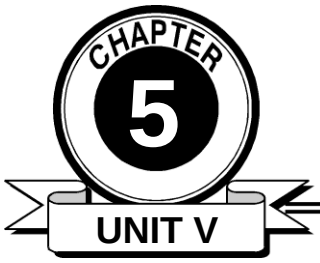


Database Operations

Syllabus

5.1 Introduction to MySQL, Create a Database.

5.2 Connecting to a MySQL database: MySQL database server from PHP.

5.3 Database Operations: Insert data, Retrieving the Query Result.

5.4 Update and Delete operations on table data

5.1 Introduction to MySQL

- **MySQL** is used to manage stored data and is an open source Database Management Software (DBMS) or Relational Database Management System (RDBMS).
- Its name **My-S-Q-L** is often pronounced as **My Sequel** is a combination of "My", the name of co-founders Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language.
- MySQL was owned and sponsored by the Swedish company MySQL AB, which was acquired by Sun Microsystems.
- MySQL is one of the best RDBMS being used for developing various web-based software applications.
- MySQL is a main component of web application software's like XAMPP, LAMP and WAMP.
- MySQL is used by many database-driven web applications which includes Drupal, Joomla, phpBB, and WordPress.
- MySQL is also used by many popular websites which includes Facebook, Flickr, MediaWiki, Twitter and YouTube.
- It is designed to allow simple requests from a database via commands such as
- SELECT name FROM student WHERE rollno=33;
- A MySQL database contains one or more tables, each of which contains records orrows. Within these rows are various columns or fields that contain the data itself.
- Each row in the table is the same as a row in a MySQL table, and each element within a row is the same as a MySQL field.
- PHP and MySQL are server side technologies, both are used on server side so the combination of these is preferred to developed cloud based application.
- MySQL operates using client/server architecture in which the server runs on the machine containing the databases and clients connect to the server over a network. The server (MySQL server) listens for client requests coming in over the network and accesses database contents according to those requests and provides that to the clients.
- Clients are programs that connect to the database server and issue queries in a pre-specified format. MySQL is compatible with the standards based SQL (SQL stands for Structured Query Language) language. The client program may contact the server programmatically (meaning a program call the server during execution) or manually.

- Typically, mySQL is supported on Windows XP, Windows Server 2003, Red Hat Fedora Linux, and Debian Linux, and others. As with any other client/server application, MySQL is a multi-user database system, meaning several users can access the database simultaneously.

5.1.1 Features of MySQL

Q. Explain features of MySQL.

- **Easy to use:** MySQL is easy to use. With the basic knowledge of SQL, you can build and interact with MySQL.MySQL has typically been configured, monitored, and managed from the command line. However, several MySQL are available with graphical interfaces.
- **Free to download:** MySQL is free to use and you can download it from MySQL official website.
- **Ease of Management:** The software very easily gets downloaded and also uses an event scheduler to schedule the tasks automatically.
- **Client/ Server Architecture:** MySQL follows a client /server architecture. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they query data, save changes, etc.
- **Compatible on many operating systems:** MySQL is compatible to run on many operating systems, like Novell NetWare, Windows OS, Linux OS, many varieties of UNIX, OS/2, FreeBSD*, and others. MySQL also provides a facility that the clients can run on the same computer as the server or on another computer (communication via a local network or the Internet).
- **Robust Transactional Support:** Holds the ACID (Atomicity, Consistency, Isolation, Durability) property, and also allows distributed multi-version support.
- **Comprehensive Application Development :** MySQL has plugin libraries to embed the database into any application. It also supports stored procedures, triggers, functions, views and many more for application development.
- **High Performance:** MySQL is faster, more reliable and cheaper because of its unique storage engine architecture.
- **No License Fee:**MySQL is available free of cost. MySQL is a "Open Source" database. MySQL is part of LAMP (Linux, Apache, MySQL, PHP / Perl / Python) environemtn, a fast growing open source enterprise software stack. This reduces licensing costs and hardware expenditures.
- **Security:** MySQL supports powerful mechanisms to ensure that only authorized users have access to the databases. MySQL consist of a solid data security layer that protects sensitive data from intruders. Passwords are encrypted in MySQL.
- **High Availability:** MySQL can run high-speed master/slave replication configurations and it offers cluster servers.
- **Scalability:** MySQL can handle almost any amount of data, up to as much as 50 million rows or more. The default file size limit is about 4 GB. However, you can increase this number to a theoretical limit of 8 TB of data.
- **High Flexibility:** MySQL supports a large number of embedded applications which makes MySQL very flexible.
- **High Productivity:** MySQL uses Triggers, Stored procedures and views which allows the developer to give a higher productivity.
- **Speed:** The speed at which a server side program runs depends on the server hardware. MySQL runs very fast even server hardware is optimal. It supports clustered servers for demanding applications

5.1.2 MySQL Tools

- **A SQL server:** This is an engine which provides access to your databases.
- **Client programs for accessing the server:** A program allows you to enter queries directly and view results.
- **A client library for writing your own programs:** You can write your own programs into the client library using C.

5.1.3 Data Types in MySQL

- A Data Type specifies a particular type of data, like integer, floating points, Boolean etc.
- MySQL supports a number of SQL standard data types in various categories.
- It uses many different data types broken into mainly three categories: numeric, date and time and string types.

1. String Data Types : String Data types allow both fixed and variable length strings.

Data Type	Size	Description
CHAR(size)	Maximum 255 characters.	Fixed-length Character strings.
VARCHAR(size)	Maximum 255 characters.	Variable-length string.
TINYTEXT(size)	Maximum 255 characters.	Where size is the number of characters to store.
TEXT(size)	Maximum 65,535 characters.	Where size is the number of characters to store.
MEDIUMTEXT(size	Maximum 16,777,215 characters/Binary Large Object.	Where size is the number of characters to store.
LONGTEXT(size	Maximum 4GB or 4,294,967,295 characters/Binary Large Object	Where size is the number of characters to store.
BINARY(size)	Maximum 255 characters.	A fixed-length binary string
VARBINARY(size)	Maximum 255 characters.	A variable-length binary string

2. Numeric Data Types : Numeric Datatypes allow both signed and unsigned integers.

Data Type	Size	Description
BIT	Signed values range from -128 to 127. Unsigned values range from 0 to 255.	A bit field
TINYINT(*m*)	Signed values range from -128 to 127. Unsigned values range from 0 to 255. (1 byte)	Very small integer value.
SMALLINT(*m*)	Signed values range from -32768 to 32767. Unsigned values range from 0 to 65535. (2 byte)	Small integer value.
MEDIUMINT(*m*)	Signed values range from -8388608 to 8388607. Unsigned values range from 0 to 16777215. (3 byte)	Medium integer value.
INT(*m*)/ INTEGER(m)	Signed values range from -2147483648 to 2147483647. Unsigned values range from 0 to 4294967295. (4 bytes)	Standard integer value.
BIGINT(*m*)	Signed values range from -9223372036854775808 to 9223372036854775807. Unsigned values range from 0 to	Big integer value.

Data Type	Size	Description
	18446744073709551615. (8 bytes)	
DECIMAL(*m*,*d*)/ DEC(m.d)/ NUMERIC(m,d)/ FIXED(m,d)	**m** defaults to 10, if not specified. **d** defaults to 0, if not specified.	Unpacked fixed point number.
FLOAT(*m*,*d*)	Where **m** is the total digits and **d** is the number of digits after the decimal. (4 bytes)	Single precision floating point number.
DOUBLE(*m*,*d*)/ REAL(*m*,*d*)	Where **m** is the total digits and **d** is the number of digits after the decimal. (8 bytes)	Double precision floating point number.
FLOAT(p)	Where **p** is the precision	Floating point number.
BOOL/ BOOLEAN	Synonym for TINYINT(1)	Treated as a Boolean data type where a value of 0 is considered to be FALSE and any other value is considered to be TRUE.

3. Date and Time Data Types : MySQL provides types for date and time as well as the combination of date and time.

Data Type	Size	Description
DATE()	Values range from '1000-01-01' to '9999-12-31'. (3 bytes)	Displayed as 'yyyy-mm-dd'.
DATETIME()	Values range from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. (8 bytes)	Displayed as 'yyyy-mm-ddhh:mm:ss'.
TIMESTAMP(m)	Values range from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' TC. (4 bytes)	Displayed as 'YYYY-MM-DD HH:MM:SS'.
TIME()	Values range from '-838:59:59' to '838:59:59'. (3 bytes)	Displayed as 'HH:MM:SS'.
YEAR[(2|4)]	Year value as 2 digits or 4 digits. (1 byte)	Default is 4 digits.

Large Object Data Types (LOB) Data Types :

Data types	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 of 4gb or 4,294,967,295 characters.

5.1.4 Accessing MySQL via the Command Line

There are three main ways in which you can interact with MySQL:

1. Command line
2. Web interface such as phpMyAdmin,
3. Programming language like PHP

Starting the Command-Line Interface

1. Windows users

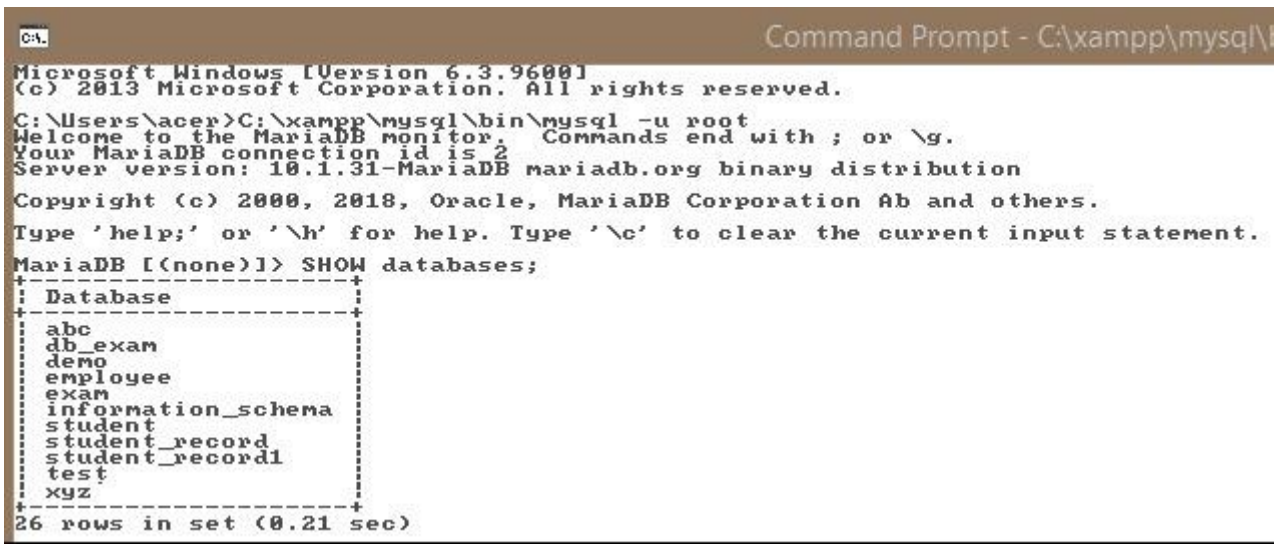
- After installing XAMPP, you will be able to access the MySQL executable from the following directory :

  C:\xampp\mysql\bin

- If anyone wants to install XAMPP in a place other than \xampp, you will need to use that directory instead.
- MySQL default user will be root and will not have had a password set.
- In order to enter MySQL's command-line interface, select Start→Run, enter CMD into the Run box, and press Return.
- This will call up a Windows command prompt. From there, enter one the following (making any appropriate changes as just discussed):

  C:\xampp\mysql\bin\mysql -u root

- Using this command you are asking MySQL to log you in as user root, without a password.
- After logging, MySQL you can start entering commands, for checking whether everything is working as it should be, enter the following command;
- SHOW databases;



2. Linux users

- On a system running a Unix-like operating system such as Linux, you will almost certainly already have PHP and MySQL installed and running

- First you should type the following to log into your MySQL system:

  mysql -u root -p

- This tells MySQL to log you in as the user root and to request your password. If youhave a password, enter it; otherwise, just press Return.

- Once you are logged in, type the following to test the program; you should see something like Figure in response:

  SHOW databases;

  (C/O – Figure not given Plz check)

5.1.5 MySQL Commands

- SQL commands and keywords are case-insensitive.
- It is recommended to use uppercase for all database operations.
- Table names are case-sensitive on Linux, but case-insensitive on Windows. So for portability purposes, you should always choose a case and stick to it. It is recommended to use lowercase for tables.
- The commonly used MySQL commands are listed in Table.

Table 5.1.1 : Common MySQL commands

Command	Action
ALTER	Alter a database or table
BACKUP	Back up a table
\c	Cancel input
CREATE	Create a database
DELETE	Delete a row from a table
DESCRIBE	Describe a table's columns
DROP	Delete a database or table
EXIT (CTRL-C)	Exit
GRANT	Change user privileges
HELP (\h, \?)	Display help
INSERT	Insert data
LOCK	Lock table(s)
QUIT	(\q) Same as EXIT
RENAME	Rename a table
SHOW	List details about an object
SOURCE	Execute a file
STATUS (\s)	Display the current status
TRUNCATE	Empty a table
UNLOCK	Unlock table(s)
UPDATE	Update an existing record
USE	Use a database

5.1.6 Database Related Commands

5.1.6(A) Creating a Database

- A database is a collection of data. MySQL allows us to store and retrieve the data from the database in a efficient way.
- We can create a database using the **CREATE DATABASE** statement. But, if database already exits, it throws an error. To avoid the error, we can use the **IF NOT EXISTS** option with the CREATE DATABASE statement.

  Syntax : CREATE DATABASE <database name>;

- If a successful command execution will return a message that doesn't mean much yet—Query OK, 1 row affected (0.00 sec)—but will make sense soon.
- We can use created database for use using MySQL USE command

  USE <database name>;

- You should now see the message Database changed and will be able to see the database name in the prompt.
- For example, we will create database with the name **college** using CREATE command and we will use it by using USE command

Example :

```
MariaDB [(none)]> CREATE DATABASE college;
Query OK, 1 row affected (0.01 sec)

MariaDB [(none)]> USE college;
Database changed
MariaDB [college]>
```

You can check the created database by the following query:

```
MariaDB [(none)]> show databases;
+--------------------------+
| Database         |
+--------------------------+
| college          |
| information_schema |
| mysql |
| performance_schema|
| phpmyadmin |
| test             |
+--------------------------+
6 rows in set (0.002 sec)
```

5.1.6(B) Select Databases

- SELECT Database is used in MySQL to select a particular database to work with, when multiple databases are available with MySQL Server.

  Syntax : USE database_name;

```
MariaDB [(none)]> use college
Database changed
MariaDB [college]>
```

5.1.6(C) Drop Database

- We can drop/delete/remove a MySQL database easily with the MySQL **DROP DATABASE** command.
- It deletes all the tables of the database along with the database permanently.
- We should be careful while deleting any database because we will loose all the data available in the database.

  Syntax : DROP DATABASE database_name;

5.1.7 Table Related Commands

5.1.7(A) Creating a Table

- In order to create a table we have to choose an appropriate database for operation using USE command
- Table can be created using CREATE TABLE command into the database and by mentioning the fields with its type.

  Syntax : CREATE TABLE [IF NOT EXISTS] <table name> (<field name>dataType [optional parameters]) ENGINE = storage Engine;
- Example a student table is created in a college database, a table is created with columns rollno, name and percentage

Example :

```
MariaDB [(none)]> USE college;
Database changed

MariaDB [college]> CREATE TABLE student(
   ->rollnoVARCHAR(16),
   -> name VARCHAR(128),
   -> percent float(5,2))ENGINE MyISAM;
Query OK, 0 rows affected (0.19 sec)
```

- After execution of the command it generates a response Query OK, 0 rows affected, along with how long it took to execute the command. If you see an error message instead, check your syntax carefully.
- Every parenthesis and comma counts, and typing errors are easy to make.
- The ENGINE MyISAM tells MySQL the type of database engine to use for this table

5.1.7(B) Describe Command

- For checking whether your new table has been created we can use DESCRIBE command:

  Syntax : DESCRIBE<table name>;

- For example student can be checked using DESCRIBE command.

```
MariaDB [college]> DESCRIBE student;
+----------+-----------------+-------+-----+----------+---------+
| Field    | Type            | Null  | Key | Default  | Extra   |
+----------+-----------------+-------+-----+----------+---------+
| rollno   | varchar(16)     | YES   |     | NULL     |         |
| name     | varchar(128)    | YES   |     | NULL     |         |
| percent  | float(5,2)      | YES   |     | NULL     |         |
+----------+-----------------+-------+-----+----------+---------+
3 rows in set (0.03 sec)
```

- The **DESCRIBE** command is an invaluable debugging aid when you need to ensure that you have correctly created a MySQL table.
- You can also use it to remind your self about a table's field or column names and the types of data in each one. Let's look at each of the headings in detail:

1. **Field :** The name of each field or column within a table.
2. **Type :** The type of data being stored in the field.
3. **Null :** Whether a field is allowed to contain a value of NULL.
4. **Key :** MySQL supports *keys* or *indexes*, which are quick ways to look up and search for data. The Key heading shows what type of key (if any) has been applied.
5. **Default :** The default value that will be assigned to the field if no value is specified when a new row is created.
6. **Extra :** Additional information, such as whether a field is set to auto-increment.

5.1.7(C) Adding Data to a Table

- INSERT command can be used to add data in a table.

  Syntax : INSERT INTO <table name> (column_1,column_2,...) VALUES (value_1,value_2,...);

- For Example, three rows are inserted in the table student using IINSERT command

```
MariaDB [college]> INSERT INTO student(rollno,name,percent) VALUES ('CO101','Prasad Koyande',95.45);
Query OK, 1 row affected (0.07 sec)
MariaDB [college]> INSERT INTO student(rollno,name,percent) VALUES ('CO102','Vijay Patil',96.85);
Query OK, 1 row affected (0.00 sec)
MariaDB [college]> INSERT INTO student(rollno,name,percent) VALUES ('CO103','Yogita Khandagale',98.45);
Query OK, 1 row affected (0.00 sec)
```

- After every second line, you should see a Query OK message. Once all lines have been entered, type the following command, which will display the table's contents.
- The first part, INSERT INTO student, tells MySQL where to insert the following data. Then, within parentheses, the three column names are listed-*rollno*, *name* and *percent*—all separate by commas. This suggests that these are the fields into which the data is to be inserted.
- The second line of each INSERT command contains the keyword VALUES followed by four strings within parentheses, and separated by commas. This supply MySQL with the four values to be inserted into the three columns previously specified.
- Each item of data will be inserted into the corresponding column, in a one-to-one correspondence.
- If listing of the column is done in the wrong order of the data, the data would go into the wrong columns.
- The number of columns must match the number of data items

5.1.7(D) Deleting a Table

- Deleting a table is very easy.
- Table can be deleted using DROP TABLE command

  Syntax : DROP TABLE <table name>;
- For Example, we have created a table name 'sample' with a field 'no' which can be deleted using DROP TABLE command.

```
MariaDB [college]> CREATE TABLE sample(no INT);
Query OK, 0 rows affected (0.30 sec)

MariaDB [college]> DESCRIBE sample;
+-------+-----------+------+-----+---------+----------+
| Field | Type      | Null | Key | Default | Extra    |
+-------+-----------+------+-----+---------+----------+
| no    | int(11)   | YES  |     | NULL    |          |
+-------+-----------+------+-----+---------+----------+
1 row in set (0.03 sec)

MariaDB [college]> DROP TABLE sample;
Query OK, 0 rows affected (0.40 sec)

MariaDB [college]> SHOW tables;
+---------------------+
| Tables_in_college   |
+---------------------+
| student             |
+---------------------+
1 row in set (0.00 sec)

MariaDB [college]>
```

5.1.8 Querying a MySQL Database

1. SELECT Command

- The SELECT command is used to extract data from a table.
- **Syntax:** SELECT something FROM <tablename>;
- The something can be an * (asterisk) as you saw before, which means every column, or you can choose to select only certain columns.
- For Example shows how to select just the rollno and name from table student.

Example.

```
MariaDB [college]> SELECT rollno,name from student;
+--------+-------------------------+
| rollno | name                    |
+--------+-------------------------+
| CO101  | Prasad Koyande          |
| CO102  | Vijay Patil             |
| CO103  | YogitaKhandagale        |
+--------+-------------------------+
3 rows in set (0.00 sec)
```

2. DELETE Command

- A row from a table can be removed using DELETE command.
- The syntax is similar to the SELECT command and allows you to narrow down the exact row or rows to delete using qualifiers such as WHERE and LIMIT.
- For example, a student with a roll no CO103 can be removed from the table using DELETE command and with where qualifiers.

Example :

```
MariaDB [college]> DELETE from student where rollno='CO103';
Query OK, 1 row affected (0.03 sec)
MariaDB [college]> SELECT * from student;
+--------+----------------+---------+
| rollno | name           | percent |
+--------+----------------+---------+
| CO101  | Prasad Koyande |   95.45 |
| CO102  | Vijay Patil    |   96.85 |
+--------+----------------+---------+
```

```
2 rows in set (0.05 sec)
```

3. WHERE Command

- The WHERE keyword enables you to narrow down queries by returning only those where a certain expression is true.
- For Example, we want to display the record with a roll no CO101 can be fetched using WHERE and the equality operator =.
  - Example : Using the WHERE keyword

```
MariaDB [college]> SELECT * FROM student WHERE rollno="CO101";
+----------+---------------------+----------+
| rollno| name          | percent |
+----------+---------------------+---------- -+
| CO101  | Prasad Koyande |  95.45 |
+----------+---------------------+-----------+
row in set (0.00 sec)
```

- User can also do pattern matching, for searches using the LIKE qualifier, which allows searches on parts of strings. LIKE qualifier should be used with a % character before or after some text. When placed before a keyword, % means anything before and after a keyword, it means anything after.
- The % will also match if there is nothing in the position it occupies; in other words, it can match an empty string.
- For Example, it searches all the name of the students, which starts with Vijay, for doing so we have to use LIKE qualifier and the % character after Vijay.

Example :

```
MariaDB [college]> SELECT * FROM student WHERE name like "Vijay%";
+--------+-------------+---------+
| rollno | name        | percent |
+--------+-------------+---------+
| CO102  | Vijay Patil |  96.85 |
+--------+-------------+---------+
1 row in set (0.00 sec)
```

5.1.9 Accessing MySQL via phpMyAdmin

- Web interface is much quicker and simpler to use a program such as phpMyAdmin to manage your databases and tables.
- To use this, type the following to call up the XAMPP main page shown in Fig. 5.1.1.

http://localhost/xampp

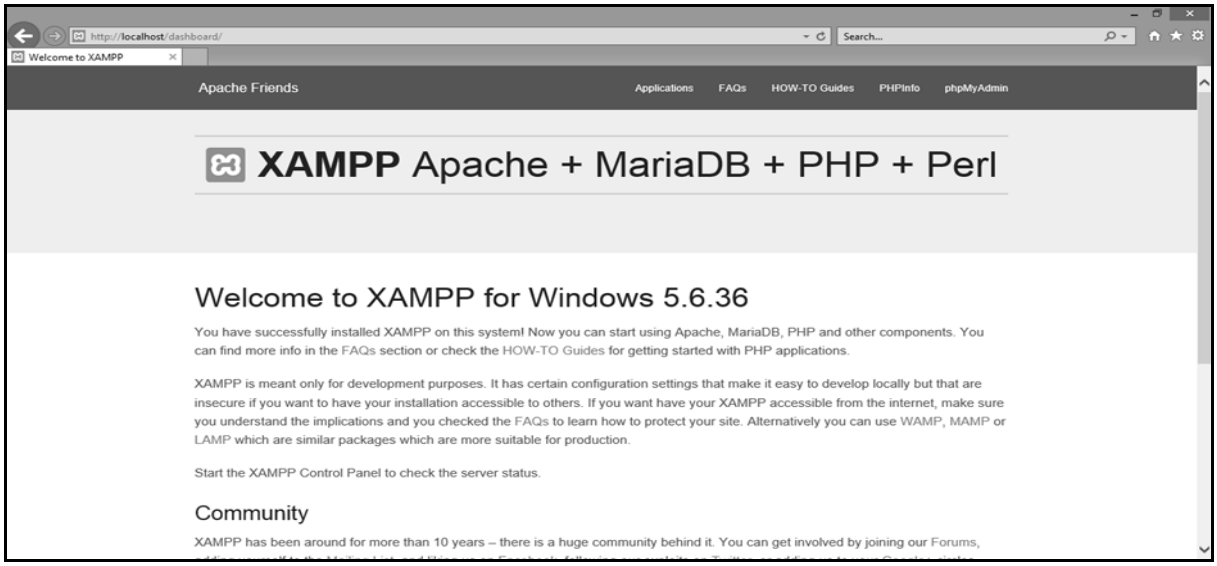


Fig. 5.1.1 : XAMPP Main Page

- On the right top of the XAMPP main page you will be able to see phpMyAdmin menu, on that menu the main pagephpMyAdmin screen will appear, you can click to select anydatabase you wish to work with. This will open the database and display its tables.

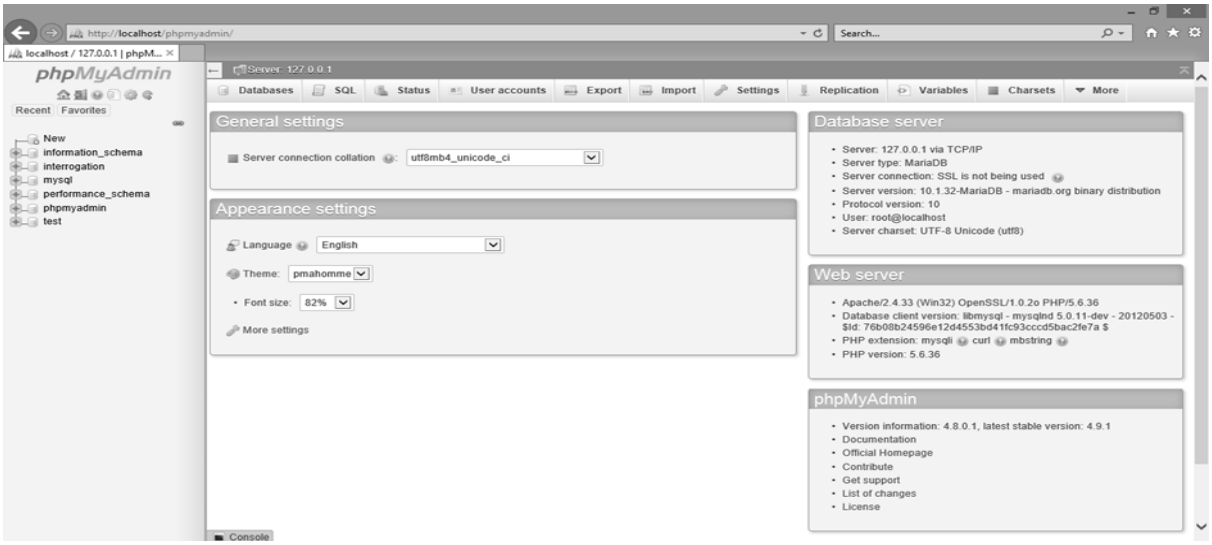


Fig. 5.1.2 : phpMyAdmin Main Page

- You can also click New to create a new database. From here you can perform all the main operations, such as creating new databases, adding tables, creating indexes, and much more. To read the supporting documentation for phpMyAdmin, visit https://docs.phpmyadmin.net.

5.1.10 MySQL Storage Engine

Q. List storage engine in MySQL. Explain any one.

- A storage engine is a software module that a database management system uses to create, read, update data from a database.
- There are two types of storage engines in MySQL: transactional and non-transactional.
- There are many storage engines available in MySQL and they are used for different purposes.
- The show engines command shows all available engines that the server supports.

Engine	Description
MyISAM	*MyISAM* is the original storage engine, fast storage engine manages non transactional tables, provides high-speed storage and retrieval, supports full text searching. It is used mostly in Web and data warehousing.
InnoDB	his is the default storage engine for MySQL 5.5 and higher. It provides transaction-safe (ACID compliant) tables, supports FOREIGN KEY referential-integrity constraints. It supports commit, rollback, and crash-recovery capabilities to protect data. It also supports row-level locking. Increases performance when used in a multiuser environment. It stores data in clustered indexes which reduces I/O for queries based on

Engine	Description
	primary keys.
MEMORY	Provides in-memory tables, formerly known as HEAP. It sores all data in RAM for faster access than storing data on disks. Useful for quick looks up of reference and other identical data. Memory storage engine is ideal for creating temporary tables or quick lookups. The data is lost when the database is restarted.
MERGE	Groups more than one similar MyISAM tables to be treated as a single table, can handle non transactional tables, included by default. Merge tables help manage large volumes of data more easily. Good for data warehousing environments.
EXAMPLE	You can create tables with this engine, but cannot store or fetch data. Purpose of this is to teach developers about how to write a new storage engine.
ARCHIVE	Archive storage engine is optimized for high speed inserting. It compresses data as it is inserted. It does not support transactions. It is ideal for storing and retrieving large amounts of seldom referenced historical, archived data.
CSV	*CSV* stores data in CSV (Comma Separated Value format in a text file)files. It provides great flexibility because data in this format is easily integrated into other applications.
BLACKHOLE	The *Blackhole* storage engine accepts but does not store data. Retrievals always return an empty set. The functionality can be used in distributed database design where data is automatically replicated, but not stored locally. This storage engine can be used to perform performance tests or other testing.
FEDERATED	Stores data in a remote database.*Federated* storage engine offers the ability to separate MySQL servers to create one logical database from many physical servers. Queries on the local server are automatically executed on the remote (federated) tables. No data is stored on the local tables. It is good for distributed environments.

5.2 Connecting to MySQL Database

- PHP used as an interface to a MySQL it will be used to format the results which are returned from SQLqueries in a form that can be visible on a web page.
- Using PHP we can perform all operations that we can perform using MySQL server using the correct username and password.
- However, instead of using MySQL's command line to enter instructions and view output,you will create query strings that are passed to MySQL.
- Query String gets executed and MySQL returns itsresponse, it will come as a data structure that PHP can recognize instead of the formattedoutput you see when you work on the command line.
- PHP commands will retrieve the data and format it in the form that can be displayed on a web page.

5.2.1 Connecting MySQL Database Server from PHP

- **The Process :** The process of using MySQL with PHP is as follows :
  1. Connect to MySQL and select the database to use.
  2. Build a query string.
  3. Perform the query.
  4. Retrieve the results and output them to a web page.
  5. Repeat steps 2 to 4 until all desired data has been retrieved.

6. Disconnect from MySQL.

5.2.2 Creating a Login File

- When a web site is developed with PHP it contains multiple program files that will require access to MySQL and will thus need the login and password details.
- Therefore, it is good to create a single file to store login credentials and then include that file wherever it is needed.

Example . The login.php file

```
<?php
$hn = 'localhost';
$db = 'college';
$un = 'root';
$pw = '';
?>
```

- Type the example, replacing username and password with the values you use for your MySQL database.
- The hostname localhost should work as long as you are using a MySQL database on your local system, and the database publications should work if you are typing the examples.
- The enclosing <?php and ?> tags are especially important for the login.php file, because they mean that the lines between can be interpreted only as PHP code.
- If you were to leave them out and someone were to call up the file directly from your website, it would display as text and reveal your secrets. But, with the tags in place, all that person will see is a blank page. The file will correctly include in your other PHP files.
- The $hn variable will tell PHP which computer to use when connecting to a database. This is required, because you can access MySQL databases on any computer connected to your PHP installation, and that potentially includes any host anywhere on the Web.
- We are working locally on our own system, therefore in place by specifying a domain such as mysql.myserver.com, you can just use the word localhost (or the IP address 127.0.0.1).
- We will be using, $db for specifying a database.

5.2.3 Connecting to a MySQL Database

- All login credentials that are used to log MySQL server are saved in file login.php, you can include it in any PHP files that will need to access the database by using the require_once statement.
- This is prefer able to an include statement, as it will generate a fatal error if the file is not found. And believe me, not finding the file containing the login details to your database is a fatal error.
- Also, using require_once instead of require means that the file will be read in only when it has not previously been included, which prevents wasteful duplicate disk accesses.

Example Connecting to a MySQL server with mysqli

```
<?php
    require_once 'login.php';
    $conn = new mysqli($hn, $un, $pw, $db);
    if ($conn->connect_error) die($conn->connect_error);
?>
```

- This example creates a new object called $ conn by calling a new instance of the mysqli method, passing all the values retrieved from the login.php file. Error checking is achieved by referencing the $conn->connect_error property.
- The -> operator indicates that the item on the right is a property or method of the object on the left. In this case, if connect_error has a value, then there was an error, so we call the die function and display that property, which details the connection error.
- The $conn object is used in the following examples to access the MySQL database. The die function is good to use when you are developing PHP code.

5.2.4 Building and Executing a Query

- It is very simple to send a query from PHP to MySQL BY using the query method of a connection object. The example shows you how to use it.
- Example. Querying a database with mysqli

```
<?php
$query = "SELECT * FROM student";
$result = $conn->query($query);
if (!$result) die($conn->error);
?>
```

- Here the variable $query is assigned a string containing the query to be made, and then passed to the query method of the $conn object, which returns a result that we place in the object $result.
- If $result is FALSE, there was a problem and the error property of the connection object will contain the details, so the die function is called to display that error.
- All the data returned by MySQL is now stored in an easily interrogatable format in the $result object.

5.2.5 Fetching a result

- Once you have an object returned in $result, you can use it to extract the data you want, one item at a time, using the fetch_assoc() method of the object.
- Example combines and extends the previous examples into a program that you can type and run yourself to retrieve these results.
- I suggest that you save this script using the filename sample1.php.
- **Example :** Fetching results one cell at a time

```
<?php
    require_once 'login.php';
    $conn = new mysqli($hn, $un, $pw, $db);
    if ($conn->connect_error) die($conn->connect_error);
    $query = "SELECT * FROM student";
    $result = $conn->query($query);
    if (!$result) die($conn->error);
    $rows = $result->num_rows;
    for ($j = 0 ; $j < $rows ; ++$j)
    {
    $result->data_seek($j);
    echo 'Roll No.: ' . $result->fetch_assoc()['rollno'] . '<br/>';
    $result->data_seek($j);
    echo 'Name: ' . $result->fetch_assoc()['name'] . '<br/>';
    $result->data_seek($j);
    echo 'Percentage: ' . $result->fetch_assoc()['percent'] . '<br/><br/>';
    }
    $result->close();
    $conn->close();
?>
```

- Here, to seek to the correct row each time around the loop, we call the data_seek method of $result before fetching each item of data.
- Then we call the fetch_assoc method to retrieve the value stored in each cell, and output the result using echo statements.
- You will probably agree that all this data seeking is rather cumbersome and that there ought to be a more efficient method of achieving the same result.
- And, indeed, there is a better method, which is to extract a row at a time.

5.2.6 Fetching a Row :

- To fetch one row at a time, fetch_array() method is used to fetched each row entirety at a time.
- This returns a single row of data as an array, which is then assigned to the array $row.
- The fetch_array() method can return three types of array according to the value passed to it:
- Example. Fetching results one row at a time using fetch_array() method.

```
<?php
    require_once 'login.php';
    $conn = new mysqli($hn, $un, $pw, $db);
    if ($conn->connect_error) die($conn->connect_error);
    $query = "SELECT * FROM student";
    $result = $conn->query($query);
    if (!$result) die($conn->error);
    $rows = $result->num_rows;
    for ($j = 0 ; $j < $rows ; ++$j)
    {
    $result->data_seek($j);
    $row = $result->fetch_array(MYSQLI_ASSOC);
    echo 'Roll No.: ' . $row['rollno'] . '<br/>';
    echo 'Name: ' . $row['name'] . '<br/>';
    echo 'Percentage: ' . $row['percent'] . '<br/><br/>';
    }
    $result->close();
    $conn->close();
?>
```

MYSQLI_NUM

- Numeric array. Each column appears in the array in the order in which you defined it when you created (or altered) the table.
- In our case, the zeroth element of the array contains the Author column, element 1 contains the Title, and so on.

MYSQLI_ASSOC

Associative array. Each key is the name of a column. Because items of data are referenced by column name

(rather than index number), use this option where possible inyour code to make debugging easier and help other programmers better manage your code.

MYSQLI_BOTH

Associative and numeric array.

Associative arrays are usually more useful than numeric ones because you can refer to each column by name, such as $row['author'], instead of trying to remember where it is in the column order. So this script uses an associative array, leading us to pass

```
MYSQLI_ASSOC.
```

5.2.7 Closing a connection

- PHP will eventually return the memory it has allocated for objects after you have finished with the script, so in small scripts, you don't usually need to worry about releasing memory yourself.
- However, if you are allocating a lot of result objects or fetching large amounts of data, it can be a good idea to free the memory you have been using to prevent problems later in your script.
- This becomes particularly important on higher-traffic pages, because the amount of memory consumed in a session can rapidly grow.
- Therefore, note the calls to the close methods of the objects $result and $conn in the preceding scripts, as soon as each object is no longer needed, like this:

```
$result->close();
$conn->close();
```

5.3 Database Operations

5.3.1 Insert Operation

- Data can be inserted into an existing database table with an INSERT INTO query.
- SQL query using the INSERT INTO statement with appropriate values, after that we will execute this insert query through passing it to the PHP query() function to insert data in table.
- For example a student data is inserted into a table using INSERT INTO statement.

```
<?php
    require_once 'login.php';
    $conn = new mysqli($hn, $un, $pw, $db);
    if ($conn->connect_error) die($conn->connect_error);
$query = "INSERT INTO student(rollno,name,percent) VALUES ('CO103','Yogita Khandagale',98.45)";
    $result = $conn->query($query);
    if (!$result) die ("Database access failed: " . $conn->error);
?>
```

- The most efficient way to populate MySQL with data is to create an arrayand insert the data with a single query.

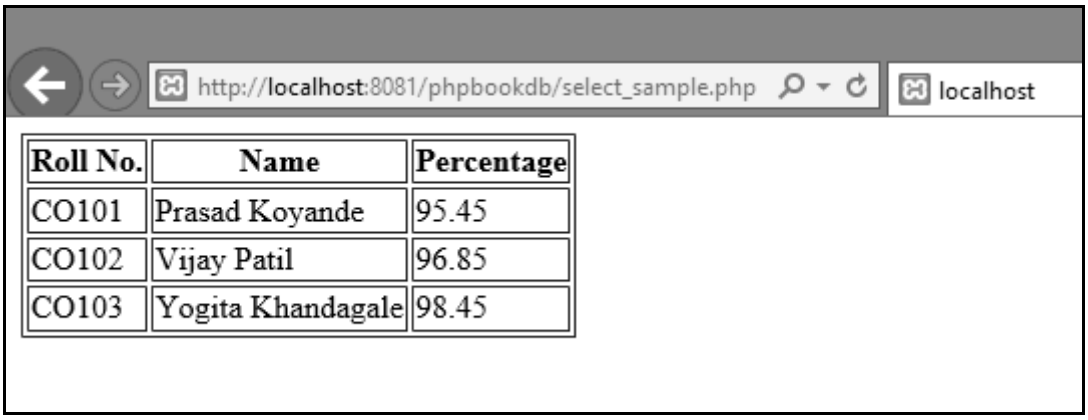
5.3.2 Retrieving the Query Result

- The data can be retrieved from the table using SQL SELECT statement which is used to select the records from database tables.
- SQL query using the SELECT statement will be executed by passing this SQL query to the PHP query() function to retrieve the table data.
- For example data from the student table can be executed by using the SELECT statement.
- **select_sample.php**

```
<?php
    require_once 'login.php';
    $conn = new mysqli($hn, $un, $pw, $db);
```

```
    if ($conn->connect_error) die($conn->connect_error);
    $query = "SELECT * FROM student";
    $result = $conn->query($query);
    if (!$result) die ("Database access failed: " . $conn->error);
    $rows = $result->num_rows;
echo "<table border='1'><tr><th>Roll No.</th><th>Name</th><th>Percentage</th></tr>";
    for ($j = 0 ; $j < $rows ; ++$j)
    {
    $result->data_seek($j);
    $row = $result->fetch_array(MYSQLI_NUM);
    echo "<tr>";
    for ($k = 0 ; $k < 3 ; ++$k) echo "<td>$row[$k]</td>";
    echo "</tr>";
    }
    echo "</table>";
?>
```

- This code simply issues the MySQL query SELECT * FROM student and then displays allthe rows returned. Its output is as follows :



Roll No.	Name	Percentage
CO101	Prasad Koyande	95.45
CO102	Vijay Patil	96.85
CO103	Yogita Khandagale	98.45

5.3.3 Update Operation

- The UPDATE statement is used to change or modify the existing records in a database table.
- UPDATE statement is typically used in conjugation with the WHERE clause to apply the changes to only those records that matches specific criteria.
- SQL query will be formed using the UPDATE statement and WHERE clause, after that a query will be executed by passing it to the PHPquery() function to update the tables records.
- For example percentage of roll no. C101 will be updated to '98.99' from student table by using UPDATE statement.

```
<?php
    require_once 'login.php';
    $conn = new mysqli($hn, $un, $pw, $db);
    if ($conn->connect_error) die($conn->connect_error);
    $query = "UPDATE student SET percent=98.99 WHERE rollno='CO101'";
    $result = $conn->query($query);
    if (!$result) die ("Database access failed: " . $conn->error);
?>
```

5.3.4 Delete Operations

- Record scan be deleted from a table using the SQL DELETE statement.
- DELETE statement is typically used in conjugation with the WHERE clause to delete only those records that matches specific criteria or condition.
- SQL query is formed using the DELETE statement and WHERE clause, after that will be executed by passing this query to the PHP query() function to delete the tables records.
- For example a student record with a roll no. 'CO103' will be deleted by using DELETE statement and WHERE clause.

```
<?php
    require_once 'login.php';
    $conn = new mysqli($hn, $un, $pw, $db);
    if ($conn->connect_error) die($conn->connect_error);
    $query = "DELETE from student WHERE rollno='CO103'";
    $result = $conn->query($query);
    if (!$result) die ("Database access failed: " . $conn->error);
?>
```

- Example : Removing roll id CO103 from the student table

```
<?php
    require_once 'login.php';
    $conn = new mysqli($hn, $un, $pw, $db);
    if ($conn->connect_error) die($conn->connect_error);
    $query = "DELETE from student WHERE rollno='CO103'";
    $result = $conn->query($query);
    if (!$result) die ("Database access failed: " . $conn->error);
?>
```

Questions

Q. 1 What is the purpose of the semicolon in MySQL queries?

Q. 2 Explain field types of MySQL

Q. 3 Which command would you use to view the available databases or tables?

Q. 4 How can you view the structure of a table?

Q. 5 Using the SELECT...WHERE construct, how would you return only rows containing the word Langhorne somewhere in the author column of the classics table used in this chapter?

Q. 6 How do you connect to a MySQL database using mysqli?

Q. 7 How do you connect mysql database with PHP ?

Q. 8 How do you submit a query to MySQL using mysqli?

Q. 9 How can you retrieve a string containing an error message when a mysqli error occurs?

Q. 10 How can you determine the number of rows returned by a mysqli query?

Q. 11 How can you retrieve a particular row of data from a set of mysqli results?

Q. 12 What negative effects can happen if you do not close the objects created by mysqli methods?

Q. 13 List types of MySQL databases

Q. 14 Write the PHP code for fetching the data from a database to webpage?

Q. 15 What is difference between mysql_fetch_object and mysql_fetch_array ?

Q. 16 Write use of mysql_fetch_row() and mysql_fetch_array() command

Q. 17 Explain mysql_num_rows and mysql_selectdb command.

Q. 18 Explain mysql_fetch_array() function.

Q. 19 Implement Database connectivity. Write a program to insert, update and delete an element from database.

Q. 20 Describe mysql_connect() function of MySQL database using PHP.

Q. 21 Write PHP script to select records in table of MySQL database

Q. 22 Write PHP script to update record in table of MySQL database

Q. 23 Write PHP script to delete records in table of MySQL database

Q. 24 List storage engine in MySQL. Explain any one

Q. 25 Explain mysql_fetch_array() function

Q. 26 Write a PHP Script to display all the records of students (Name, Address, Class, Semester) from MySQL table std_info in table format.

Q. 27 Explain use of query string with example in PHP.

Q. 28 Write a PHP script to insert one record in student registration table (roll_no, name, city, mobile_no) in MySQL database.

Q. 29 Write a PHP script to read Account information for customer name, account_no, account_type, branch_name, city, amount from Account table and display all this information in table format on output screen. Account table is in MySQL database

.

❑❑❑