numeric array and prints the values.

PHP set Timezone Programmatically

The `date_default_timezone_set` function allows you to set the default time zone from a PHP script.

The set time zone will then be used by all date php function scripts. It has the following syntax.


```
<?php
    date_default_timezone_set( string $timezone_identifier );
?>
```

- “date_default_timezone_set()” is the function that sets the default time zone
- “string \$timezone_identifier” is the time zone identifier

The script below displays the time according to the default time zone set in php.ini. It then changes the default time zone to Asia/Calcutta and displays the time again.

```
<?php// Script save as PHP_SetTimeZone.php
    echo "The time in " . date_default_timezone_get() . " is " . date("H:i:s");

    date_default_timezone_set("Asia/Calcutta");

    echo "The time in " . date_default_timezone_get() . " is " . date("H:i:s");
?>
```

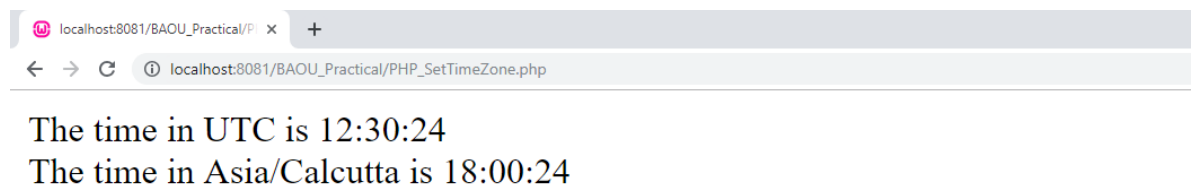


Figure 38 : PHP_SetTimeZone output

PHP Mktime Function:

The mktime function returns the timestamp in a Unix format.

```
<?php
    mktime(hour, minute, second, month, day, year, is_dst);
?>
```

Brief about mktimeparameters :

- “mktime(…)” is the make php timestamp function
- “hour” is optional, it is the number of hour
- “minute” is optional, it is the number of minutes

- “second” is optional, it is the number of seconds
- “month” is optional, it is the number of the month
- “day” is optional, it is the number of the day
- “year” is optional, it is the number of the year
- “is_dst” is optional, it is used to determine the day saving time (DST). 1 is for DST, 0 if it is not and -1 if it is unknown.

PHP Time parameters		
Parameter	Description	Example
“r”	Returns the full date and time	<?php echo date("r"); ?>
“a”, “A”	Returns whether the current time is am or pm, AM or PM respectively	<?php echo date("a"); echo date("A"); ?>
“g”, “G”	Returns the hour without leading zeroes [1 to 12], [0 to 23] respectively	<?php echo date("g"); echo date("G"); ?>
“h”, “H”	Returns the hour with leading zeros [01 to 12],[00 to 23] respectively	<?php echo date("h"); echo date("H"); ?>
“i”, “s”	Returns the minutes/seconds with leading zeroes [00 to 59]	<?php echo date("i"); echo date("s"); ?>
Day parameters		
Parameter	Description	Example
“d”	Returns the day of the month with leading zeroes [01 to 31]	<?php echo date("d"); ?>
“j”	Returns the day of the month without leading zeroes [1 to 31]	<?php echo date("j"); ?>
“D”	Returns the first 3 letters of the day name [Sub to Sat]	<?php echo date("D"); ?>

“l”	Returns day name of the week [Sunday to Saturday]	<?php
		echo date("l");
		?>
“w”	Returns day of the week without leading zeroes [0 to 6] Sunday is represent by zero (0) through to Saturday represented by six (6)	<?php
		echo date("w");
		?>
“z”	Returns the day of the year without leading spaces [0 through to 365]	<?php
		echo date("z");
		?>
Month Parameters		
Parameter	Description	Example
“m”	Returns the month number with leading zeroes [01 to 12]	<?php
		echo date("m");
		?>
“n”	Returns the month number without leading zeroes [01 to 12]	<?php
		echo date("n");
		?>
“M”	Returns the first 3 letters of the month name [Jan to Dec]	<?php
		echo date("M");
		?>
“F”	Returns the month name [January to December]	<?php
		echo date("F");
		?>
“t”	Returns the number of days in a month [28 to 31]	<?php
		echo date("t");
		?>
Year Parameters		
Parameter	Description	Example
“L”	Returns 1 if it's a leap year and 0 if it is not a leap year	<?php
		echo date("L");
		?>
“Y”	Returns four digit year format	<?php
		echo date("Y");
		?>
“y”	Returns two (2) digits year format (00 to 99)	<?php
		echo date("y");
		?>

In the date function we learned,

- The date function is used to format the timestamp into a human desired format.
- The timestamp is the number of seconds between the current time and 1st January, 1970 00:00:00 GMT. It is also known as the UNIX timestamp.
- All date functions use the default time zone set in the php.ini file
- The default time zone can also be set programmatically using PHP scripts.

3.9 FILE INCLUSION FUNCTION

There are two PHP functions which can be used to include one PHP file into another PHP file.

- The `include()` Function
- The `require()` Function

The `include` (or `require`) statement takes all the text/code/markup that exists in the specified file and copies it into the file that uses the include statement.

Including files is very useful when you want to include the same PHP, HTML, or text on multiple pages of a website.

This is a strong point of PHP which helps in creating functions, headers, footers, or elements that can be reused on multiple pages. This will help developers to make it easy to change the layout of complete website with minimal effort. If there is any change required then instead of changing thousands of files just change included file.

It is possible to insert the content of one PHP file into another PHP file (before the server executes it), with the `include` or `require` statement.

The `include` and `require` statements are identical, except upon failure:

- `require` will produce a fatal error (`E_COMPILE_ERROR`) and stop the script
- `include` will only produce a warning (`E_WARNING`) and the script will continue

So, if you want the execution to go on and show users the output, even if the include file is missing, use the include statement. Otherwise, in case of FrameWork, CMS, or a complex PHP application coding, always use the require statement to include a key file to the flow of execution. This will help avoid compromising your application's security and integrity, just in-case one key file is accidentally missing.

Including files saves a lot of work. This means that you can create a standard header, footer, or menu file for all your web pages. Then, when the header needs to be updated, you can only update the header include file.

Syntax:

```
include 'filename';  
  
or  
  
require 'filename';
```

Theinclude() Function:

Theinclude() function takes all the text in a specified file and copies it into the file that uses the include function. If there is any problem in loading a file then the include() function generates a warning but the script will continue execution.

Assume you want to create a common menu for your website. Then create a file BAOU_menu.php with the following content.

```
<!--// save script as BAOU_Menu.php -->  
  
    <a href="index.php">Home</a> -  
  
    <a href="About_us.php">About BAOU</a> -  
  
    <a href="courses.php">Courses</a> -  
  
    <a href="contact.php">Contact us</a><br />
```

Now create as many pages as you like and include this file to create header. For example, now your PHP_Include.php file can have following content.

```
<html><?php // Script save as PHP_include.php ?>
```

```

<body>

<?php include("BAOU_Menu.php"); ?>

<p>This is an example to show how to include PHP file!</p>

</body>

</html>

```

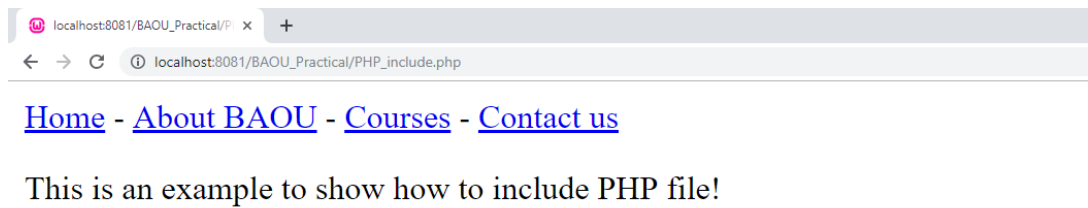


Figure 39PHP_include script output

It will produce the following result –

The require() Function

The require() function takes all the text in a specified file and copies it into the file that uses the include function. If there is any problem in loading a file then the require() function generates a fatal error and halt the execution of the script.

So there is no difference in require() and include() except they handle error conditions. It is recommended to use the require() function instead of include(), because scripts should not continue executing if files are missing or misnamed.

You can try using above example with require() function and it will generate same result. But if you will try following two examples where file does not exist then you will get different results.

Now lets try same example with require() function.

```

<html><?php // Script save as PHP_Require.php ?>

<body>

```

```
<?phprequire("BAOU_Menu.php "); ?>
```

```
<p>This is an example to show how to include wrong PHP file!</p>
```

```
</body>
```

```
</html>
```

Note:

- Use require when the file is required by the application.
- Use include when the file is not required and application should continue when file is not found.

3.10 FILE I/O OPERATION FUNCTION

What is a File?

A file is simply a resource for storing information on a computer. Files are usually used to store information such as Configuration settings of a program, Simple data such as contact names against the phone numbers, Images, Pictures, Photos, etc.

Basic File functions:

- PHP File Formats Support
- PHP files Functions
- PHP File_exists Function
- PHP Fopen Function
- PHP Fwrite Function
- PHP Fclose Function
- PHP Fgets Function
- PHP Copy Function
- Deleting a file
- PHP File_get_contents Function

File Formats Support a wide range of file formats that include;

File.txt, File.log, File.custom_extensioni.e.file.xyz, File.csv, File.gif, file.jpg etc

Files provide a permanent cost effective data storage solution for simple data compared to databases that require other software and skills to manage DBMS systems.

You want to store simple data such as server logs for later retrieval and analysis, store program settings i.e. program.ini

PHP files Functions

PHP provides a convenient way of working with files via its rich collection of built in functions.

Operating systems such as Windows and MAC OS are not case sensitive while Linux or Unix operating systems are case sensitive.

Adopting a naming convention such as lower case letters only for file naming is a good practice that ensures maximum cross platform compatibility.

Let's now look at some of the most commonly used PHP file functions.

PHP File_exists Function

This function is used to determine whether a file exists or not. It comes in handy when we want to know if a file exists or not before processing it. You can also use this function when creating a new file and you want to ensure that the file does not already exist on the server.

The file_exists function has the following syntax.

```
<?php
    file_exists($filename);
?>
```

Explanation:

- “file_exists()” is the PHP function that returns true if the file exists and false if it does not exist.
- “\$file_name” is the path and name of the file to be checked

The code below uses `file_exists` function to determine if the file `my_settings.txt` exists.

```
<?php //Script save as PHP_file_check.php

    if (file_exists('baou_intro.txt'))
    {
        echo 'file found!';
    }
else
    {
        echo 'my_settings.txt does not exist';
    }

?>
```

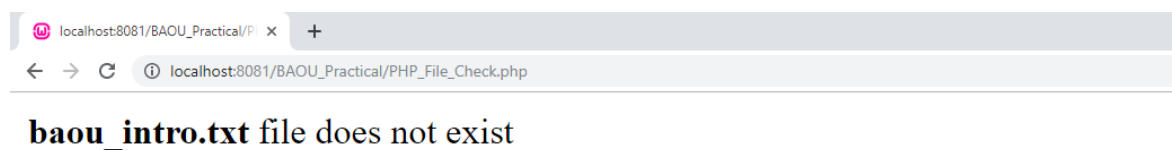


Figure 40 Output of PHP_file_Check.php

PHP Fopen Function

The `fopen` function is used to open files. It has the following syntax

```
<?php
    fopen($file_name,$mode,$use_include_path,$context);
?>
```

Explanation:

- “`fopen`” is the PHP open file function
- “`$file_name`” is the name of the file to be opened
- “`$mode`” is the mode in which the file should be opened, the table below shows the modes

Mode	Description
------	-------------

- r
 - Read file from beginning.
 - Returns false if the file doesn't exist.
 - Read only

- r+
 - Read file from beginning
 - Returns false if the file doesn't exist.
 - Read and write

- w
 - Write to file at beginning
 - truncate file to zero length
 - If the file doesn't exist attempt to create it.
 - Write only

- w+
 - Write to file at beginning, truncate file to zero length
 - If the file doesn't exist attempt to create it.
 - Read and Write

- a
 - Append to file at end
 - If the file doesn't exist attempt to create it.
 - Write only

- a+
 - Php append to file at end
 - If the file doesn't exist attempt to create it
 - Read and write

- “\$use_include_path” is optional, default is false, if set to true, the function searches in the include path too.
- “\$context” is optional, can be used to specify the context support.

PHP Fwrite Function is used to write files.

```
<?php
    fwrite($handle, $string, $length);
?>
```

Explanation:

- “fwrite” is the PHP function for writing to files
- “\$handle” is the file pointer resource

- “\$string” is the data to be written in the file.
- “\$length” is optional, can be used to specify the maximum file length.

PHP FcloseFunction is used to close a file in php which is already open

```
<?php
    fclose($handle);
?>
```

Explanation:

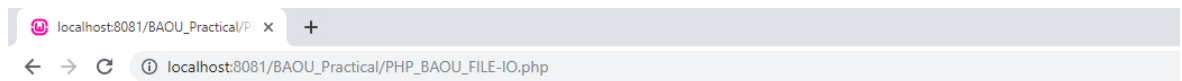
- “fclose” is the PHP function for closing an open file
- “\$handle” is the file pointer resource.
- Let’s now look at an example that creates baou_intro.txt.

We will use the following functions.

Fopen, Fwrite, fclose

The code below “PHP_BAOU_FILE-IO.php” implements the above example.

```
<?php //Script save as PHP_BAOU_FILE-IO.php
    $fh = fopen("baou_intro.txt", 'w') or die("Failed to create file");
    $text = "BAOU File IO Operation Testing Code";
    fwrite($fh, $text) or die("Could not write to file");
    fclose($fh);
    echo "File 'baou_intro.txt' written successfully";
?>
```



File 'baou_intro.txt' written successfully

Figure 41 Output of PHP_BAOU_FILE-IO.php

Note: if your disk is full or you do not have permission to write files, you will get an error message. Kindly refresh the same URL.

PHP Fgets Function is used to read php files line by line. It has the following basic syntax. fgets(\$handle)

Explanation:

- “fgets” is the PHP function for reading file lines
- “\$handle” is the file pointer resource.

Let's now look at an example that reads baou_intro.txt file using the fopen and fgets functions.

```
<?php //Script Save as fileread_baou_intro.php and self tested code
    $fh = fopen("baou_intro.txt", 'r') or die("File does not exist or you lack
    permission to open it");
    $line = fgets($fh);
    echo $line; fclose($fh);
?>
```

- “fopen” function returns the pointer to the file specified in the file path
- “die()” function is called if an error occurs. It displays a message and exists execution of the script

PHP Copy Function

The PHP copy function is used to copy files. It has the following basic syntax.

```
copy($file,$copied_file);
```

- “\$file” specifies the file path and name of the file to be copied.
- “copied_file” specified the path and name of the copied file

The code below illustrates the implementation

```
<?php //Script Save as copyfile_baou_intro.php and self-tested code
    copy('baou_intro.txt', 'baou_intro_clone.txt') or die("Could not copy file");
    echo "File successfully copied to 'baou_intro_clone.txt'";
?>
```

Deleting a file: The unlink function is used to delete the file. The code below illustrates the implementation.

```
<?php // Script save as filedelete_baou_intro_clone.php and self-tested code
    if (!unlink('baou_intro_clone.txt'))
    {
        echo "Could not delete file";
    }
    else
    {
        echo "File 'baou_intro_clone.txt' successfully deleted";
    }
?>
```

PHP File_get_contentsFunction is used to read the entire file contents.

The code below illustrates the implementation.

The difference between `file_get_contents` and `fgets` is that `file_get_contents` returns the file data as a string while `fgets` reads the file line by line.

```
<?php// Script save as file_get_content_baou.php and self-tested code
    echo "<pre>"; // Enables display of line feeds
    echo file_get_contents("baou_intro.txt ");
    echo "</pre>"; // Terminates pre tag
?>
```

Conclusion of File I/O function:

- A file is a resource for storing data
- PHP has a rich collection of built in functions that simplify working with files.
- Common file functions include `fopen`, `fclose`, `file_get_contents`
- The table below shows a summary of the functions covered

Function	Description
File_exists	Used to determine if a file exists or not
fopen	Used to open a file. Returns a pointer to the opened file
fwrite	Used to write to files
fclose	Used to open closed files
fgets	Used to read a file line by line
copy	Used to copy an existing file
unlink	Used to delete an existing file
file_get_contents	Used to return the contents of a file as a string

Unit 4: Working with Arrays

4

Unit Structure

- 4.1. Learning Objectives
- 4.2. Introduction
- 4.3. Storing data in arrays using PHP
- 4.4. Numeric/Indexed Arrays
- 4.5. PHP Associative Array
- 4.6. PHP Multi-dimensional arrays
- 4.7. PHP Array operators
- 4.8. Manipulating arrays.
- 4.9. PHP Array Constants
- 4.10. List of Functions

4.1 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand one dimensional and multi dimensional array
- Understand manipulation of array
- Understand array constant,operator and function

1.2 INTRODUCTION

What is a PHP Array?

An array is a data structure that stores one or more similar type of values in a single value. For example, if you want to store 100 numbers then instead of defining 100 variables it's easy to define an array of 100 length.

A PHP array is a variable that stores more than one piece of related data in a single variable. Think of an array as a box of chocolates with slots inside.

The box represents the array itself while the spaces containing chocolates represent the values stored in the arrays.

4.3 STORING DATA IN ARRAYS USING PHP

PHP storing value in same structural format which is define as below

An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of course names, for example), storing the course in single variables could look like this:

```
$course1 = "BCA";
```

```
$course2 = "MCA";
```

```
$course3 = "M.Sc-IT";
```

Above example is show variable name is common along with numeric index, if we want to store 100+ course name of BAOU then strongly recommended, we should go with Array concept only rather than individual variable define and store value.

```
$course = ["BCA", "MCA", "M.Sc-IT"];
```

This is the simplest way to define array and store value on it.

```
print_r($course);
```

print_r in-built function show array out on screen in structural manner.

4.4 NUMERIC/INDEXED ARRAY

These arrays can store numbers, strings and any object but their index will be represented by numbers. By default, array index starts from zero.

- Numeric arrays use number as access keys.
- An access key is a reference to a memory slot in an array variable.
- The access key is used whenever we want to read or assign a new value an array element.
- Numeric array also known as Indexed array

Below is the syntax for creating numeric array in php. Array Example

```
<?php
    $variable_name[n] = value;
?>

Or

<?php
    $variable_name = array(n => value, ...);
?>
```

Explanation:

- “\$variable_name...” is the name of the variable
- “[n]” is the access index number of the element
- “value” is the value assigned to the array element.

Suppose we have to create 5 (1,2, ...) numeric value base array that we want to store in array variables or 5 numeric as text (One, Two, ...) value.

We can use the example shown below to do as per above assumption. we have used array() function to create array.

```
<?php // Script save as PHP_ARRAY_BASIC.PHP
```

```
    /* First method to create array. */
```

```
    $numbers = array( 1, 2, 3, 4, 5);
```

```
foreach( $numbers as $value ) {
```

```
    echo "Array Value is $value <br />";
```

```
}
```

```
/* Second method to create array. */
```

```
$numbers[0] = "One";
```

```
$numbers[1] = "Two";
```

```
$numbers[2] = "Three";
```

```
$numbers[3] = "Four";
```

```
$numbers[4] = "Five";
```

```
foreach( $numbers as $value ) {
```

```
    echo "Array (As text) Value is $value <br />";
```

```
}
```

```
?>
```

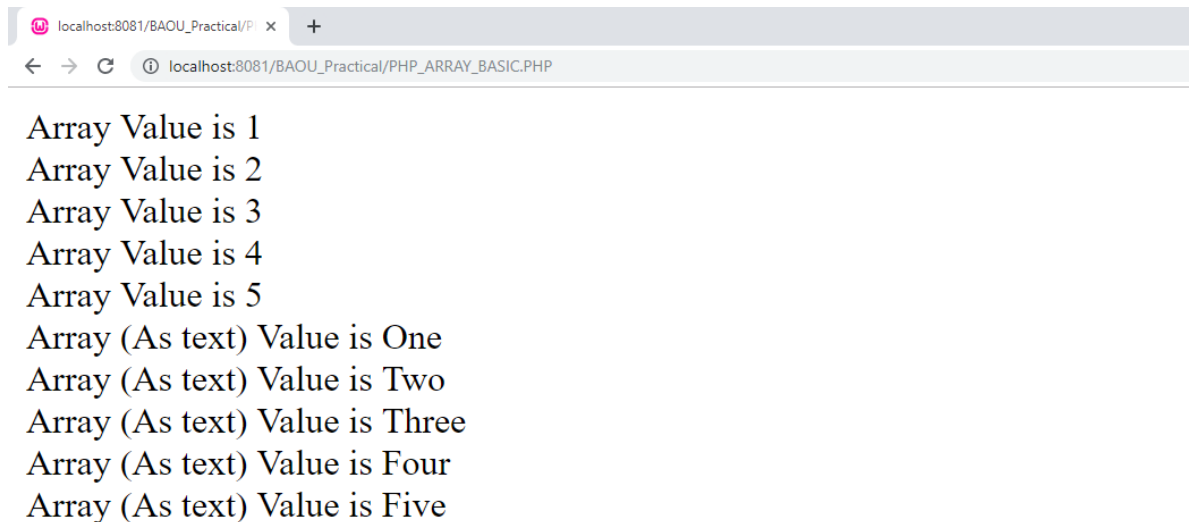


Figure 42 Output of PHP_ARRAY_BASIC.PHP

If we want separate index value form numeric array then need to following method, let's assume array contain is text (One, Two, ...)

<?php//Script save as PHP_ARRAY_INDEX_VALUE.PHP

```
$numbers = array("ZERO","ONE", "TWO","THREE","FOUR","FIVE");  
echo $numbers[2];  
echo "<br>";  
echo $numbers[5];
```

?>

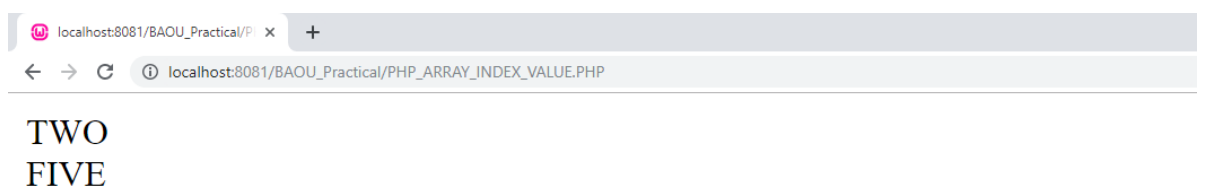


Figure 43 Output of PHP_ARRAY_INDEX_VALUE

This is the simplest way to implement numeric/indexed array in PHP.

4.5 ASSOCIATIVE ARRAY

The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that you can establish a strong association between key and values.

Below is the syntax for creating associative array in php.

```
<?php

    $variable_name['key_name'] = value;

    $variable_name = array('keyname' => value);

?>
```

Explanation:

- “\$variable_name...” is the name of the variable
- “[‘key_name’]” is the access index number of the element
- “value” is the value assigned to the array element.

Let’s suppose that we have a group of persons, and we want to assign the gender of each person against their names.

We can use an associative array to do that. The code below helps us to do that.

```
<?php // Script save as PHP_ASSO_ARRAY.PHP

    $persons = array("Lina" => " Female ", "Ashish" => "Male", "Riya" =>
    "Female");

    echo "<PRE>";

    print_r($persons);

    echo "</PRE>";

    echo "Riya is a " . $persons["Riya"];

?>
```

```
localhost:8081/BAOU_Practical/P x +
localhost:8081/BAOU_Practical/PHP_ASSO_ARRAY.PHP

Array
(
    [Lina] => Female
    [Ashish] => Male
    [Riya] => Female
)

Riya is a Female
```

Figure 44 Output of PHP_ASSO_ARRAY.PHP

Associative array is also very useful when retrieving data from the database.

The field names are used as id keys.

4.6MULTI-DIMENSIONAL ARRAYS

A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index. or alternate way to say multidimensional is that contain other nested arrays.

The advantage of multidimensional arrays is that they allow us to group related data together.

Let's now look at a practical example that implements a php multidimensional array. let's create a two-dimensional array to store marks of three students in three subjects -This example is an associative array, you can create numeric array in the same fashion.

```
<?php // Script save as PHP_MULTI_DIME_ARRAY.PHP
    $marks = array(
        "ASHISH" => array (
            "HTML" => 98,
            "JAVA" => 78,
            "PHP" => 96
        ),
        "LINA" => array (
            "HTML" => 75,
            "JAVA" => 86,
            "PHP" => 72
        )
    );
```

```

),

"RIYA" => array (
    "HTML" =>rand(1,100),
    "JAVA" =>rand(1,100),
    "PHP" =>rand(1,100)
)
);
// In nested array of Riya's all marks are random range is 1 to 100
/* Accessing multi-dimensional array values */
echo "Marks for ASHISH in PHP: “;
echo $marks['ASHISH']['PHP'] . "<br />";

echo "Marks for LINA in JAVA: ";
echo $marks['LINA']['JAVA'] . "<br />";

echo "Marks for RIYA in HTML: “;
echo $marks['RIYA']['HTML'] . "<br />";
?>

```

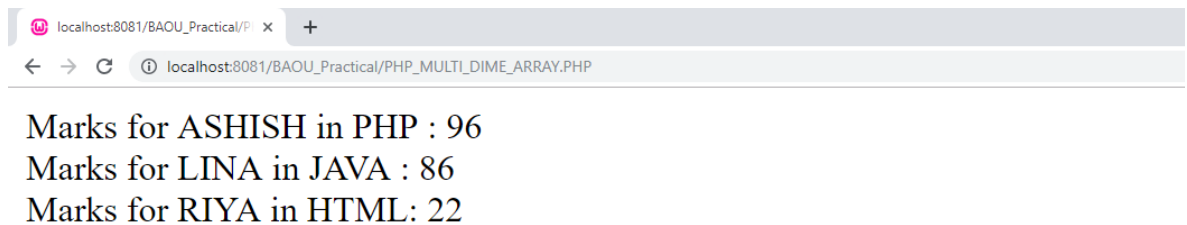


Figure 45 Output of PHP_MULTI_DIME_ARRAY.PHP using Riya's marks as random

Using this method data can be stored in multidimensional arrays, PHP understands multidimensional arrays that are two, three, four, five, or more levels deep. However, arrays more than three levels deep are hard to manage for most people.

4.7 PHP ARRAY OPERATORS

The PHP array operators are used to compare arrays.

Operator	Name	Example	Result
+	Union	\$x + \$y	Union of \$x and \$y
==	Equality	\$x == \$y	Returns true if \$x and \$y have the same key/value pairs
===	Identity	\$x === \$y	Returns true if \$x and \$y have the same key/value pairs in the same order and of the same types
!=	Inequality	\$x != \$y	Returns true if \$x is not equal to \$y
<>	Inequality	\$x <> \$y	Returns true if \$x is not equal to \$y
!==	Non-identity	\$x !== \$y	Returns true if \$x is not identical to \$y

4.8 MANIPULATING ARRAYS

Joining the Arrays

The best way to merge two or more arrays in PHP is to use the `array_merge()` function. Items of arrays will be merged together, and values with the same string keys will be overwritten with the last value:

```
<?php //Script Save as PHP_Array_Join.php
    $array1 = ['a' => 'a', 'b' => 'b', 'c' => 'c'];
    $array2 = ['m' => 'M', 'n' => 'N', 'o' => 'O'];
    echo "Array 1:-";
    print_r($array1);
    echo "<br>Array 2:-";
    print_r($array2);
    $merge = array_merge($array1, $array2);
    echo "<br><hr> Result after merge of Array1 & Array2";
    echo "<pre>";
    print_r($merge);
    echo "</pre>";
?>
```

```
localhost:8081/BAOU_Practical/P x +
localhost:8081/BAOU_Practical/PHP_Array_Join.php

Array 1:-Array ( [a] => a [b] => b [c] => c )
Array 2:-Array ( [m] => M [n] => N [o] => O )

Result after mearge of Array1 & Array2

Array
(
    [a] => a
    [b] => b
    [c] => c
    [m] => M
    [n] => N
    [o] => O
)
```

Figure 46 output of PHP_Array_Join.php

To remove array values from another array (or arrays), use `array_diff()`. To get values which are present in given arrays, use `array_intersect()`. The next examples will show how it works:

```
<?php //Script Save as PHP_Array_diff_inter.php

$array1 = [1, 2, 3, 4];

$array2 = [3, 4, 5, 6];

echo "Array 1:-";

print_r($array1);

echo "<br>Array 2:-";

print_r($array2);

echo "<br><br> Result after array difference of Array1 & Array2";

$diff = array_diff($array1, $array2);

echo "<pre>";

print_r($diff); // [0 => 1, 1 => 2]

echo "</pre>";
```

```

echo "<hr> Result after array intersection of Array1 & Array2";

$intersect = array_intersect($array1, $array2);

echo "<pre>";

print_r($intersect); // [2 => 3, 3 => 4]

echo "</pre>";

?>

```

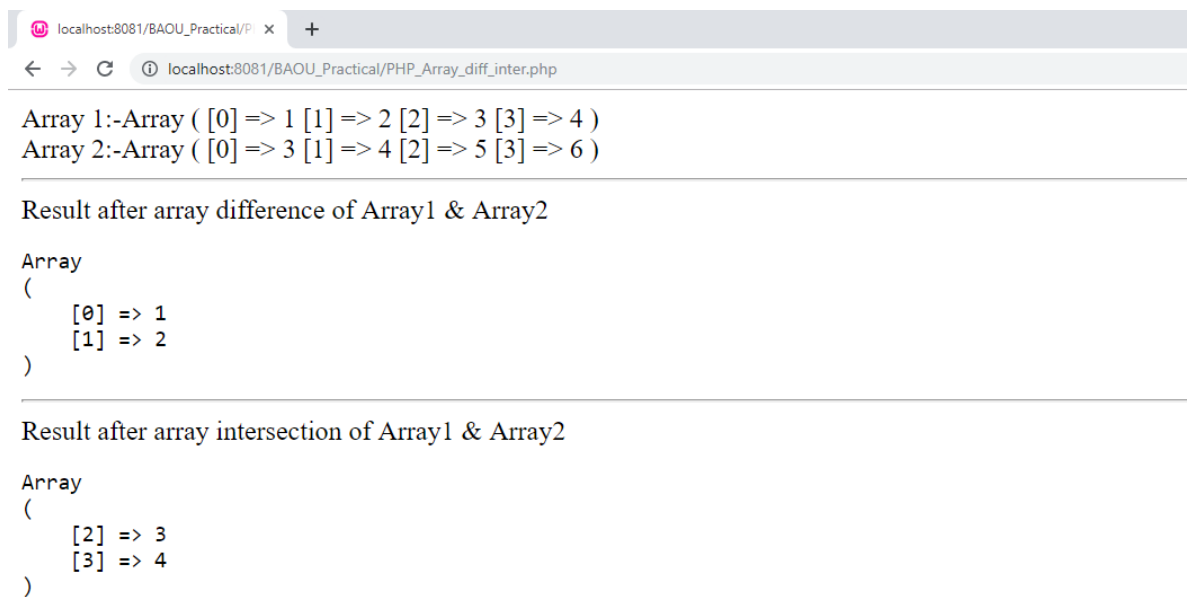


Figure 47 Output of PHP_Array_diff_inter.php

sizeof(\$arr) :

This function returns the number of elements in an array. Use this function to find out how many elements an array contains; this information is most commonly used to initialize a loop counter when processing the array.

```

<?php
    $data = array("red", "green", "blue");
    echo "Array has " . sizeof($data) . " elements";

?>
Output:Array has 3 elements

```

array_values(\$arr):

This function accepts a PHP array and returns a new array containing only its values (not its keys). Its counterpart is the `array_keys()` function.

Use this function to retrieve all the values from an associative array.

```
<?php
$data = array("hero" => "Holmes", "villain" => "Moriarty");
print_r(array_values($data));
?>
```

Output:

```
Array
(
    [0] => Holmes
    [1] => Moriarty
)
```

`array_keys($arr):`

This function accepts a PHP array and returns a new array containing only its keys (not its values). Its counterpart is the `array_values()` function.

Use this function to retrieve all the keys from an associative array.

```
<?php
$data = array("hero" => "Holmes", "villain" => "Moriarty");
print_r(array_keys($data));
?>
```

Output:

```
Array
(
    [0] => hero
    [1] => villain
)
```

array_pop(\$arr):

This function removes an element from the end of an array.

```
<?php
    $data = array("Donald", "Jim", "Tom");
    array_pop($data);
    print_r($data);
?>
```

Output:

```
Array
(
    [0] => Donald
    [1] => Jim
)
```

array_push(\$arr, \$val):

This function adds an element to the end of an array.

```
<?php
    $data = array("Donald", "Jim", "Tom");
    array_push($data, "Harry");
    print_r($data);
?>
```

Output:

```
Array
(
```

```
[0] => Donald
[1] => Jim
[2] => Tom
[3] => Harry
)
```

array_shift(\$arr):

This function removes an element from the beginning of an array.

```
<?php
    $data = array("Donald", "Jim", "Tom");
    array_shift($data);
    print_r($data);
?>
```

Output:

```
Array
(
    [0] => Jim
    [1] => Tom
)
```

array_unshift(\$arr, \$val):

This function adds an element to the beginning of an array.

```
<?php
    $data = array("Donald", "Jim", "Tom");
    array_unshift($data, "Sarah");
```

```
        print_r($data);  
?>
```

Output:

Array

```
(  
    [0] => Sarah  
    [1] => Donald  
    [2] => Jim  
    [3] => Tom  
)
```

each(\$arr):

This function is most often used to iteratively traverse an array. Each time each() is called, it returns the current key-value pair and moves the array cursor forward one element. This makes it most suitable for use in a loop.

```
<?php  
    $data = array("hero" => "Holmes", "villain" => "Moriarty");  
    while (list($key, $value) = each($data)) {  
        echo "$key: $value \n";  
    }  
?>
```

Output:

hero: Holmes

villain: Moriarty

sort(\$arr):

This function sorts the elements of an array in ascending order. String values will be arranged in ascending alphabetical order.

Note: Other sorting functions include `asort()`, `arsort()`, `ksort()`, `krsort()` and `rsort()`.

```
<?php
    $data = array("g", "t", "a", "s");
    sort($data);
    print_r($data);
?>
```

Output:

Array

```
(
    [0] => a
    [1] => g
    [2] => s
    [3] => t
)
```

array_flip(\$arr):

The function exchanges the keys and values of a PHP associative array.

Use this function if you have a tabular (rows and columns) structure in an array, and you want to interchange the rows and columns.

```
<?php
    $data = array("a" => "apple", "b" => "ball");
    print_r(array_flip($data));
?>
```

Output:

Array

```
(  
    [apple] => a  
    [ball] => b  
)
```

array_reverse(\$arr):

The function reverses the order of elements in an array.

Use this function to re-order a sorted list of values in reverse for easier processing—for example, when you're trying to begin with the minimum or maximum of a set of ordered values.

```
<?php  
    $data = array(10, 20, 25, 60);  
    print_r(array_reverse($data));  
?>
```

Output:

Array

```
(  
    [0] => 60  
    [1] => 25  
    [2] => 20  
    [3] => 10  
)
```

array_merge(\$arr):

This function merges two or more arrays to create a single composite array. Key collisions are resolved in favor of the latest entry.

Use this function when you need to combine data from two or more arrays into a single structure—for example, records from two different SQL queries.

```
<?php
    $data1 = array("cat", "goat");
    $data2 = array("dog", "cow");
    print_r(array_merge($data1, $data2));
?>
```

Output:

```
Array
(
    [0] => cat
    [1] => goat
    [2] => dog
    [3] => cow
)
```

array_rand(\$arr):

This function selects one or more random elements from an array.

Use this function when you need to randomly select from a collection of discrete values—for example, picking a random color from a list.

```
<?php
    $data = array("white", "black", "red");
    echo "Today's color is " . $data[array_rand($data)];
?>
```

Output:

```
Today's color is red
```

array_search(\$search, \$arr):

This function searches the values in an array for a match to the search term, and returns the corresponding key if found. If more than one match exists, the key of the first matching value is returned.

Use this function to scan a set of index-value pairs for matches, and return the matching index.

```
<?php
$data = array("blue" => "#0000cc", "black" => "#000000", "green" =>
"#00ff00");
echo "Found " .array_search("#0000cc", $data);
?>
```

Output:

Found blue

array_slice(\$arr, \$offset, \$length):

This function is useful to extract a subset of the elements of an array, as another array. Extracting begins from array offset \$offset and continues until the array slice is \$length elements long.

Use this function to break a larger array into smaller ones—for example, when segmenting an array by size ("chunking") or type of data.

```
<?php
$data = array("vanilla", "strawberry", "mango", "peaches");
print_r(array_slice($data, 1, 2));
?>
```

Output:

Array


```
(  
    [0] => strawberry  
    [1] => mango  
)
```

array_unique(\$data):

This function strips an array of duplicate values.

Use this function when you need to remove non-unique elements from an array—for example, when creating an array to hold values for a table's primary key.

```
<?php  
    $data = array(1,1,4,6,7,4);  
    print_r(array_unique($data));  
?>
```

Output:

```
Array  
  
(  
    [0] => 1  
    [3] => 6  
    [4] => 7  
    [5] => 4  
)
```

array_walk(\$arr, \$func):

This function "walks" through an array, applying a user-defined function to every element. It returns the changed array.

Use this function if you need to perform custom processing on every element of an array—for example, reducing a number series by 10%.

```
<?php
```

```

function reduceBy10(&$val, $key) {
    $val -= $val * 0.1;
}

$data = array(10,20,30,40);
array_walk($data, 'reduceBy10');
print_r($data);

?>

```

Output:

```

Array
(
    [0] => 9
    [1] => 18
    [2] => 27
    [3] => 36
)

```

All above methods are help for manipulate array.

4.9PHP ARRAY CONSTANTS

These functions allow you to interact with and manipulate arrays in various ways. Arrays are essential for storing, managing, and operating on sets of variables.

Constant	Description
CASE_LOWER	Used with <code>array_change_key_case()</code> to convert array keys to lower case
CASE_UPPER	Used with <code>array_change_key_case()</code> to convert array keys to upper case

SORT_ASC	Used with <code>array_multisort()</code> to sort in ascending order
SORT_DESC	Used with <code>array_multisort()</code> to sort in descending order
SORT_REGULAR	Used to compare items normally
SORT_NUMERIC	Used to compare items numerically
SORT_STRING	Used to compare items as strings
SORT_LOCALE_STRING	Used to compare items as strings, based on the current locale

And other normal constants are as below, for more detail and detail example you can refer php.net

COUNT_NORMAL,
COUNT_RECURSIVE,
EXTR_OVERWRITE,
EXTR_SKIP,
EXTR_PREFIX_SAME,
EXTR_PREFIX_ALL,
EXTR_PREFIX_INVALID,
EXTR_PREFIX_IF_EXISTS,
EXTR_IF_EXISTS,
EXTR_REFS

4.10 LIST OF ARRAY FUNCTIONS

Function	Description
array()	Creates an array
array_change_key_case()	Changes all keys in an array to lowercase or uppercase
array_chunk()	Splits an array into chunks of arrays
array_column()	Returns the values from a single column in the input array
array_combine()	Creates an array by using the elements from one "keys" array and one "values" array
array_count_values()	Counts all the values of an array
array_diff()	Compare arrays, and returns the differences (compare values only)

array_diff_assoc()	Compare arrays, and returns the differences (compare keys and values)
array_diff_key()	Compare arrays, and returns the differences (compare keys only)
array_diff_uassoc()	Compare arrays, and returns the differences (compare keys and values, using a user-defined key comparison function)
array_diff_ukey()	Compare arrays, and returns the differences (compare keys only, using a user-defined key comparison function)
array_fill()	Fills an array with values
array_fill_keys()	Fills an array with values, specifying keys
array_filter()	Filters the values of an array using a callback function
array_flip()	Flips/Exchanges all keys with their associated values in an array
array_intersect()	Compare arrays, and returns the matches (compare values only)
array_intersect_assoc()	Compare arrays and returns the matches (compare keys and values)
array_intersect_key()	Compare arrays, and returns the matches (compare keys only)
array_intersect_uassoc()	Compare arrays, and returns the matches (compare keys and values, using a user-defined key comparison function)
array_intersect_ukey()	Compare arrays, and returns the matches (compare keys only, using a user-defined key comparison function)
array_key_exists()	Checks if the specified key exists in the array
array_keys()	Returns all the keys of an array
array_map()	Sends each value of an array to a user-made function, which returns new values
array_merge()	Merges one or more arrays into one array
array_merge_recursive()	Merges one or more arrays into one array

	recursively
array_multisort()	Sorts multiple or multi-dimensional arrays
array_pad()	Inserts a specified number of items, with a specified value, to an array
array_pop()	Deletes the last element of an array
array_product()	Calculates the product of the values in an array
array_push()	Inserts one or more elements to the end of an array
array_rand()	Returns one or more random keys from an array
array_reduce()	Returns an array as a string, using a user-defined function
array_replace()	Replaces the values of the first array with the values from following arrays
array_replace_recursive()	Replaces the values of the first array with the values from following arrays recursively
array_reverse()	Returns an array in the reverse order
array_search()	Searches an array for a given value and returns the key
array_shift()	Removes the first element from an array, and returns the value of the removed element
array_slice()	Returns selected parts of an array
array_splice()	Removes and replaces specified elements of an array
array_sum()	Returns the sum of the values in an array
array_udiff()	Compare arrays, and returns the differences (compare values only, using a user-defined key comparison function)
array_udiff_assoc()	Compare arrays, and returns the differences (compare keys and values, using a built-in function to compare the keys and a user-defined function to compare the values)
array_udiff_uassoc()	Compare arrays, and returns the differences (compare keys and values, using two user-defined key comparison functions)

array_uintersect()	Compare arrays, and returns the matches (compare values only, using a user-defined key comparison function)
array_uintersect_assoc()	Compare arrays, and returns the matches (compare keys and values, using a built-in function to compare the keys and a user-defined function to compare the values)
array_uintersect_uassoc()	Compare arrays, and returns the matches (compare keys and values, using two user-defined key comparison functions)
array_unique()	Removes duplicate values from an array
array_unshift()	Adds one or more elements to the beginning of an array
array_values()	Returns all the values of an array
array_walk()	Applies a user function to every member of an array
array_walk_recursive()	Applies a user function recursively to every member of an array
arsort()	Sorts an associative array in descending order, according to the value
asort()	Sorts an associative array in ascending order, according to the value
compact()	Create array containing variables and their values
count()	Returns the number of elements in an array
current()	Returns the current element in an array
each()	Returns the current key and value pair from an array
end()	Sets the internal pointer of an array to its last element
extract()	Imports variables into the current symbol table from an array
in_array()	Checks if a specified value exists in an array
key()	Fetches a key from an array
krsort()	Sorts an associative array in descending order, according to the key

ksort()	Sorts an associative array in ascending order, according to the key
list()	Assigns variables as if they were an array
natcasesort()	Sorts an array using a case insensitive "natural order" algorithm
natsort()	Sorts an array using a "natural order" algorithm
next()	Advance the internal array pointer of an array
pos()	Alias of current()
prev()	Rewinds the internal array pointer
range()	Creates an array containing a range of elements
reset()	Sets the internal pointer of an array to its first element
rsort()	Sorts an indexed array in descending order
shuffle()	Shuffles an array
sizeof()	Alias of count()
sort()	Sorts an indexed array in ascending order
uasort()	Sorts an array by values using a user-defined comparison function
uksort()	Sorts an array by keys using a user-defined comparison function
usort()	Sorts an array using a user-defined comparison function

Block-4
Processing Web Forms and
Handling Database in PHP

Unit 1: Working with Forms in PHP

1

Unit Structure

- 1.1. Learning Objectives
- 1.2. Introduction
- 1.3. Working with forms in PHP
- 1.4. Validating input data
- 1.5. using magic quotes
- 1.6. Storing form data in file
- 1.7. Saving form data using cookies
- 1.8. saving form data using sessions

1.1 LEARNING OBJECTIVE

After studying this unit student should be able to:

- understand dynamic web site creation using form
- understand validation controls to provide validation on input
- understand saving data using cookie and session

1.2 INTRODUCTION

Dynamic Websites: The Websites provide the functionalities that can use to store, update, retrieve, and delete the data in a database.

What is the Form?

A Document that containing black fields, that the user can fill the data or user can select the data and Casually the data will store in the database, file and session.

PHP Data Collection method is: GET, POST, REQUEST

It is time to apply the knowledge you have obtained thus far and put it to real use. A very common application of PHP is to have an HTML form gather information from a website's visitor and then use PHP to do process that information. In this lesson we will simulate a small business's website that is implementing a very simple order form.

Imagine we are an education community store that participant personal information, education detail and communication. To gather order information from our prospective students or participant we will have to make a page with an HTML form to gather the participant's information for the events.

1.3WORKING WITH FORMS IN PHP

We first create an HTML form that will collectstudent's basic information eg. Name, email, education. When the user fills out the form above and clicks the submit button,the form data is sent for processing to a PHP file named "**baou_form.php**" using action attribute value. The form data is sent with the HTTP GET method.

If you need a refresher on how to properly make an HTML form, recall the HTML Form tags before continuing to create PHP from.

The example below displays a simple HTML form with three input fields and a submit button:

```
<html>
<body>
<!--file save as form_demo.html -->
<h1> HTML From demo in PHP </h1><hr>
    <form action="baou_form.php" method="post">
        Name: <input type="text" name="name"><br><br>
        E-mail: <input type="text" name="email"><br><br>
        Education:
            <select name="edu">
                <option selected>select Education</option>
                <option selected>M.Sc-IT</option>
                <option selected>BCA</option>
                <option selected>PGDCA</option>
                <option selected>B.Sc-IT</option>
            </select><br><br>
        <input type="submit">
    </form>
</body>
</html>
```

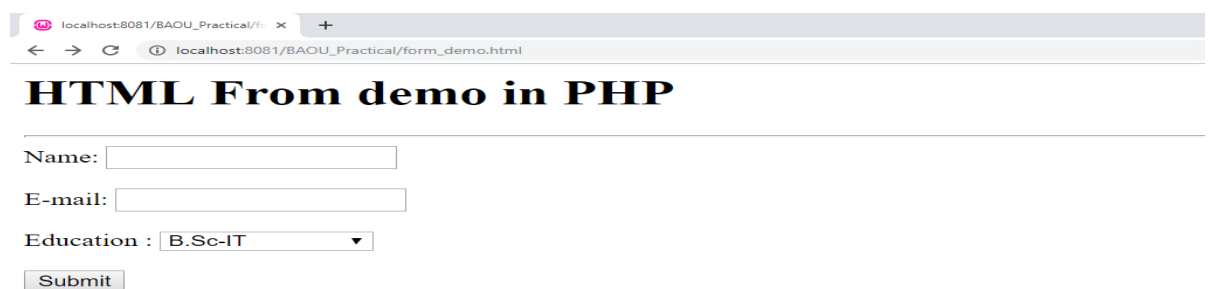


Figure 48 output of form_demo.html (Only HTML as response)

To display the submitted data you could simply echo all the variables. The "baou_form.php" looks like this:

```

<html>
  <body>
    Welcome <?php echo $_POST["name"]; ?>, <br>
    Your email address is: <?php echo $_POST["email"]; ?><br>
    Your Education is: <?php echo $_POST["edu"]; ?><br>
  </body>
</html>

```

As you probably noticed, the name in `$_POST['name']` corresponds to the name that we specified in our HTML form.

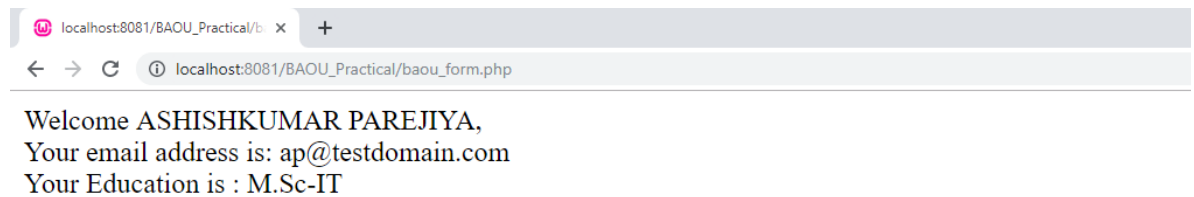


Figure 49 output of form_demo.html &baou_form.php

Note: In the above example you can passing same data using GET method,

The GET Method:

GET is used to request data from a specified resource. GET is one of the most common HTTP methods.

Note that the query string (name/value pairs) is sent in the URL of a GET request:

`/test/form_demo.php?name1=value1&name2=value2`

- GET requests can be cached
- GET requests remain in the browser history
- GET requests can be bookmarked
- GET requests should never be used when dealing with sensitive data
- GET requests have length restrictions(maximum URL length is 2048 characters)
- GET requests is only used to request data (not modify)

The POST Method:

Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request. POST is used to send data to a server to create/update a resource. The data sent to the server with POST is stored in the request body of the HTTP request: (in chrome press F12>network>all you can see below output)

```
POST /test/form_demo.php HTTP/1.1
```

```
Host: baou.edu.in
```

```
name1=value1&name2=value2
```

- POST is one of the most common HTTP methods for parsing large amount of data.
- POST requests are never cached
- POST requests do not remain in the browser history
- POST requests cannot be bookmarked
- POST requests have no restrictions on data length

Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.

The following table compares the two HTTP methods: GET and POST.

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history

Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

1.4 VALIDATING INPUT DATA

What is Validation?

Validation means check the input submitted by the user. There are two types of validation are available in PHP. Mentioned as below

- **Client-Side Validation:** Validation is performed on the client machine web browsers.
- **Server-Side Validation:** After submitted by data, the data has sent to a server and perform validation checks in server machine.

Required field will check whether the field is filled or not in the proper way. Most of cases we will use the * symbol for required field.

Some of Validation rules for field

Field	Validation Rules
Name	Should required letters and white-spaces

Email	Should required @ and .
Website	Should required a valid URL
Radio	Must be selectable at least once
Check Box	Must be checkable at least once
Drop Down menu	Must be selectable at least once

Validate Name :The code below shows a simple way to check if the name field only contains letters and whitespace. If the value of the name field is not valid, then store an error message:

```
$name = test_input($_POST["name"]);
if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
    $nameErr = "Only letters and white space allowed";
}
```

Valid URLBelow code shows validation of URL:

```
$website = input($_POST["site"]);
if(!preg_match("/\b(?:?:https?|ftp):\/\/www\.)[-a-z0-9+&@#V%?~_!|:,;]*[-a-z0-9+&@#V%~_]/i",$website)) {
    $websiteErr = "Invalid URL";
}
```

Above syntax will verify whether a given URL is valid or not. It should allow some keywords as https, ftp, www, a-z, 0-9,..etc..

Valid EmailBelow code shows validation of Email address:

```
$email = input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    $emailErr = "Invalid format and please re-enter valid email";
}
```

Above syntax will verify whether given Email address is well-formed or not. If it is not, it will show an error message.

Let's discuss the example of form fields validation:

```
<!--Script save as form_validate.php ->
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>
<?php
    // define variables and set to empty values
    $nameErr = $emailErr = $genderErr = $websiteErr = "";
    $name = $email = $gender = $comment = $website = "";
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        if (empty($_POST["name"])) {
            $nameErr = "Name is required";
        }else {
            $name = test_input($_POST["name"]);
        }
        if (empty($_POST["email"])) {
            $emailErr = "Email is required";
        }else {
            $email = test_input($_POST["email"]);
```



```

// check if e-mail address is well-formed
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    $emailErr = "Invalid email format";
}
}

if (empty($_POST["website"])) {
    $website = "";
}
}

}else {
    $website = test_input($_POST["website"]);
}

if (empty($_POST["comment"])) {
    $comment = "";
}
}

}else {
    $comment = test_input($_POST["comment"]);
}

if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
}
}

}else {
    $gender = test_input($_POST["gender"]);
}
}

function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}

```

```
}
```

```
?>
```

```
<h2>Absolute classes registration</h2>
```

```
<p><span class = "error">* required field.</span></p>
```

```
<form method = "post" action = "<?php
```

```
    echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

```
<table>
```

```
<tr>
```

```
<td>Name:</td>
```

```
<td><input type = "text" name = "name">
```

```
<span class = "error">* <?php echo $nameErr;?></span>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td>E-mail: </td>
```

```
<td><input type = "text" name = "email">
```

```
<span class = "error">* <?php echo $emailErr;?></span>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td>Time:</td>
```

```
<td><input type = "text" name = "website">
```

```
<span class = "error"><?php echo $websiteErr;?></span>
```

```
</td>
```

```
</tr>
```

```

<tr>
<td>About you:</td>
<td><textarea name = "comment" rows = "5" cols = "40"></textarea></td>
</tr>
<tr>
<td>Gender:</td>
<td>
<input type = "radio" name = "gender" value = "female">Female
<input type = "radio" name = "gender" value = "male">Male
<input type = "radio" name = "gender" value =
"others">others
<span class = "error">* <?php echo $genderErr;?></span>
</td>
</tr>
<td>
<input type = "submit" name = "submit" value = "Submit">
</td>
</table>
</form>
<?php
    echo "<h2>Your entered values are:</h2>";
    echo $name;
    echo "<br>";
    echo $email;
    echo "<br>";
    echo $website;

```

```
        echo "<br>";  
        echo $comment;  
        echo "<br>";  
        echo $gender;  
    ?>  
</body>  
</html>
```

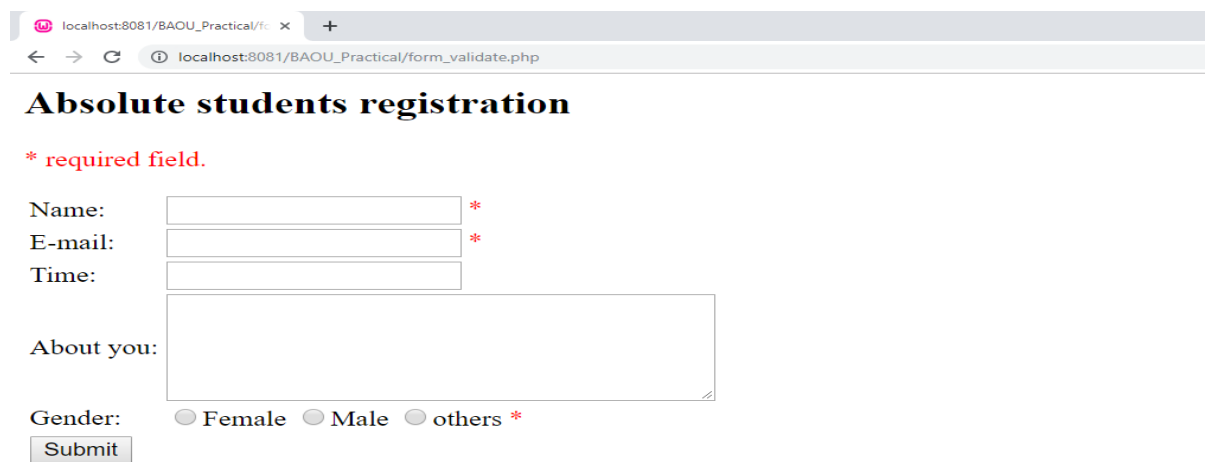


Figure 50 First output of form_validate.php (When access first time)

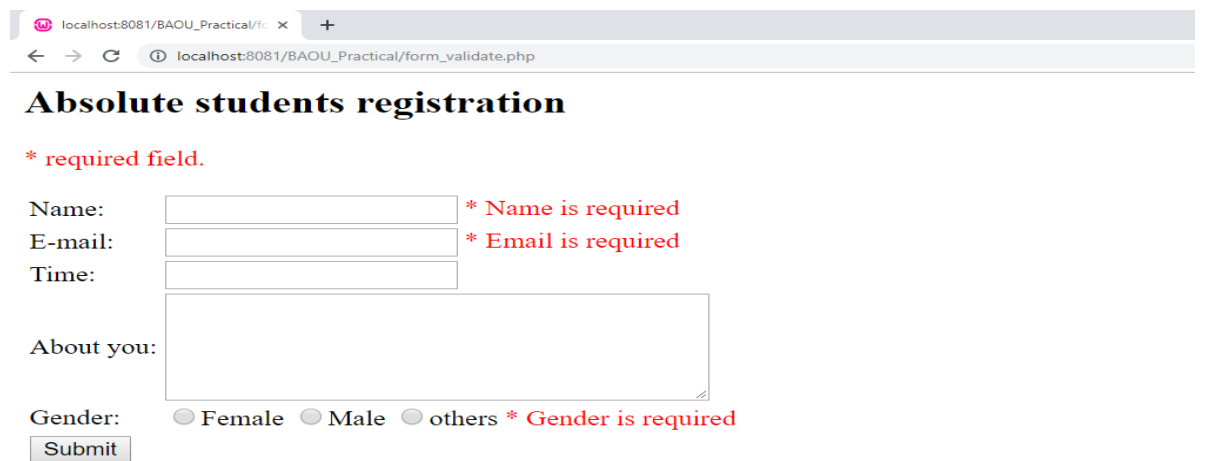


Figure 51 Click on submit button without entering data, it is shows * marks fields are required

localhost:8081/BAOU_Practical/fo x +
 localhost:8081/BAOU_Practical/form_validate.php

Absolute students registration

* required field.

Name: *

E-mail: *

Time:

About you:

Gender: Female Male others *

Your entered values are:

Pratik P
 pratik@testdomain.com
 03:00PM
 I am student of M.Sc-IT
 male

Figure 52 output of entered sample data

1.5USING MAGIC QUOTES

PHP - magic quotes: Prior to PHP 6 there was a feature called magic quotes that was created to help protect newbie programmers from writing bad form processing code. Magic quotes would automatically escape risky form data that might be used for SQL Injection with a backslash \. The characters escaped by PHP include: quote ', double quote ", backslash \ and NULL characters.

However, this newbie protection proved to cause more problems than it solved and is not in PHP 6. If your PHP version is any version before 6 then you should use this topicsto learn more about how magic quotes can affect you.

magic quotes - are they enabled?

First things first, you need to check to see if you have magic quotes enabled on you server. The `get_magic_quotes_gpc` function will return a 0 (off) or a 1 (on). These boolean values will fit nicely into an if statement where 1 is true and 0 is false.

PHP Code:

```
if(get_magic_quotes_gpc())
    echo "Magic quotes are enabled";
else
    echo "Magic quotes are disabled";
```

Output:

Magic quotes are enabled

If you received the message "Magic quotes are enabled" then you should definitely continue reading this topic, if not feel free to learn about it in case you are developing for servers that might have quotes on or off.

magic quotes in action: Now let's make a simple form processor to show how machines with magic quotes enabled will escape those potentially risky characters. This form submits to itself, so you only need to make one file, "magic-quotes.php" to test it out.

magic-quotes code:

```
<!-- Script save as magic-quotes.php -->

<?php
    echo "Altered Text: ".$_POST['question'];
?>

<form method='post' action="#"><! -- # char is return page itself value -->
    Question: <input type='text' name='question' /><br />
    <input type='submit' />
</form>
```

This simple form will display to you what magic quotes is doing. If you were to enter and submit the string: Sandy said, "It's a beautiful day outside and I like to use \"'s." You would receive the following output.

Output:

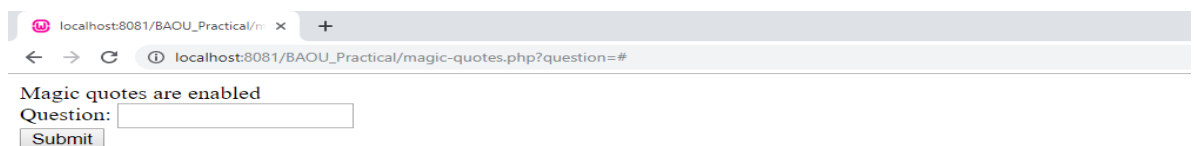


Figure 53 Magic quotes are enable and display output.

Magic quotes did a number on that string, didn't it? Notice that there is a backslash before all of those risky characters we talked about earlier. After magic quotes:

A backslash \ becomes \\

A quote ' becomes \'

A double-quote " becomes \"

Now say that you wanted to remove the escaping that magic quotes puts in, you have two options: disable magic quotes or strip the backslashes magic quotes adds.

removing backslashes - stripslashes()

Before you use PHP's backslash removal function stripslashes it's smart to add some magic quote checking like our "Are They Enabled?" section above. This way you won't accidentally be removing slashes that are legitimate in the future if your PHP's magic quotes setting changes in the future.

magic-quotes.php Code:

```
<?php
echo "Removed Slashes: ";
// Remove those slashes
if(get_magic_quotes_gpc())
    echo stripslashes($_POST['question']);
else
    echo $_POST['question'];
?>
<form method='post'>
Question: <input type='text' name='question'/><br />
<input type='submit'>
</form>
```

You can run above code and generate same output with magic quotes enable and disabled base.

Magic Quotes is a process that automatically escapes incoming data to the PHP script. This feature has been DEPRECATED as of PHP 5.3.0 and REMOVED as of PHP 5.4.0. It's preferred to code with magic quotes off and to instead escape the data at runtime, as needed.

You can enable/disable `magic_quotes_gpc` in `php.ini` & `.htaccess` depending on how `php/apache` are compiled.

1. Enable/disable using `php.ini`.

Copy server's `php.ini` under your `public_html`. Find for `magic_quotes_gpc`. Set it to On OR Off as required.

Now, open `.htaccess` and add `"SetEnv PHPRC /home/user/public_html"`. Doing this, the `php.ini` will be used by all files & directories. Not just the main directory.

2. Enable/disable using `.htaccess`.

Adding the following line will disable it. Change from off to on if you want it to be enabled.

1.6 STORING FORM DATA IN FILE

In this topic you will learn essential task, storing form data in text file or row file, how can be easily done with Core PHP and HTML form. Sometimes it happens that we need to store some data in local storage file rather than making it complex using the database. Yes, it's a fact that in many cases we don't want to store our text data in database always.

Here is an example, suppose you have an HTML form and you want to store the data submitted by the user in a text file so that you can easily access it later from that file without opening your database.

PHP Program to store HTML Form data in a .txt File

Below I have provided the PHP code to store the form data in a text file. Just took a glance at this code.

For easy understanding after the code, below provided the explanation and how to use this code step by step.

Example:

```
<?php
```



```
// Script save as store_form_data_to_text.php
if(isset($_POST['textdata']))
{
    $data=$_POST['textdata'];
    $fp = fopen('data.txt', 'a');
    fwrite($fp, $data);
    fclose($fp);
}
?>
```

Here **'textdata'** is the name of our HTML form field that is provided below. **data.txt** is a file that we have to create for storing our form submission data in it. **\$data** is a PHP variable to store the form field data entered by the user. Now the HTML part

```
<!DOCTYPE html>
<html>
<head>
<title>Store form data in .txt file</title>
</head>
<body>
    <form method="post">
        Enter Your Text Here:<br>
        <input type="text" name="textdata"><br>
        <input type="submit" name="submit">
    </form>
</body>
</html>
```

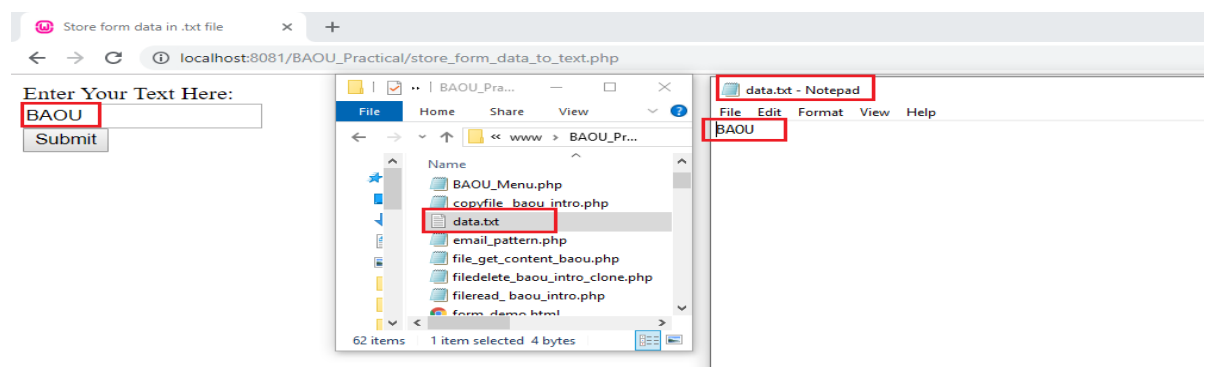


Figure 54 Output of store_form_data_to_text.php> Data.txt file will create and store contain BAOU store in same directory of application

1.7 SAVING FORM DATA USING COOKIES

Cookies are text files stored on the client computer and they are kept of use tracking purpose. PHP transparently supports HTTP cookies.

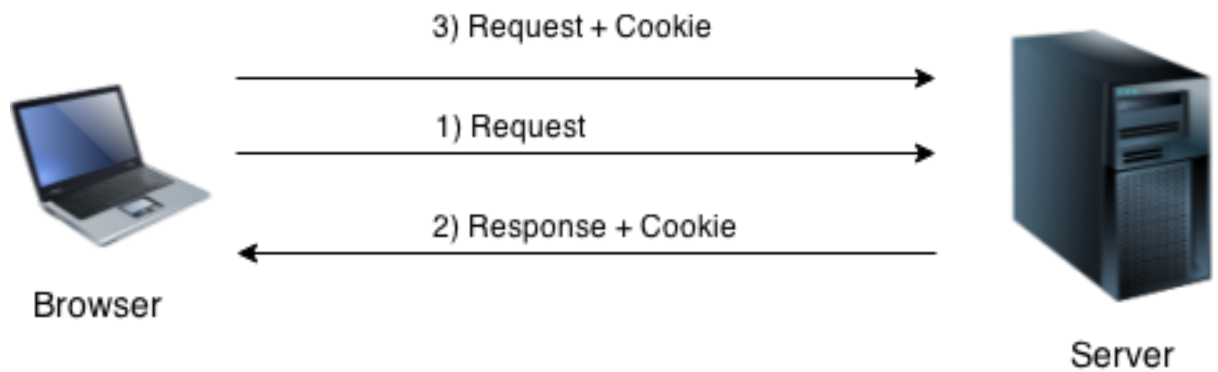
There are three steps involved in identifying returning users :

- Server script sends a set of cookies to the browser. For example: name, age, or identification number etc.
- Browser stores this information on local machine for future use.
- When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

A cookie is a small file with the maximum size of 4KB that the web server stores on the client computer. Once a cookie has been set, all page requests that follow return the cookie name and value. A cookie can only be read from the domain that it has been issued from. For example, a cookie set using the domain **www.baou.edu.in** cannot be read from the domain **www.student.baou.edu.in**.

Most of the websites on the internet display elements from other domains such as advertising. The domains serving these elements can also set their own cookies. These are known as third party cookies. A cookie created by a user can only be visible to them. Other users cannot see its value. Most web browsers have options for disabling cookies, third party cookies or both. If this is the case then PHP responds by passing the cookie token in the URL.

Cookie is created at server side and saved to client browser. Each time when client sends request to the server, cookie is embedded with request. Such way, cookie can be received at the server side.



In short, cookie can be created, sent and received at server end.

This chapter you will learn,

- how to set cookies?
- how to access them?
- how to delete or remove them.

The Anatomy of a Cookie

Cookies are usually set in an HTTP header (although JavaScript can also set a cookie directly on a browser). A PHP script that sets a cookie might send headers that look something like this –

```

HTTP/1.1 200 OK
Date: Fri, 04 Feb 2000 21:03:38 GMT
Server: Apache/1.3.9 (UNIX) PHP/4.0b3
Set-Cookie: name=xyz; expires=Friday, 04-JUN-1921:03:38 GMT;
           path=/; domain=www.baou.edu.in
Connection: close
Content-Type: text/html
  
```

As you can see, the Set-Cookie header contains a name value pair, a GMT date, a path and a domain. The name and value will be URL encoded. The expires field is an instruction to the browser to "forget" the cookie after the given time and date.

If the browser is configured to store cookies, it will then keep this information until the expiry date. If the user points the browser at any page that matches the path and domain of the cookie, it will resend the cookie to the server. The browser's headers might look something like this –

```

GET / HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.6 (X11; I; Linux 2.2.6-15apmac ppc)
  
```

Host: test.demon.co.in:1126
Accept: image/gif, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Cookie: name=xyz

A PHP script will then have access to the cookie in the environmental variables `$_COOKIE` or `$HTTP_COOKIE_VARS[]` which holds all cookie names and values. Above cookie can be accessed using `$HTTP_COOKIE_VARS["name"]`.

Why and when to use Cookies?

- Http is a stateless protocol; cookies allow us to track the state of the application using small files stored on the user's computer.
- The path where the cookies are stored depends on the browser.
- Internet Explorer usually stores them in Temporal Internet Files folder.
- Personalizing the user experience – this is achieved by allowing users to select their preferences.
- The page requested that follow are personalized based on the set preferences in the cookies.
- Tracking the pages visited by a user

Setting Cookies with PHP

PHP provided `setcookie()` function to set a cookie. This function requires upto six arguments and should be called before `<html>` tag. For each cookie this function has to be called separately.

Syntax

```
bool setcookie( string $name [, string $value [, int $expire = 0 [, string $path  
[, string $domain [, bool $secure = false [, bool $httponly = false ]]]]] )
```

Example

- `setcookie(name, value, expire, path, domain, security);`
- `setcookie("CookieName", "CookieValue");/* defining name and value only*/`
- `setcookie("CookieName", "CookieValue", time()+1*60*60);//using expiry in 1 h
our(1*60*60 seconds or 3600 seconds)`

- `setcookie("CookieName", "CookieValue", time()+1*60*60, "/mypath/", "mydomain.com", 1);`

Here is the detail of all the arguments description:

Name: This sets the name of the cookie and is stored in an environment variable called HTTP_COOKIE_VARS. This variable is used while accessing cookies.

Value: This sets the value of the named variable and is the content that you actually want to store.

Expiry: This specifies a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.

Path: This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.

Domain: This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.

Security: This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which means cookie can be sent by regular HTTP.

Following example will create two cookies name and age these cookies will be expired after one hour.

```
<?php
// Script save as set_cookie.php
setcookie("name", "BAOU Visit", time()+3600, "/", "", 0);
setcookie("age", "36", time()+3600, "/", "", 0);
?>
<html>
<head>
<title>Setting Cookies with PHP</title>
</head>
<body>
<?php echo "Set Cookies"?>
</body>
</html>
```

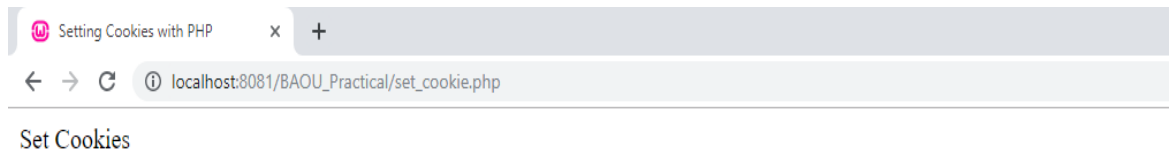


Figure 55 output of set_cookie.php file

Accessing Cookies with PHP

PHP provides many ways to access cookies. Simplest way is to use either `$_COOKIE` or `$HTTP_COOKIE_VARS` variables. Following example will access all the cookies set in above example.

```
<!-- Script save as access_cookie.php -->
<?php
    echo "Cookie set value: " . $_COOKIE["name"] . "<br />";
    /* is equivalent to */
    //echo $HTTP_COOKIE_VARS["name"] . "<br />";
    echo "Cookie Set value: " . $_COOKIE["age"] . "<br />";
    /* is equivalent to */
    //echo $HTTP_COOKIE_VARS["age"] . "<br />";
    ?>
<hr><b>You can use isset() function to check if a cookie is set or not.</b><hr>
<html>
<head>
<title>Accessing Cookies with PHP</title>
</head>
<body>
<?php
if( isset($_COOKIE["name"]))
    echo "<br>Welcome " . $_COOKIE["name"] . "<br />";
else
    echo "<br>Sorry... Not recognized" . "<br />";
    ?>
</body>
</html>
```

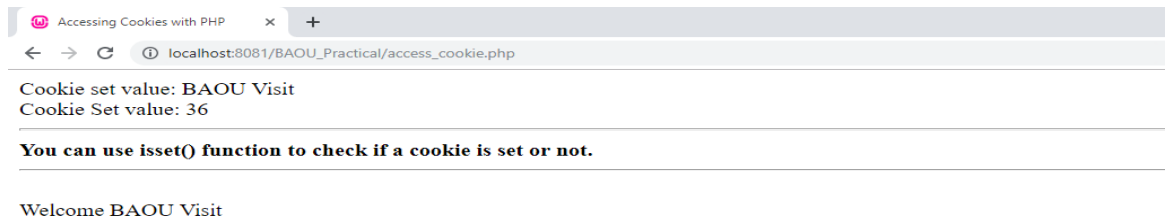


Figure 56 Output of access_cookie.php cookie set and access same value

Deleting or removing Cookie with PHP:

Officially, to delete a cookie you should call `setcookie()` with the name argument only but this does not always work well, however, and should not be relied on.

It is safest to set the cookie with a date that has already expired:

```
<?php
setcookie( "name", "", time()- 60, "/", "", 0);
setcookie( "age", "", time()- 60, "/", "", 0);
?>
<html>
<head>
<title>Deleting Cookies with PHP</title>
</head>
<body>
<?php echo "Deleted Cookies" ?>
</body>
</html>
```

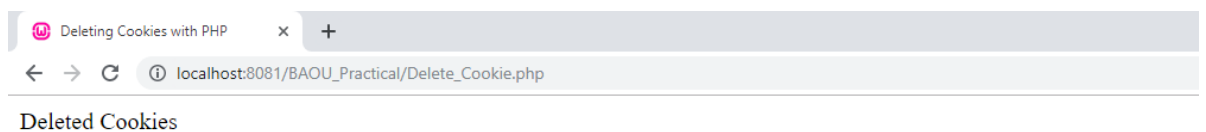


Figure 57 Output of delete_cookie.php

Note: You should pass exactly the same path, domain, and other arguments that you have used when you first created the cookie in order to ensure that the correct cookie is deleted.

1.8SAVING FORM DATA USING SESSIONS& TRACKING

What is a PHP Session?

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet, there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favoritecolor, etc). By default, session variables last until the user closes the browser.

So; Session variables hold information about one single user, and are available to all pages in one application.

- A session is a global variable stored on the server.
- Each session is assigned a unique id which is used to retrieve stored values.
- Whenever a session is created, a cookie containing the unique session id is stored on the user's computer and returned with every request to the server. If the client browser does not support cookies, the unique php session id is displayed in the URL
- Sessions have the capacity to store relatively large data compared to cookies.
- The session values are automatically deleted when the browser is closed. If you want to store the values permanently, then you should store them in the database.
- Just like the `$_COOKIE` array variable, session variables are stored in the `$_SESSION` array variable. Just like cookies, the session must be started before any HTML tags.

Note: A session is a way to store information (in variables) to be used across multiple pages. Unlike a cookie, the information is not stored on the user's computer.

An alternative way to make data accessible across the various pages of an entire website is to use a PHP Session.

A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit.

The location of the temporary file is determined by a setting in the php.ini file called session.save_path. Before using any session, variable make sure you have setup this path.

When a session is started following things happen:

- PHP first creates a unique identifier for that particular session which is a random string of 32 hexadecimal numbers such as 3c7foj34c3jj973hjkop2fc937e3443.
- A cookie called PHPSESSID is automatically sent to the user's computer to store unique session identification string.
- A file is automatically created on the server in the designated temporary directory and bears the name of the unique identifier prefixed by sess i.e. sess_3c7foj34c3jj973hjkop2fc937e3443.

When a PHP script wants to retrieve the value from a session variable, PHP automatically gets the unique session identifier string from the PHPSESSID cookie and then looks in its temporary directory for the file bearing that name and a validation can be done by comparing both values.

A session ends when the user loses the browser or after leaving the site, the server will terminate the session after a predetermined period of time, commonly 30 minutes duration.

Why and when to use Sessions?

- You want to store important information such as the user id more securely on the server where malicious users cannot temper with them.
- You want to pass values from one page to another.
- You want the alternative to cookies on browsers that do not support cookies.
- You want to store global variables in an efficient and more secure way compared to passing them in the URL
- You are developing an application such as a shopping cart that has to temporary store information with a capacity larger than 4KB.

Start a PHP Session

A session is started with the `session_start()` function. Session variables are set with the PHP global variable: `$_SESSION`.

Now, let's create a new PHP page called "baou_start_session.php" and we start a new PHP session and set some session variables:

Start session Example:

```
<?php
// Script save as baou_start_session.php
    session_start();// Start the session
?>
<html>
    <body>
        <?php
            // Set session variables
                $_SESSION["favcolor"] = "green";
                $_SESSION["favanimal"] = "cat";
                echo "Session variables are set.";
        ?>
    </body>
</html>
```

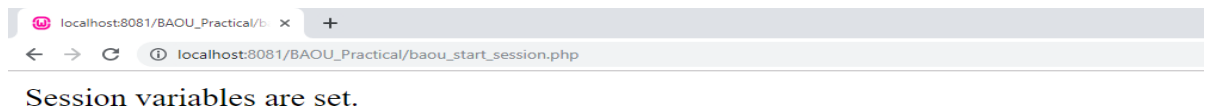


Figure 58 output of baou_start_session.php

Get PHP Session Variable Values

Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php").

Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (`session_start()`).

Also notice that all session variable values are stored in the global `$_SESSION` variable:

Get PHP Session Variable

```
<?php
    session_start();
?>
<html>
```

```

<body>
  <?php
    // Echo session variables that were set on previous page
    echo "Favorite color is " . $_SESSION["favcolor"] . "<br>";
    echo "Favorite animal is " . $_SESSION["favanimal"] . ".";

  ?>
  /// or Print all session variable as an array
  <?php
    echo "Print all session variable as an array";
    print_r($_SESSION);

  ?>
</body>
</html>

```

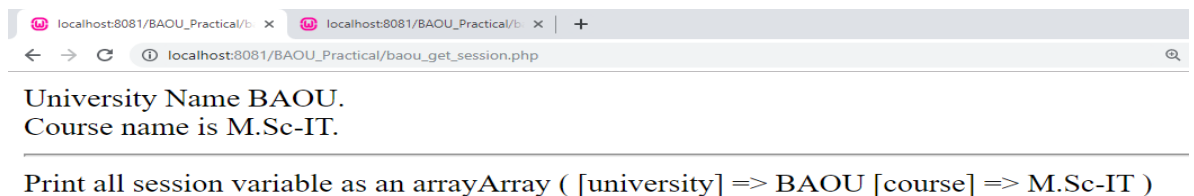


Figure 59 output of baou_get_session.php (Individual & Array)

Destroying Session Variables:

The `session_destroy()` function is used to destroy the whole PHP session variables.

If you want to destroy only a session single item, you use the `unset()` function.

The code below illustrates how to use both methods.

```

<?php
  session_destroy(); //destroy entire session
?>
<?php
  unset($_SESSION['product']); //destroy product session item
?>

```

`session_destroy` removes all the session data including cookies associated with the session. `unset` only frees the individual session variables. Other data remains intact. Every PHP session has a timeout value — a duration, measured in seconds — which determines how long a session should remain alive in the absence of any user activity. You can adjust this timeout duration by changing the value of `session.gc_maxlifetime` variable in the PHP configuration file (`php.ini`).

Unit 2: File and directory access in PHP

2

Unit Structure

2.1 Learning Objectives

2.2 Introduction

2.3 File Handling Functions

2.4 Directory Handling Functions

2.5 Suggested Answer for Check your Progress

2.1 LEARNING OBJECTIVES

After completion of this unit students will be able to

- Read data from file.
- Write data to file.
- Read and Write data from random position.
- Handle directories

2.2 INTRODUCTION

There are occasions where we need to interact with the file and directory in web applications. You might have some data available in text file or csv file and you need to read that data in your web application to have some analysis on it. These requires to open file and perform some reading and writing operations on it. PHP provides different functions to interact with file and directories. PHP has several functions for creating, reading, writing, updating and uploading files. In this unit we will learn all these functions with examples.

2.3 FILE HANDLING

To perform any reading or writing operation on the file we need to open it first. PHP provides `fopen()` to open the file.

fopen(filename,mode) – is used to open the file for different operation. Filename is the name of file that we want to open and mode decides which operation we want to perform on the file. Different modes are:

- r - open a file for reading only. File pointer starts from the beginning of the file.
- w - open a file for write only. Deletes the contents of the file if already exist or creates a new file if it doesn't exist. File pointer starts at the beginning of the file.
- a - Open a file for write only. The existing data in file is not deleted if file already exist. File pointer starts at the end of the file. Creates a new file if the file doesn't exist.
- x - creates a new file for write only. Returns FALSE and an error if file already exists.
- r+ - open a file for read/write. File pointer starts at the beginning of the file
- w+ - open a file for read/write. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
- a+ - open a file for read/write. The existing data in file is not deleted. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
- x+ - creates a new file for read/write. Returns FALSE and an error if file already exists.

fclose(filepointer) – This function is used to close the opened file.

```
Example: $fp=fopen("Myfile.txt","r");  
         fclose($fp)
```

The above example opens the file Myfile.txt in read mode. If the file is available in the same directory, then there is no need to mention full path, else we need to write full path of the file. \$fp is a file pointer which holds the file which is opened. The next statement closes the file which is pointed by \$fp.

fread(filepointer, length) - Read the contents from the file and stops when it reaches to end of file or at specified length which comes first.

Example:

```
$fp=fopen("Myfile.txt","r");
```

```
echo fread($fp,filesize("Myfile.txt"));
```

it will read all the contents from Myfile.txt because instead of specifying length of bytes to be read we have used filesize function which will return the size of Myfile.txt in bytes. If we write 5 as second argument then it will read first 5 bytes of data from Myfile.txt file.

fgetc(filepointer) – it is used to read single character from the file.

fgets(filepointer) – it is used to read single line from the file.

fputs(filepointer,string) – it is used to write single line into the file.

fwrite(filepointer,string,length) - fwrite is used to write string into file. length is optional. if length is specified then that number of bytes will be written into the file. If reaches to the end of file then stops writing. It returns the number of bytes written into the file.

Example:

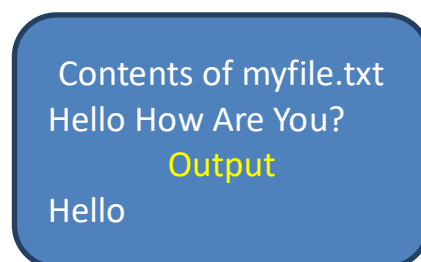
```
$fp1=fopen("write.txt","w");  
fwrite($fp1,"I am learning PHP");
```

The above statements will write I am learning PHP text into write.txt file. If write.txt file has some contents already written into it then they are erased because w mode starts writing from the beginning of the file. If you want to preserve the old contents and want to add the new contents at the end of the file then instead of w mode use a (append) mode.

file_exists(file) – it is used to check that specified file exists or not. Return true on success and false on failure.

Example:

```
if(file_exists("myfile.txt"))  
{  
$fp=fopen("myfile.txt","r");  
echo fread($fp,5);  
}  
else
```



Contents of myfile.txt
Hello How Are You?
Output
Hello

```
echo "<br> file not exist";
```

Above code first checks that the file myfile.txt exists in the current directory or not? If it exists then it will read first 5 characters of that file and print it. If the file is not available in the current directory, then the code will print the message that file not exist.

Example of fgets and fputs

```
$fp=fopen("read.txt","r");
```

```
$fp1=fopen("write.txt","a");
```

```
while(!feof($fp))//feof checks for end of file. Returns true if file pointer reaches to end of file.
```

```
{  
$read=fgets($fp);  
fputs($fp1,$read);  
}  
fclose($fp);
```

Content of read.txt

PHP is Easy to use.
PHP is Easy to Learn.

Contents of write.txt

PHP is Easy to use.
PHP is Easy to Learn.

Above code reads the contents from read.txt file and writes it into write.txt file.

Reading the data from Random position

In the above examples we read or write data from beginning or end but sometimes we need to read or write data from random position of the file. To read or write data from random position, we need to set the file pointer to desire position. In PHP we have fseek() function to set file pointer at random position.

fseek(file , offset , whence) - fseek function moves the file pointer from its current position to a new position, forward or backward, specified by the number of bytes. It is used for random location reading and writing in file. File position starts with 0.

- offset specifies the new position of the pointer. It is measured in bytes.
- whence can be SEEK_SET, SEEK_CUR, SEEK_END
- SEEK_SET - It sets position equal to offset.
- SEEK_CUR - It sets position to current location plus offset.

- SEEK_END - It sets position to EOF plus offset. To move to a position before EOF, the offset must be a negative value.

Example 1:

```
$fp=fopen("random.txt","r");
fseek($fp,0); // Sets the pointer to beginning of the file.
fseek($fp,7); // Sets the file pointer to 7th position.
$str=fread($fp,4); // read next 4 bytes from 7th position and assign them to $str
variable.
echo "<br>".$str;
```

Content of random.txt
Hello. This is sample file to test fseek function.
Output
This

Example 2:

```
$fp=fopen("random.txt","r");
fseek($fp,7,SEEK_SET); // Sets the file pointer to 7th position and read from there.
$str=fgets($fp);
echo "<br>Seek Set: ".$str;
fseek($fp,7);
fseek($fp,8,SEEK_CUR); // Sets the file pointer to 7th position and then move 8
position in forward direction and read from the new position which is 15.
$str=fgets($fp);
echo "<br>Seek Cur: ".$str;

fseek($fp,0);
fseek($fp,-10,SEEK_END); // Set
$str=fgets($fp);
echo "<br>Seek end: ".$str;
```

Content of random.txt
Hello. This is sample file to test fseek function.
Output
Seek Set: This is sample file to test fseek function.
Seek Cur: sample file to test fseek function.
Seek End: function

Check your Progress – 1:

1. What is the difference between 'w' mode and 'a' mode in fopen()?

.....
.....
.....
.....

2. What are various file reading and writing functions available in PHP?

.....
.....
.....
.....

2.4 DIRECTORY HANDLING

The directory functions in PHP allow you to retrieve information about directories and to manipulate them. There are different functions available in PHP to work with directories. In this section we are going to discuss important directory handling functions of PHP.

getcwd() - this function is used to get the current working directory. It returns the current working directory on success and returns false in case of failure.

```
<?php  
echo getcwd();  
?>
```

Output: C:\wamp64\www\PHP Programs

chdir(directory) -The chdir() function is used to change the current directory. Directory is the name of directory that we need to change.

```
<?php  
echo getcwd();  
chdir("Images");  
echo "<br>";  
echo getcwd();  
?>
```

Output:

C:\Xampp64\www\PHP Programs

C:\Xamp64\www\PHP Programs\Images

opendir() – it opens the directory handle. You need to specify the path of the directory which you want to open.

closedir(*dir*) –it is used to close the directory handle resource specified by *dir*.

readdir(*\$dir*) - It returns the name of the next file from the opened directory. *\$dir* is the directory handle resource opened with `opendir()`. It returns the name of the file on success or it returns FALSE on failure.

Example:

```
<?php
$d = opendir("Images");
$i=1;
while($file = readdir($d))
{
    echo "<br>". "file$i:". $file;
    $i++;
}
closedir($d);
?>
```

Output:

```
file1:..
file2:...
file3:product1.jpg
file4:product2.jpg
file5:product3.jpg
```

Above code will open Images directory and then read files one by one from it.

Check your Progress – 2:

1. How can you count total files of particular directory in PHP?

.....

.....

.....

.....

2.5 Suggested Answers for Check your Progress

3

Check your progress – 1:

1. w - open a file for write only. Deletes the contents of the file if already exist or creates a new file if it doesn't exist. File pointer starts at the beginning of the file.
a - Open a file for write only. The existing data in file is not deleted if file already exist. File pointer starts at the end of the file. Creates a new file if the file doesn't exist.
2. Various file reading and writing functions available in PHP:
fread(), fwrite(), fgetc(), fgets(), fputs() and fseek().

Check your progress – 2:

1. Following code will count total number of files from Images directory.

```
<?php
$d = opendir("Images");
$i=0;
while($file = readdir($d))
{
    echo "<br>". "file:". $file;
    $i++;
}
echo "total files available in Image folder are:". $i;
closedir($d);
?>
```

Unit 3: Working and formatting with strings

Unit Structure

- 3.1 Learning Objectives
- 3.2 Introduction

3.3 Difference between Single and Double Quoted Strings

3.4 String Handling Functions

3.5 Suggested Answer for Check your Progress

3.1 LEARNING OBJECTIVES

After completion of this unit students will be able to

- Understand the difference between single and double quoted strings.
- Handle the string data in PHP.
- Format the string data as per requirements.

3.2 Introduction

In web applications, there are certain occasions where we need to store text data for further processing such as to store name of person, city, address, name of product, description etc. to store all these types of information, strings are used.

PHP string is a sequence of characters which is used to store and manipulate text. Normally PHP string is represented using single and double quoted string.

Example:

```
$name="Rahul Kumar";
```

Or

```
$name='Rahul Kumar';
```

Both is valid representation.

3.3 Difference between Single and Double Quoted Strings

However, there is a difference between single and double quoted strings in PHP where we are combining string with variable to print any message. Consider the following example:

```
$name="Rahul";
```

```
Echo "My name is $name";
```

Output: My name is Rahul

```
$name="Rahul";  
Echo 'My name is $name';  
Output: My name is $name
```

In double quoted string the variable is replaced with its value but it is not replaced in single quoted string.

3.4 String Handling Functions

Sometimes we need to perform certain operations on the string values for better results such as converting the name of person into lower or upper case, concatenating two strings, replacing certain parts of string, counting number of characters and words of the string etc. All these tasks require certain string handling functions. Let's us discuss some of the important string handling functions in brief.

strlen() – returns the length (number of characters) of the string.

```
$name="Hello";  
$len=strlen($name);  
echo "Length of the string is $len";
```

Output
Length of the string is 5

str_word_count() – It counts the number of word of the given string.

```
$name="Hello How Are You";  
$count=str_word_count($name);  
echo "Total words of the string is $count";
```

Output
Total words of the string
is 4

strtolower() – It converts the string in to lowercase. It Returns the lowercased string. Original string will remain unchanged.

```
$name="HELLO";  
echo strtolower($name);
```

Output
hello

strtoupper() – Converts the string in to upper case. This function is useful where you want to store all the names of persons in upper case. Consider the scenario where you have designed customer registration page where user is supposed to enter his or her name in upper case but some of the users are still entering their names in lower case. Here you can convert the names in to upper case with the use of **strtoupper()** before storing them in to your data base.

```
$name="hello";  
echo strtoupper($name);
```

Output
HELLO

ucfirst() – Converts the first character of the string in to uppercase.

```
$name="hello";  
echo ucfirst($name);
```

Output
Hello

lcfirst() – It converts the first character of the string in to lower case.

```
$name="Hello";  
echo lcfirst($name);
```

Output
hello

ucwords() – Converts the first character of the each word of string in to uppercase.

```
$name="my name is rahul";  
echo ucwords($name);
```

Output
My Name Is Rahul

strrev() – Reverse the string.

```
$name="rahul";  
echo strrev($name);
```

Output
luhar

strcmp() – compares two strings. Returns 0 if both the strings are same. It is case sensitive (rahul and Rahul are not same)

```
$name1="rahul";  
$name2="rahul";  
if(strcmp($name1,$name2)==0)  
echo "<br> Both are same ";  
else  
echo "<br> both the strings are different ";
```

Output
Both are same

strcasecmp() – compares two strings. Returns 0 if both the strings are same. It is case insensitive (rahul and Rahul are same)

```
$name1="rahul";  
$name2="Rahul";
```

Output
Both are same

```
if(strcasecmp($name1,$name2)==0)
echo "<br> Both are same";
else
echo "<br> Both the strings are different ";
```

strstr() – finds the first occurrence of the string inside the string and return rest of the string. It is case sensitive.

```
$name="My name is Rahul";
$find="is";
echo strstr($name,$find);
```

Output
is Rahul

In the above example we need to find 'is' from the string 'My name is Rahul'. It searches 'is' from the string and when finds it returns the rest of the string starting from the search string 'is'.

stristr() – finds the first occurrence of the string inside the string and return rest of the string. It is case insensitive.

```
$name="My name is Rahul";
$find="Is";
echo stristr($name,$find);
```

Output
is Rahul

ltrim() – remove space or characters from the left side of the string and returns new string.

```
$name=" My name is Rahul";
echo ltrim($name);
```

Output
My name is Rahul

```
$str = "Hello World!";
echo ltrim($str,"Hello");
```

Output
World!

Here in the first example ltrim has removed white spaces from the left side of the string and in second example it has removed the word hello from the string.

rtrim() – remove space or characters from the right side of the string.

```
$name="My name is Rahul ";
echo rtrim($name);
```

Output
My name is Rahul

```
$str = "Hello World!";  
echo rtrim($str,"World!");
```

Output
Hello

trim() – remove space or characters from both the side of the string

```
$name=" My name is Rahul ";  
echo trim($name);
```

Output
My name is Rahul

str_replace() – Replace the characters from the string.

Parameters:

- Find – the value to find. [Required Argument]
- Replace – the value to replace with the find value. [Required Argument]
- String – the string to be searched. [Required Argument]
- Count – variable that counts the number of replacements. [Optional Argument]

```
$url="http://www.mca.baou.ac.in";  
echo str_replace("mca","mcsit",$url,$count);  
echo $count
```

Output : http://www.mcsit.baou.ac.in
count:->1

Here mca is replaced with mcsit and total one replacement is done so count equals 1.

substr_replace() – It replaces a part of a string with another string.

Parameters:

- string – the string to check. [Required Argument]
- Replacement – specifies the string to insert. [Required Argument]
- Start – Specifies where to start replacing in the string. [Required Argument]
- length – Specifies how many characters should be replaced. [Optional Argument]

```
$string="ramnagar";  
echo "<br>".substr_replace($string,"pur",3);
```

Output :rampur

Here it places the pointer to the 3rd position and replace all the characters from 3rd position to the end because we have not specified the length.

```
$string="Uttarpradesh";  
echo "<br>".substr_replace($string,"Madhya",0,5);
```

Output :Madhyapradesh

Here we have specified the length so it places the pointer to the 0th position and replace first five characters only.

substr() – it returns the part of the string. If we want to return any part of the string with its position then we can use this function,

Parameters:

- string – the string to check. [Required Argument]
- Start – Specifies where to start in the string. [Required Argument]
- length – Specifies the length of the returned string. [Optional Argument]

```
$name="My name is Rahul";  
echo "<br> substr: ".substr($name,11,5);
```

Output : Rahul

So, these are some of the important string handling functions used to perform actions on the string data in PHP.

Check your progress – 1:

1. How can you convert the given string into upper, lower and initcap?

.....
.....
.....

3.5 Suggested Answer for Check Your Progress

Check your progress – 1:

To convert the given string into upper, lower and initcap following functions are used.

strtolower() – Converts the string in to lowercase. It Returns the lowercased string. Original string will remain unchanged.

strtoupper() – Converts the string in to upper case. This function is useful where you want to store all the names of persons in upper case.

ucwords() – Converts the first character of the each word of string in to uppercase.

Unit 4: Handling Databases in PHP

4

Unit Structure

4.1 Learning Objectives

4.2 Introduction

4.3 Connecting PHP with MySQL

4.4 Database Manipulation Operations – Insert, Update, Delete

4.5 Retrieving Records from Database

4.6 Suggested Answer for Check your Progress

4.1 LEARNING OBJECTIVES

After completion of this unit students will be able to

- Connect PHP with MySQL.
- Perform database manipulation operations like Insert, Update and Delete in PHP.
- Retrieve data from tables and represent them in readable format.

4.2 Introduction

Data is any important entity for any web site and hence it should be properly organised and retrieved when needed. In web applications we need to store various kinds of data such as if we are building online shopping site then we may need to store data such as customer details, supplier details, products, orders, invoice details etc. For these we have to design a form to collect data from various stake holders of the web site and then to store these data in the proper table. To do so we have to connect PHP with database application. In this unit we are going to learn that how can we connect PHP with MySQL and how can we perform various database operations on it.

4.3 Connecting PHP with MySQL

The most common and popular database used with most of the PHP web applications is MySQL database because the connectivity of PHP with MySQL database is very easy.

To connect PHP 5 and later versions with database we can use:

- ✓ MySQLi extension ('i' stands for improved)
- ✓ PDO (PHP Data Objects)

Earlier versions use MySQL extension which was deprecated in 2012.

MySQLi Extension: To connect PHP with MySQL database, the MySQLi extension is used. Following function is used to connect PHP with MySQL.

```
mysqli_connect($servername,$username,$password,$databasename)
```

- \$servername – Name or IP address of server/host.
- \$username – MySQL username
- \$password – MySQL Password
- \$databasename – Database name to be used.

For example

```
$con=mysqli_connect("localhost","root","","student");
```

The above function establish connection with MySQL database called student. In the server name we have mentioned localhost because our database is located in the local server. Here root is default username of MySQL database and default password is blank. If you have set another user name and password then you need to mention that username and password. Student is the name of database that we want to connect with. We need to create student database first in MySQL. The function returns the object representing the connection to MySQL server. It returns False on failure.

If there is no error means we have established a successful connection with MySQL database called student.

Example:

```
$con = mysqli_connect("localhost","root","","student");
if ($con)
{
echo("Database connection successful);
}
```

```

else
{
die("Can't Connect. Connection
Error:".mysql_connect_errno()."".mysql_connect_error());
}

```

The above code will establish the connection of PHP with MySQL student database. If there is no error than it prints the message Database connection successful else in case of failure it shows error with error number and description with the use of functions like mysql_connect_errno() and mysql_connect_error().

Check Your Progress – 1:

1. Which function is used to connect PHP with MySQL database? What are the parameters of it?

.....

.....

.....

.....

.....

4.4 Database Manipulation Operations – Insert, Update, Delete

To perform any database manipulation operations like inserting a record, updating the record or deletion of record requires to execute respective queries. mysql_query() function is used to execute any SQL query in PHP. We can use it as:

```
$result=mysql_query($con,$query);
```

The above function is used to perform SQL query on the mentioned database table. \$con is the connection object that we set using mysql_connect(). \$query is a SQL query that we want to execute on the database table. On successful select queries it returns mysql_result object. For other queries it returns True on success and False on failure.

PHP-MySQL connection example – Inserting record in table:

Following example shows how we can insert record in student_master table located under student database. The structure of the student_master table is:

student_master	
Field name	Field Type

sno	Integer
name	Varchar
city	Varchar
pin	Integer

You need to create student database in MySQL and then create student_master table under it. To create database and table we can use phpMyAdmin, a web interface to handle MySQL. Just type localhost/phpmyadmin in the browser address bar to open it. You can create database and tables over there.

```

$con = mysqli_connect("localhost","root","","student");
if (!$con)
{
die("Can't Connect. Connection Error:".mysqli_connect_errno()."
".mysqli_connect_error());
}
$sno=1001;
$name="Ram";
$city="Ahmedabad";
$pin=383240;

$query = "insert into student_master values($sno,$name,$city,$pin)";
$result = mysqli_query($con, $query);
if($result)
{
echo "Record inserted successfully";
}
else
{
echo "Record not inserted <br>";
echo "error no is:".mysqli_errno($con)."Error is:".mysqli_error($con);
}

mysqli_close($con);

```

The above code inserts one record in the student_master table. Here we have directly specified the values of the fields. We can also design input form to enter all those details. Let's see how we can do that. We have created two files – one is input design form named as insert.html and other is PHP file where connection code is written which is named as insertrecord.php

Insert.html

```

<html>
<body>
<form method="post" action="insertrecord.php">
<table border="1" align="center">
<tr><td colspan=2 align="center">Record Insert</td></tr>
<tr>
<td>Enter Your Eno </td>
<td><input type="text" name="eno"></td>
</tr>
<tr>
<td>Enter Your Name </td>
<td><input type="text" name="name"></td></tr>
<tr>
<td>Enter Your City </td>
<td><select name="city">
<option>Mehsana</option>
<option>Surat</option>
<option>Ahmedabad</option>
<option>Rajkot</option>
<option>Bharuch</option>
</select></td></tr>
<tr>
<td>Enter Your Pin </td>
<td><input type="number" name="pin"></td></tr>
<tr>
<td colspan=2 align="center"><input type="submit" value="Insert"></td></tr>
</table>
</body>
</html>

```

Insertrecord.php:

```

<?php
$con = mysqli_connect("localhost","root","","student");

if (!$con)
{
die("Can't Connect. Connection Error:".mysqli_connect_errno()."
".mysqli_connect_error());
}

$sno=$_POST['eno'];
$name=$_POST['name'];
$city=$_POST['city'];

```

```

$pin=$_POST['pin'];

$query = "insert into student_master values($sno,'$name','$city',$pin)";
$result = mysqli_query($con, $query);

if($result)
{
    echo "Record inserted successfully";
}
else
{
    echo "Record not inserted <br>";
    echo "error no is:".mysqli_errno($con)."Error is:".mysqli_error($con);
}
mysqli_close($con);

```

insert.html file is executed first. After adding fields press insert button that redirect us to insertrecord.php file where we have collected all the entered information with \$_POST[] super global array. mysqli_query() executes insert query and add record in student_master table. We can cross verify by opening student_master table in MySQL.

Delete operation

Sometimes we need to delete the records from the MySQL table. There are multiple reasons of delete such as duplicate record, unwanted entry, wrong entry etc. To delete the record we need to execute delete query in mysqli_query(). Following example shows the delete operation.

Delete.html

```

<html>
<body>
<form method="post" action="deleterecord.php">
<table border="1" align="center">
<tr><td colspan=2 align="center">Record Delete</td></tr>
<tr>
<td>Enter Eno you want to delete </td>
<td><input type="text" name="eno"></td>
</tr>
<tr>
<td colspan=2 align="center"><input type="submit" value="Delete"></td></tr>
</table>

```



```
</body>
</html>
```

This file is created to ask for the student enrolment number that we want to delete from student_master table.

deleterecord.php

```
<?php
$con = mysqli_connect("localhost","root","","student");

if (!$con)
{
die("Can't Connect. Connection Error:".mysqli_connect_errno()."
".mysqli_connect_error());
}

$eno=$_POST['eno'];

$query = "delete from student_master where sno=$eno";
$result = mysqli_query($con, $query);

if(mysqli_affected_rows($con) >= 1)
{
echo "Record deleted successfully";
}
else
{
echo "Record not deleted <br>";
echo "error no is:".mysqli_errno($con)."Error is:".mysqli_error($con);
}
mysqli_close($con);
```

Here to confirm that the record is deleted or not we have used `mysqli_affected_rows()` function. The `mysqli_affected_rows()` function returns the number of affected rows in the table by the SELECT, INSERT, UPDATE or DELETE queries. If it returns one means one row is affected by the executed query. Here in the given example if the return value of the function is one or greater equal one means at least one record is deleted.

Update Operation

Like insert and delete records, update record is also an important operation required to be performed on the database. We need to perform update operation in case -

Entries which are inserted wrong or the entries whose value is now changed (city of the student is changed now or mobile number is changed). Following example shows how we can perform update operation on tables in PHP.

Update.html

```
<html>
<body>
<form method="post" action="update.php">
Enter enrollment no to update: <input type="text" name="num1">
<input type="submit" value="Search">
</body>
</html>
```

update.php

```
<form name="fname" method="post" action="updaterecord.php">
<?php

$link = mysqli_connect("localhost","root","","student");

if (!$link)
{
die('Connect Error: ' . mysqli_connect_errno().mysqli_connect_error());
}
$no=$_POST['num1'];
$query = "select * from student_master where sno='$no'";

$result = mysqli_query($link, $query);

if(!$result)

{
echo "error no is:".mysqli_errno($link)."Error is:".mysqli_error($link);
}
else
{
while($row = mysqli_fetch_row($result))
{
?>
<table border="1">
```

```

<tr><td>Sno</td><td><input type="text" name="num" value="<?php echo $row[0];
?>">
</td></tr>
<tr><td>Sname</td><td><input type="text" name="sname" value="<?php echo
$row[1]; ?>">
</td></tr>
<tr><td>Scity</td><td><input type="text" name="scity" value="<?php echo $row[2];
?>">
</td></tr>
<tr><td>Pin</td><td><input type="text" name="pin" value="<?php echo $row[3];
?>"></td></tr>
<tr><td colspan="2" align="center"><input type="submit" value="Update"></td></tr>
<?php
}
}
mysqli_close($link);
?>

```

This file is created to show the existing records of the student to the user before update. Here we have fetched the records from the database and put these values in the textboxes in edit mode so that user can change them. To retrieve records from the table we have used `mysqli_fetch_row()` method of data fetching. Kindly refer next section to understand various data fetching methods. In this file user sees the existing records, update them and then press update button which redirects it to the `updaterecord.php` file.

updaterecord.php

```

<?php

$link = mysqli_connect("localhost","root","","student");

if (!$link)
{
die('Connect Error: ' . mysqli_connect_errno().mysqli_connect_error());
}
$no=$_POST['num'];
$name=$_POST['sname'];
$city=$_POST['scity'];
$pin=$_POST['pin'];

```

```

$query = "update student_master set sno=$no, name='$name', city='$city', pin=$pin
where sno='$no'";

$result = mysqli_query($link, $query);

if(!$result)

{
echo "error no is:".mysqli_errno($link)."Error is:".mysqli_error($link);
}
else
{
echo "record updated successfully";
}

mysqli_close($link);

?>

```

Check Your Progress – 2:

1. What is the use of mysqli_query()?

.....

.....

.....

.....

.....

2. Write SQL query to update the city of the student from Gandhinagar to Ahmedabad whose sno is 1002.

.....

.....

.....

4.5 Retrieving Records from Database

After inserting and updating the records, retrieving of records from database is equally important. In web applications we need to retrieve the data stored in database tables for various purposes such as if we consider online shopping site then we require to fetch the data from tables to generate customer bill, to view products, to view suppliers and customers, to find out the monthly sales of a product, to search particular product, to display shopping cart etc.

PHP supports different data fetching methods which are used to fetch records from tables. These methods are:

- `mysqli_fetch_row()`
- `mysqli_fetch_assoc()`
- `mysqli_fetch_array()`
- `mysqli_fetch_object()`

In this section we are going to learn these methods in detail.

`mysqli_fetch_row(result)`—it fetches one row from a result-set and returns it as an indexed array. Returns NULL in case of no rows in result set.

Result is an identifier returned by `mysqli_query()`. Let's take an example where we need to search student by its enrolment number.

Example:

Searchbox.php

```
<html>
<body>
<form method="post" action="search.php">
Enter enrolment no to search: <input type="text" name="no">
<input type="submit" value="Search">
</body>
</html>
```

Above code will ask the user to enter the enrolment number of student whom we are searching the details. By clicking the search button, it will redirect to search.php file.

search.php

```
$link = mysqli_connect("localhost","root","","student");
$no=$_POST['no'];
$query = "select * from student_master where sno='$no'";
$result = mysqli_query($link, $query);
if(mysqli_affected_rows($link) == 0)
{
echo "Record not found";
}
else
{
```

```

while($row = mysqli_fetch_row($result)) // here $row becomes any index array and
size of array = total fields in table.
{
echo "Sno: ".$row[0]; // retrieve first field value which is sno
echo "Sname: ".$row[1]; // retrieve second field value which is student name.
echo "City: ".$row[2]; // retrieve third field value which is student city.
    echo "Pin: ".$row[3]; // retrieve forth field value which is pincode.
}
}

```

Above script will perform select query in student_master table to retrieve the details of the student whose enrolment number is entered by user in searchbox.php file. mysqli_fetch_row() method will fetch the details in the form of indexed array and hence \$row now becomes an index array whose size is equivalent to number of fields in the table. Here in student_master table we have total 4 fields (sno, name, city, pin) so the \$row array size is 4. Though it is an index array its values are retrieved by integer index like \$row[0], \$row[1], \$row[2], \$row[3]. We put mysqli_fetch_row() in while loop so if there multiple records are retrieved then the loop will continue and it gets the details of next records.

mysqli_fetch_assoc(result)—it fetches a result row as an associative array. Returns NULL in case of no rows in result set.Result is an identifier returned by mysqli_query().

The difference between mysqli_fetch_row and mysqli_fetch_assoc is the return type. First one returns index array while the second one returns associative array.

Lets take the same example and retrieve the student record using mysqli_fetch_assoc(). Searchbox.php will remain same. We need to change the code of search.php file.

search.php

```

$link = mysqli_connect("localhost","root","","student");
$no=$_POST['no'];
$query = "select * from student_master where sno='$no'";
$result = mysqli_query($link, $query);
if(mysqli_affected_rows($link) == 0)
{
echo "Record not found";
}
else
{

```

```

while($row = mysqli_fetch_assoc($result))
{
echo "Sno: ".$row["sno"];
echo "Sname: ".$row["name"];
echo "City: ".$row["city"];
echo "Pin: ".$row["pin"];
}
}

```

if you observe the difference between these two methods then it states that in previous method, we use integer index to retrieve records and, in this method, we use table field names as array index because here the method returns associative array and in associative array the values are retrieved by its key. Here fields names are keys for the array \$row.

mysqli_fetch_array(result,returntype)—it fetches a result row as an associative array, a numeric array, or both. Returns NULL in case of no rows in result set.Result is an identifier returned by mysqli_query().

returntype is used to select the return type of array. Possible values are

- MYSQLI_NUM – if you want to return only index array.
- MYSQLI_ASSOC – if you want to return associative array.
- MYSQLI_BOTH (this is default) – Both type of array will be returned.

mysqli_fetch_array() returns both index as well as associative array by default and hence you can use integer index or key to retrieve values. Let's continue the same example using mysqli_fetch_array().

search.php

```

$link = mysqli_connect("localhost","root","","student");
$no=$_POST['no'];
$query = "select * from student_master where sno='$no'";
$result = mysqli_query($link, $query);
if(mysqli_affected_rows($link) == 0)
{
echo "Record not found";
}
else
{
while($row = mysqli_fetch_array($result))
{
echo "Sno: ".$row[0]; // retrieve student number by index array
echo "Sname: ".$row[1]; // retrieve student name by index array
echo "City: ".$row["city"]; // retrieve student city by the key city.
}
}

```

```

echo "Pin:". $row["pin"]; // retrieve student pin code by the key pin.
}
}

```

mysqli_fetch_object(result)—it returns the current row of a result-set, as an object. Returns NULL in case of no rows in result set. Result is an identifier returned by mysqli_query(). Apart from this method, all three methods that we learn so far return either index array, associative array or both but mysqli_fetch_object() returns current row as an object. Fields names are used along with object to retrieve data from tables.

Lets now retrieve data from student_master table using mysqli_fetch_object()

Example:

Search.php

```

$link = mysqli_connect("localhost","root","","student");
$no=$_POST['no'];
$query = "select * from student_master where sno='$no'";
$result = mysqli_query($link, $query);
if(mysqli_affected_rows($link) == 0)
{
echo "Record not found";
}
else
{
while($row = mysqli_fetch_object($result))
{
echo "Sno:". $row->sno;
echo "Sname:". $row->name;
echo "City:". $row->city;
echo "City:". $row->pin;
}
}
}

```

in the above code we have used mysqli_fetch_object() which returns object and so \$row is now object and to retrieve records we use table field names along with \$row object.

In this unit we have discussed all the data fetching methods with example. You can use any of these methods according to the need of an application. You can use these methods while you are retrieving records from database.

Check Your Progress – 3:

1. What is the difference between `mysqli_fetch_row()` and `mysqli_fetch_assoc()`?

.....
.....
.....
.....

4.6 Suggested Answer for Check your Progress:

Check your Progress 1:

1. `mysqli_connect()` is used to connect PHP with MySQL

`mysqli_connect($servername,$username,$password,$databasename)`

it takes following parameters

- `$servername` – Name or IP address of server/host.
- `$username` – MySQL username
- `$password` – MySQL Password
- `$databasename` – Database name to be used.

Check your Progress 2:

1. `mysqli_query()` function is used to execute any SQL query in PHP. We can use it as:

`$result=mysqli_query($con,$query);`

2. `update student_master set city="Ahmedabad" where sno=1002`

Check your Progress 3:

1. `mysqli_fetch_row()` and `mysqli_fetch_assoc()` methods are used to retrieve records from database. First one returns index array whereas the second one returns associative array.