**Group A : Study of Databases**

# Assignment No:- 1

**Aim:-** Study of MySQL Open source software. Discuss the characteristics like efficiency, scalability, performance and transactional properties

**What is Database?**

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds. So nowadays, we use relational database management systems (RDBMS) to store and manage huge Volume of data. This is called relational database because all the data is stored into different tables and Relations are established using primary keys or other keys known as foreign keys.

**What is DBMS?**

A software system that enables users to define, create, maintain, and control access to the database. The DBMS is the software that interacts with the users’ application programs and the database.

Typically, a DBMS provides the following facilities:

* It allows users to define the database, usually through a Data Definition Language (DDL). The DDL allows users to specify the data types and structures and the constraints on the data to be stored in the database.
* It allows users to insert, update, delete, and retrieve data from the database, usually through a Data Manipulation Language(DML). The most common query language is the Structured Query Language (SQL, pronounced ‘S-Q-L’, or sometimes ‘See-Quel’), which is now both the formal and de facto standard language for relational DBMSs.
* It provides controlled access to the database. For example, it may provide:
* A security system, which prevents unauthorized users accessing the database.
* An integrity system, which maintains the consistency of stored data
* A concurrency control system, which allows shared access of the database.
* A recovery control system, which restores the database to a previous consistent state following a hardware or software failure
* A user-accessible catalog, which contains descriptions of the data in the database.

**What is RDBMS?**

A relational database refers to a database that stores data in a structured format, using rows and columns. This makes it easy to locate and access specific values within the database. It is "relational" because the values within each table are related to each other. Tables may also be related to other tables. The relational structure makes it possible to run queries across multiple tables at once.

**RDBMS Terminology:**

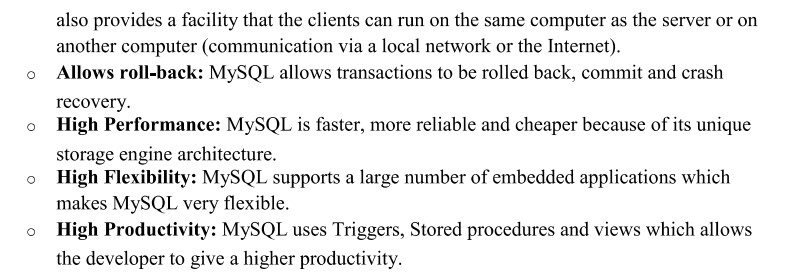
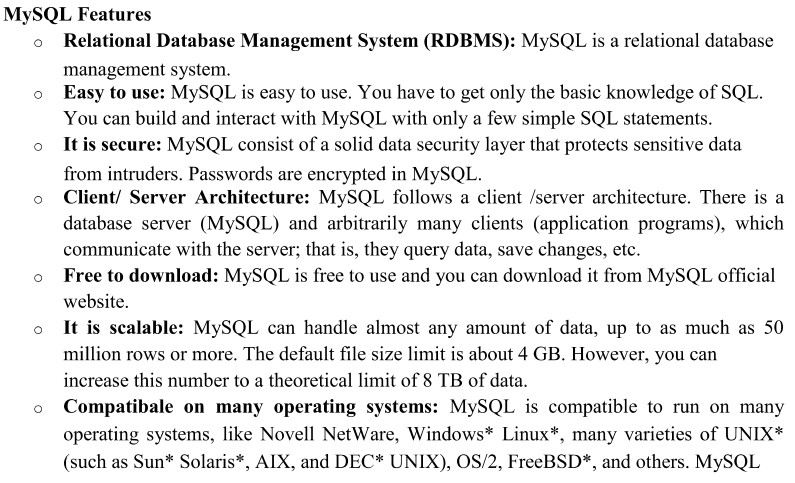
Before we proceed to explain MySQL database system, let's revise few definitions related to database.

* Database: A database is a collection of tables, with related data.
* Table: A table is a matrix with data. A table in a database looks like a simple spread sheet.
* Column: One column(data element) contains data of one and the same kind, for example the Column post code.
* Row: A row (tuple, entry or record) is a group of related data, for example the data of one subscription.
* Redundancy: Storing data twice, redundantly to make the system faster.
* Primary Key: A primary key is unique. A key value cannot occur twice in one table. With a key, you can find at most one row.
* Foreign Key: A foreign key is the linking pin between two tables.
* Compound Key: A compound key(composite key) is a key that consists of multiple columns, Because one column is not sufficiently unique.

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons — > MySQL is released under an open-source license. So you have nothing to pay to use it. > MySQL is a very powerful program in its own right.

It handles a large subset of the functionality of the most expensive and powerful database packages. > MySQL uses a standard form of the well-known SQL data language. > MySQL works on many operating systems and with many languages including PIP, PERL, C, C++, JAVA, etc. > MySQL works very quickly and works well even with large data sets. > MySQL is very friendly to HIP, the most appreciated language for web development. > MySQL supports large databases, up to 50 million rows or more in a table.

The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB). > MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.



**Conclusion:** In this way , study of MYSQL and its features is done.

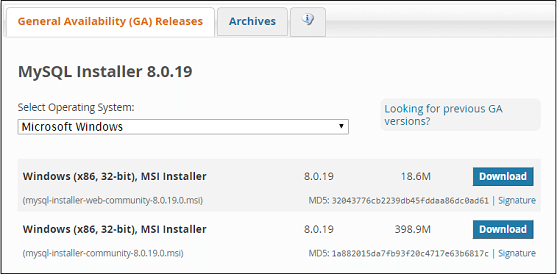
# Assignment No :- 2

**Aim :-**Install and configure client and server of MySQL.(Show all commands and necessary steps for installation and configuration)

**Download and Installation Steps :-**

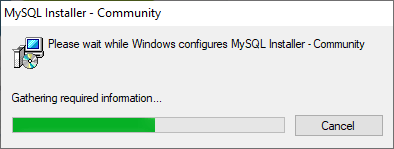
**Step 1:** Go to the [official website](https://dev.mysql.com/downloads/installer/) of MySQL and download the community server edition software. Here, you will see the option to choose the Operating System, such as Windows.

**Step 2:** Next, there are two options available to download the setup. Choose the version number for the MySQL community server, which you want. If you have good internet connectivity, then choose the mysql-installer-web-community. Otherwise, choose the other one.

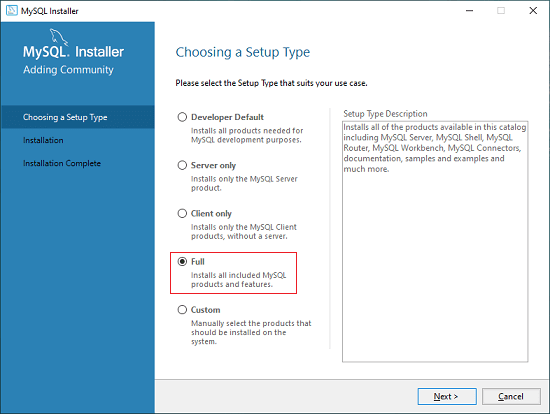


### **Installing MySQL on Windows**

**Step 1:** After downloading the setup, unzip it anywhere and double click the MSI **installer .exe file.** It will give the following screen:

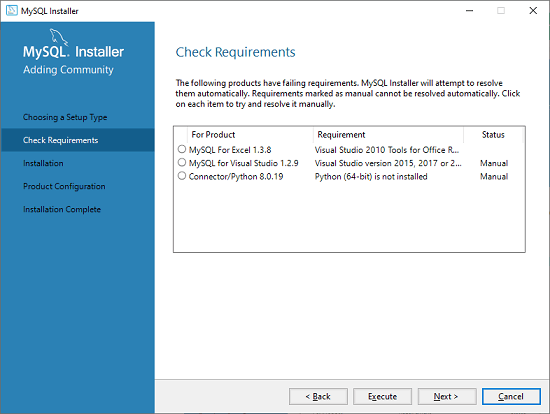


**Step 2:** In the next wizard, choose the **Setup Type**. There are several types available, and you need to choose the appropriate option to install MySQL product and [features](https://www.javatpoint.com/mysql-features). Here, we are going to select the **Full** option and click on the Next button.

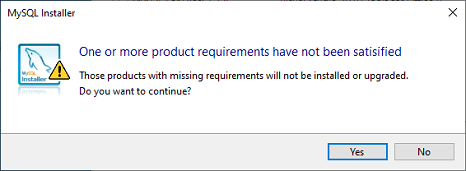


This option will install the following things: MySQL Server, MySQL Shell, MySQL Router, [MySQL Workbench](https://www.javatpoint.com/mysql-workbench), MySQL Connectors, documentation, samples and examples, and many more.

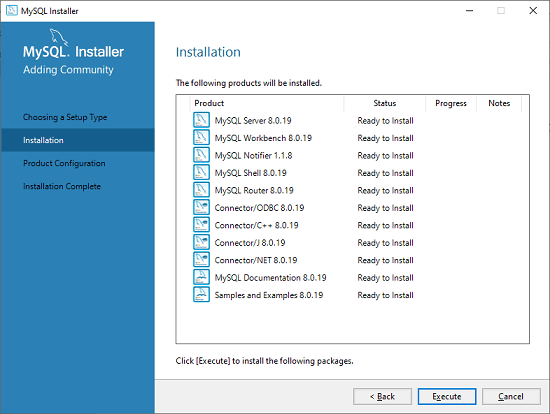
**Step 3:** Once we click on the Next button, it may give information about some features that may fail to install on your system due to a lack of requirements. We can resolve them by clicking on the **Execute** button that will install all requirements automatically or can skip them. Now, click on the Next button.



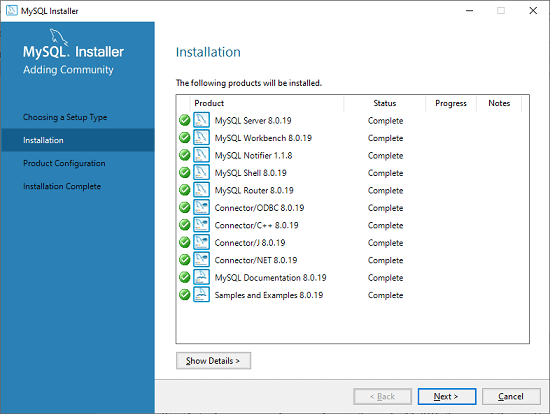
**Step 4:** In the next wizard, we will see a dialog box that asks for our confirmation of a few products not getting installed. Here, we have to click on the **Yes** button.



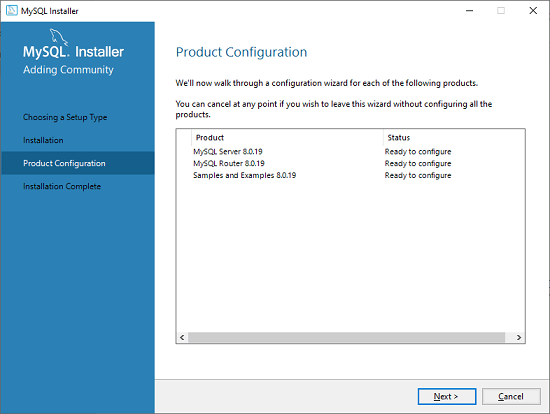
After clicking on the Yes button, we will see the list of the products which are going to be installed. So, if we need all products, click on the Execute button.



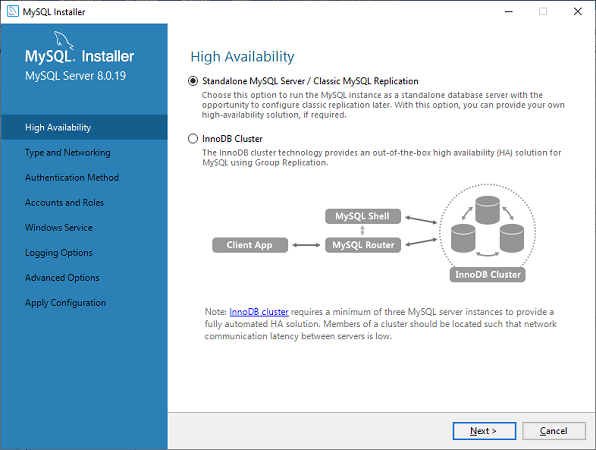
**Step 5:** Once we click on the Execute button, it will download and install all the products. After completing the installation, click on the Next button.



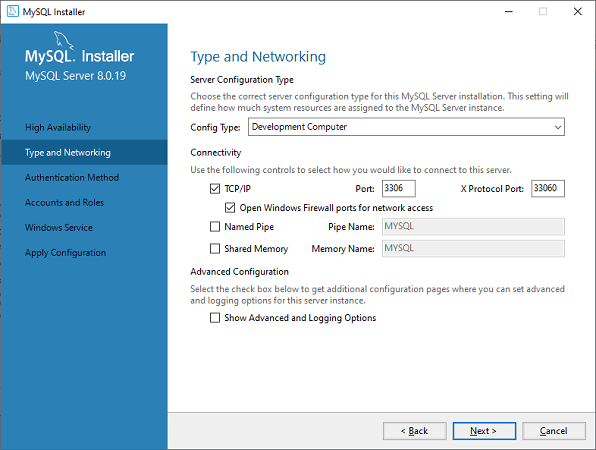
**Step 6:** In the next wizard, we need to configure the MySQL Server and Router. Here, I am not going to configure the Router because there is no need to use it with MySQL. We are going to show you how to configure the server only. Now, click on the Next button.



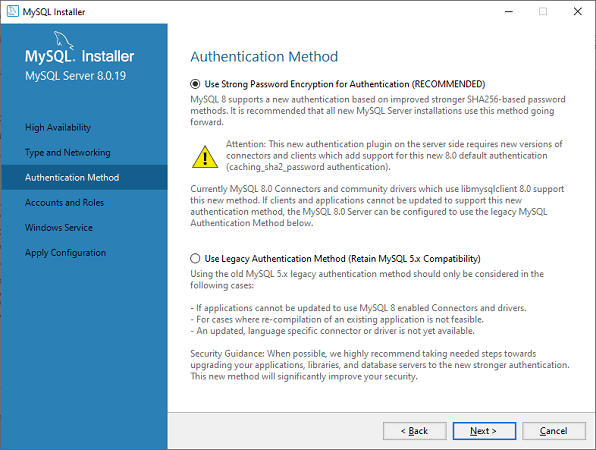
**Step 7:** As soon as you will click on the Next button, you can see the screen below. Here, we have to configure the MySQL Server. Now, choose the Standalone MySQL Server/Classic MySQL Replication option and click on Next. Here, you can also choose the InnoDB Cluster based on your needs.



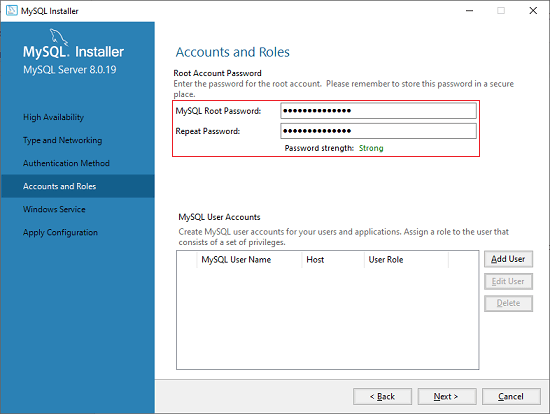
**Step 8:** In the next screen, the system will ask you to choose the Config Type and other connectivity options. Here, we are going to select the **Config Type** as 'Development Machine' and Connectivity as **TCP/IP,** and **Port Number** is 3306, then click on Next.



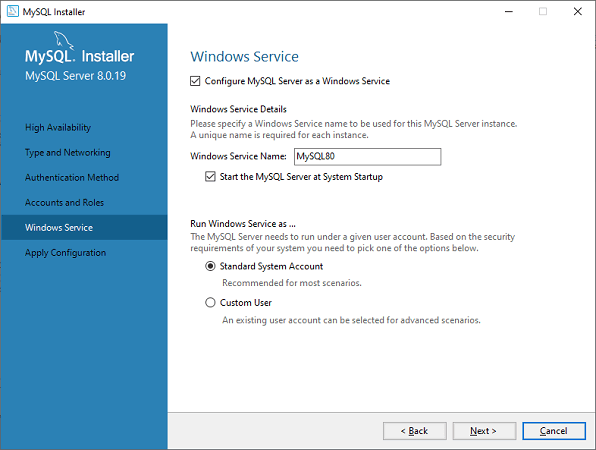
**Step 9:**Now, select the Authentication Method and click on Next. Here, I am going to select the first option.



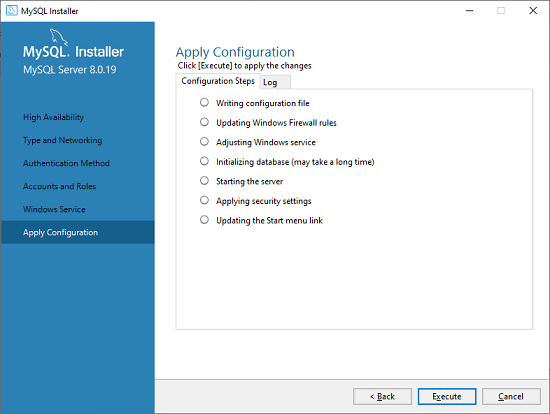
**Step 10:** The next screen will ask you to mention the MySQL Root Password. After filling the password details, click on the Next button.



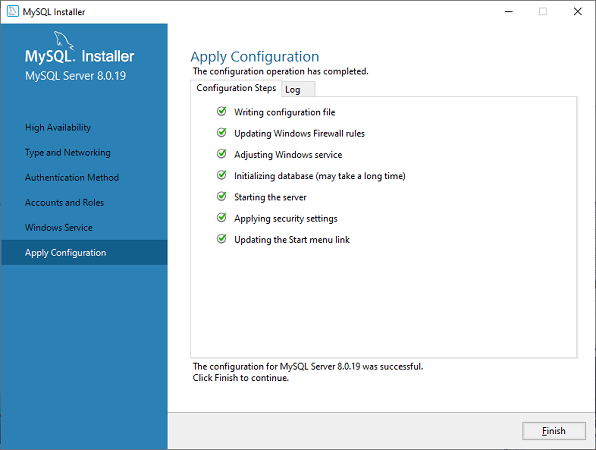
**Step 11:** The next screen will ask you to configure the Windows Service to start the server. Keep the default setup and click on the Next button.



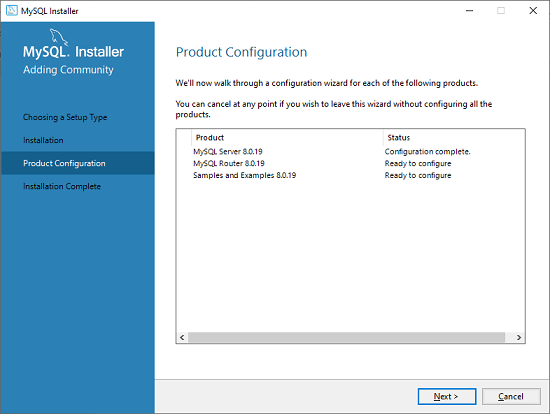
**Step 12:**In the next wizard, the system will ask you to apply the Server Configuration. If you agree with this configuration, click on the Execute button.



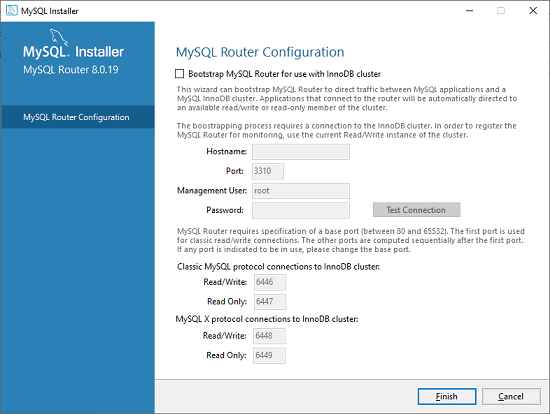
**Step 13:** Once the configuration has completed, you will get the screen below. Now, click on the **Finish** button to continue.



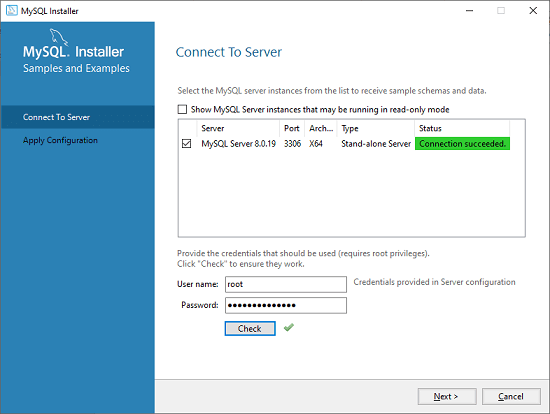
**Step 14:** In the next screen, you can see that the Product Configuration is completed. Keep the default setting and click on the Next-> Finish button complete the MySQL package installation.to



**Step 15:** In the next wizard, we can choose to configure the Router. So click on Next->Finish and then click the Next button.

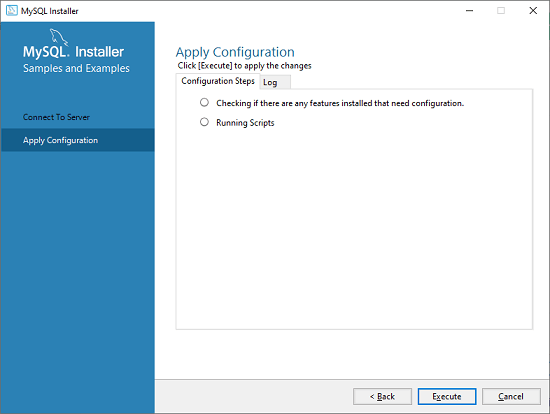


**Step 16:** In the next wizard, we will see the Connect to Server option. Here, we have to mention the root password, which we had set in the previous steps.

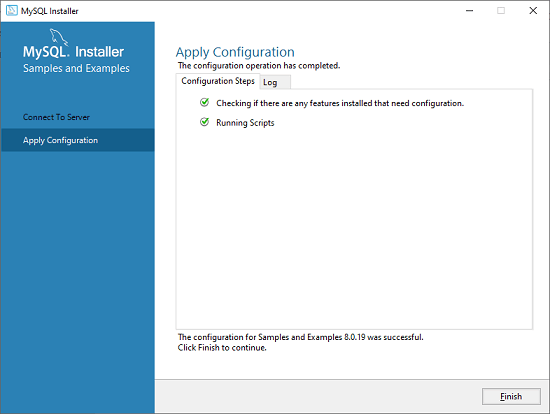


In this screen, it is also required to check about the connection is successful or not by clicking on the Check button. If the connection is successful, click on the Execute button. Now, the configuration is complete, click on Next.

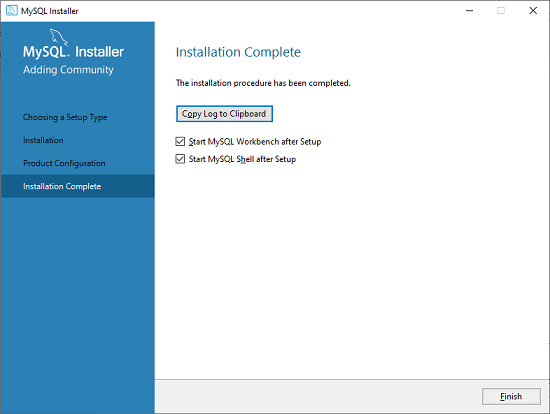
**Step 17:** In the next wizard, select the applied configurations and click on the Execute button.



**Step 18:** After completing the above step, we will get the following screen. Here, click on the Finish button.



**Step 19:** Now, the MySQL installation is complete. Click on the Finish button.

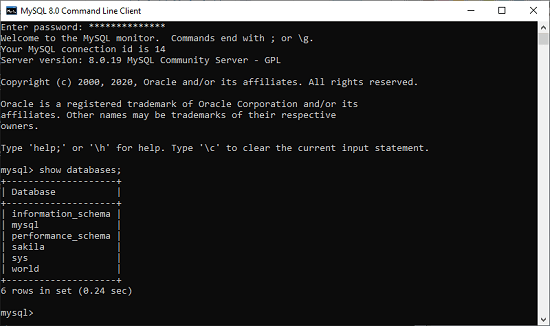


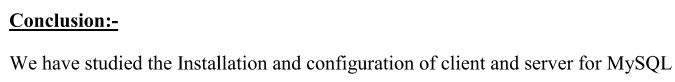
### **Verify MySQL installation**

Once MySQL has been successfully installed, the base tables have been initialized, and the server has been started, you can verify its working via some simple tests.

Open your MySQL **Command Line Client**; it should have appeared with a **mysql> prompt**. If you have set any password, write your password here. Now, you are connected to the MySQL server, and you can execute all the SQL command at mysql> prompt as follows:

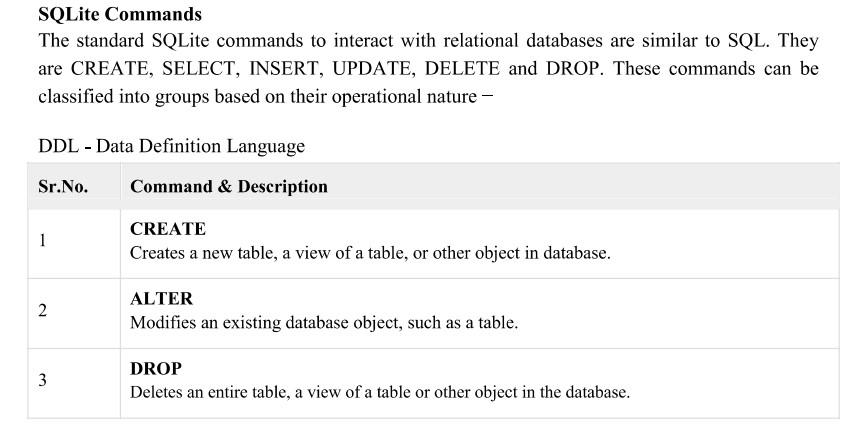
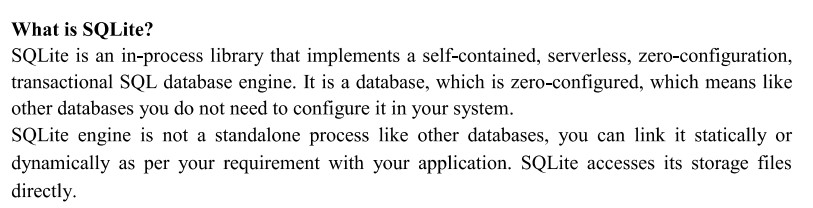
**For example**: Check the already created databases with show databases command:



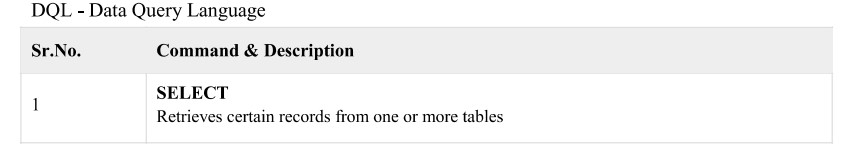


**Assignment No :- 3**

**Aim :-**Study of SQLite: What is SQLite? Uses of Sqlite. Building and installing SQLite.

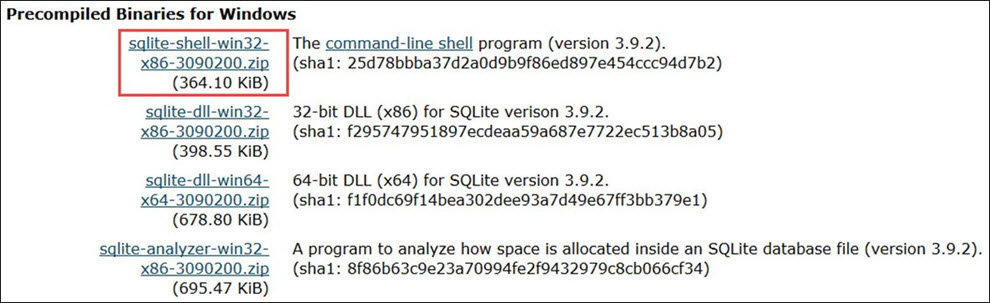






**Download and Installation of SQLite**

From the SQLite [official website](https://www.sqlite.org/index.html) in the download section. The following screenshot allows you to download different SQLite's installation packages for Windows:

[](https://www.guru99.com/images/SQLite/012116_0530_Downloadand1.jpg)

**The command line shell program:**

The highlighted download package is called the **Command-Line Program (CLP)**. CLP is a command line application that let you access the SQLite database management system and all the features of the SQLite. Using CLP, you can create and manage the SQLite database. And it is the tool that we will use throughout the tutorial.

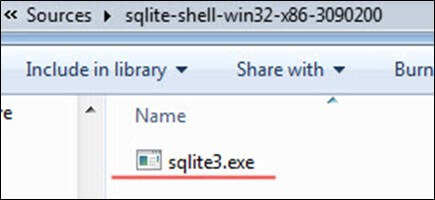
* 32-bit DLL(x86): The SQLite Database system core library for x86 platforms.
* 64-bit DLL (x64): The SQLite Database system core library for x64 platforms.

### Installing the Command-Line Program (CLP) on your machine:

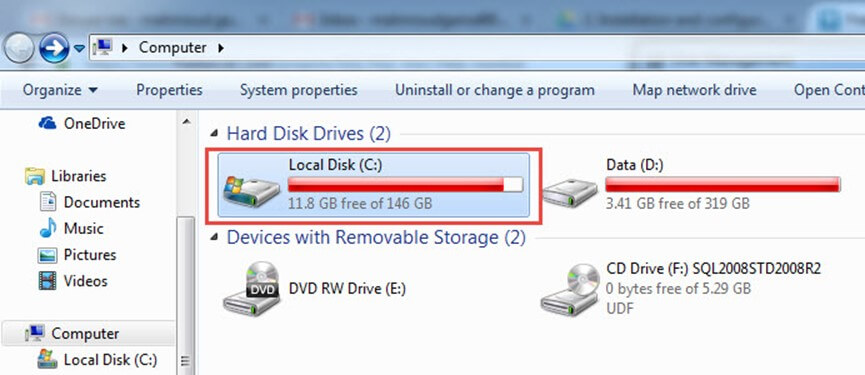
In the following steps, you will find the steps for how to install the Command-Line Program (CLP) on your machine:

**Step 1)** Download the highlighted download package from the previous image to your PC. It is a "**zip**" file.

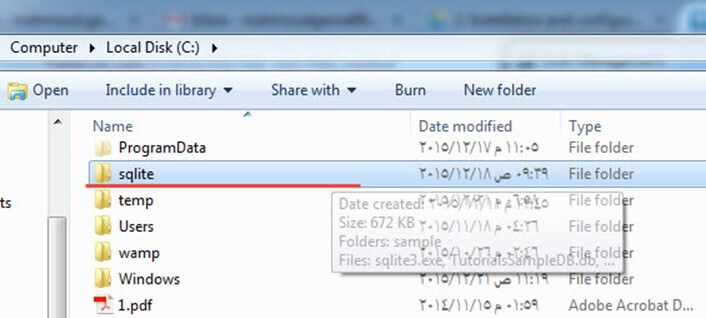
**Step 2)** Extract the zip file. You will find the "**sqlite3.exe**" in the extracted file as following:

[](https://www.guru99.com/images/SQLite/012116_0530_Downloadand2.jpg)

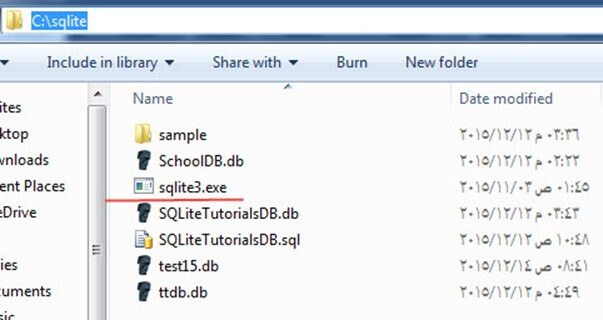
**Step 3)** Open My Computer, and double-click the partition **"C"** to navigate to it:

[](https://www.guru99.com/images/SQLite/012116_0530_Downloadand3.jpg)

**Step 4)** Create a new directory "**sqlite**":

[](https://www.guru99.com/images/SQLite/012116_0530_Downloadand4.jpg)

**Step 5)** Copy the file "**sqlite3.exe**" into it. This is what we will use through the tutorials to run SQLite queries:

[](https://www.guru99.com/images/SQLite/012116_0530_Downloadand5.jpg)

**//Execute some commands in sqlite such as create, insert….**

**Put here the screenshot of that command prompt**

**Conclusion:-**

We have studied the SQLite database and its uses.

**Group B MySQL**

**Assignment No:1**

**Aim:** Design any database with at least 3 entities and relationships between them. Draw suitable ER/EER diagram for the system.

**Entity Relationship Diagram:**

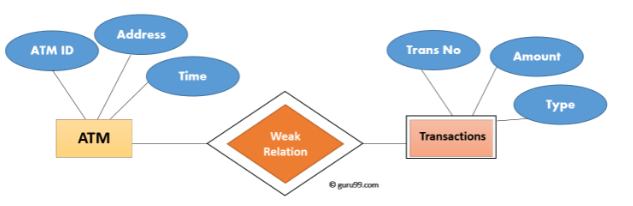
An **Entity–relationship model (ER model)** describes the structure of a database with the help of a diagram, which is known as **Entity Relationship Diagram (ER Diagram)**. An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

## What is an Entity Relationship Diagram (ER Diagram)?

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Lets have a look at a simple ER diagram to understand this concept.

## A simple ER Diagram:

In the following diagram we have two entities ATM and Transactions and their relationship. The relationship between ATM and Transactions is weak relation. The attributes for ATM entities are ATM id, Address and Time. For transaction entities are Trans no, Amount and type.



ER Diagram

**Geometric shapes and their meaning:**

**Rectangle**: Represents Entity sets.  
**Ellipses**: Attributes  
**Diamonds**: Relationship Set  
**Lines**: They link attributes to Entity Sets and Entity sets to Relationship Set  
**Double Ellipses:** Multivalued Attributes  
**Dashed Ellipses**: Derived Attributes  
**Double Rectangles**: Weak Entity Sets  
**Double Lines**: Total participation of an entity in a relationship set.

## Components of a ER Diagram

An ER diagram has three main components:  
1. Entity  
2. Attribute  
3. Relationship

### **Entity**

An entity is an object or component of data. An entity is represented as rectangle in an ER diagram.

**Weak Entity:**  
An entity that cannot be uniquely identified by its own attributes and relies on the relationship with other entity is called weak entity. The weak entity is represented by a double rectangle.

### **Attribute**

An attribute describes the property of an entity. An attribute is represented as Oval in an ER diagram. There are four types of attributes:

1. Key attribute  
2. Composite attribute  
3. Multivalued attribute  
4. Derived attribute

#### 1. Key attribute:

A key attribute can uniquely identify an entity from an entity set. For example, student roll number can uniquely identify a student from a set of students

Key attribute is represented by oval same as other attributes however the **text of key attribute is underlined**.

#### 2. Composite attribute:

An attribute that is a combination of other attributes is known as composite attribute. For example, In student entity, the student address is a composite attribute as an address is composed of other attributes such as pin code, state, country.

#### 3. Multi valued attribute:

An attribute that can hold multiple values is known as multi valued attribute. It is represented with **double ovals** in an ER Diagram. For example – A person can have more than one phone numbers so the phone number attribute is multi valued.

#### 4. Derived attribute:

### A derived attribute is one whose value is dynamic and derived from another attribute. It is represented by **dashed oval** in an ER Diagram. For example – Person age is a derived attribute as it changes over time and can be derived from another

### **Relationship**

A relationship is represented by diamond shape in ER diagram, it shows the relationship among entities. There are four types of relationships/cardinality ratio:  
1. One to One  
2. One to Many  
3. Many to One  
4. Many to Many

#### 1. One to One Relationship

When a single instance of an entity is associated with a single instance of another entity then it is called one to one relationship. For example, a person has only one passport and a passport is given to one person.

#### 2. One to Many Relationship

When a single instance of an entity is associated with more than one instances of another entity then it is called one to many relationship. For example – a customer can place many orders but a order cannot be placed by many customers.

#### 3. Many to One Relationship

When more than one instances of an entity is associated with a single instance of another entity then it is called many to one relationship. For example – many students can study in a single college but a student cannot study in many colleges at the same time.

#### 4. Many to Many Relationship

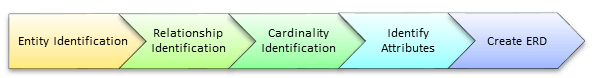
When more than one instances of an entity is associated with more than one instances of another entity then it is called many to many relationship. For example, a can be assigned to many projects and a project can be assigned to many students.

### **Participation Constraints**

* **Total Participation** − Each entity is involved in the relationship. Total participation is represented by double lines.
* **Partial participation** − Not all entities are involved in the relationship. Partial participation is represented by single lines.

**How to Create an Entity Relationship Diagram (ERD)**

Following are the steps to create an ER Diagram:



Step 1) Entity Identification

Step 2) Relationship Identification

Step 3): Cardinality Identification:

Step 4) Identify Attributes

**Conclusion**: We have studied and created a database with at least 3 entities and relationships between them with creation of ER/EER diagram for the system

**Assignment No: 2**

**Aim:-** Design and implement a database (for assignment no 1) using DDL statements and apply normalization on them

**RDBMS:**

A relational database refers to a database that stores data in a structured format, using rows and columns. This makes it easy to locate and access specific values within the database. It is "relational" because the values within each table are related to each other. Tables may also be related to other tables. The relational structure makes it possible to run queries across multiple tables at once.

**RDBMS Concepts:**

Before we proceed to explain MySQL database system, let's revise few definitions related to database.

* Database: A database is a collection of tables, with related data.
* Table: A table is a matrix with data. A table in a database looks like a simple spread sheet.
* Column: One column (data element) contains data of one and the same kind, for example the Column post code.
* Row: A row (tuple, entry or record) is a group of related data, for example the data of one subscription.
* Redundancy: Storing data twice, redundantly to make the system faster.
* Primary Key: A primary key is unique. A key value cannot occur twice in one table. With a key, you can find at most one row.
* Foreign Key: A foreign key is the linking pin between two tables.
* Compound Key: A compound key(composite key) is a key that consists of multiple columns, Because one column is not sufficiently unique.

**Structured Query Language(SQL):**

* SQL is a database computer language designed for the retrieval and management of data in a relational database. SQL stands for Structured Query Language.
* SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database.
* SQL is the standard language for Relational Database System. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.
* SQL uses certain commands like Create, Drop, Insert, etc. to carry out the required tasks.

These SQL commands are mainly categorized into four categories as:

* DDL – Data Definition Language
* DQl – Data Query Language
* DML – Data Manipulation Language
* DCL – Data Control Language

**Data Definition Language:**

Data Definition Language helps you to define the database structure or schema. Let's learn about DDL commands with syntax.

Five types of DDL commands in SQL are:

* CREATE
* INSERT
* UPDATE
* ALTER
* DROP
* DELETE

**CREATE**:

Create statements is used to define the database structure schema:

**Syntax:**

CREATE TABLE TABLE\_NAME (COLUMN\_NAME DATATYPES [,....]);

**Example**:

CREATE TABLE EMPLOYEE (Name VARCHAR2(20), Email VARCHAR2(100), DOB DATE);

**INSERT:**

The INSERT statement is a SQL query. It is used to insert data into the row of a table.

**Syntax:**

INSERT INTO TABLE\_NAME

(column1, column2, column3 ,.... columnN)

VALUES (value1, value2, value3 , .... valueN);

Or

INSERT INTO TABLE\_NAME

VALUES (value1, value2, value3 , .... valueN);

**Example:**

INSERT INTO employee (Name, Salary) VALUES ("Meghana", 20000);

**UPDATE:**

This command is use d to update or modify the value of a column in the table.

**Syntax:**

UPDATE table\_name SET [column\_name1=value1,…,column\_nameN=valueN]

[WHERE CONDITION]

Example:

UPDATE students

SET User\_Name = 'Sonoo'

WHERE Student\_Id = '3'

**ALTER**:

**Syntax to add new column:**

ALTER TABLE table\_name ADD column\_namedatatype;

**Syntax to drop column:**

ALTER TABLE table\_name DROP COLUMN column\_name;

**Syntax to change datatype:**

ALTER TABLE table\_name MODIFY COLUMN column\_namedatatype;

**Example:**

ALTER TABLE CUSTOMERS ADD Address Varchar(20);

ALTER TABLE CUSTOMERS DROP Address;

**DROP**

It is used to delete both the structure and record stored in the table.

**Syntax:**

DROP TABLE ;

**Example:**

DROP TABLE EMPLOYEE;

**DELETE**

This command is used to remove one or more rows from a table.

**Syntax:**

DELETE FROM table\_name [WHERE condition];

**Example:**

DELETE FROM EMPLOYEE

WHERE EMP\_NAME = 'Kristen';

**Normalization:**

* Normalization is the process of organizing the data in the database.
* Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.
* Normalization divides the larger table into the smaller table and links them using relationship.
* The normal form is used to reduce redundancy from the database table.

**Types of Normalization:**

**First Normal Form(1NF)**:

A relation is in 1NF if it contains an atomic value.

An attribute (column) of a table cannot hold multiple values. It should hold only atomic values.

**// Give your database example here for 1NF**

**Second Normal Form(2NF):**

A table is said to be in 2NF if both the following conditions hold:

* Table is in 1NF (First normal form)
* No non-prime attribute is dependent on the proper subset of any candidate key of table.

An attribute that is not part of any candidate key is known as non-prime attribute.

**// Give your database example here for 2NF**

**Third Normal Form(3NF):**

A table design is said to be in 3NF if both the following conditions hold:

* Table must be in 2NF
* [Transitive functional dependency](https://beginnersbook.com/2015/04/transitive-dependency-in-dbms/) of non-prime attribute on any super key should be removed.

An attribute that is not part of any [candidate key](https://beginnersbook.com/2015/04/candidate-key-in-dbms/) is known as non-prime attribute.

**// Give your database example here for 3NF**

**Boyce Codd normal form (BCNF)**

* It is an advance version of 3NF that’s why it is also referred as 3.5NF.
* BCNF is stricter than 3NF.
* A table complies with BCNF if it is in 3NF and for every [functional dependency](https://beginnersbook.com/2015/04/functional-dependency-in-dbms/) X->Y, X should be the super key of the table.

**// Give your database example here for BCNF**

**// Output Screenshot of all DDL Queries in MySQL.**

**Conclusion:** In this way we have implemented and created database using DDL Statements with the normalization technique to bring the database into normalized state.

**Assignment No. 3**

**Aim:** Create Table with primary key and foreign key constraints.

a. Alter table with add n modify b. Drop table

**Primary key:**

The PRIMARY KEY constraint uniquely identifies each record in a table.

Primary keys must contain UNIQUE values, and cannot contain NULL values.

A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

CREATE TABLE Persons (  
    Person\_IDint NOT NULL,  
    LastName var char(255) NOT NULL,  
    FirstName var char(255),  
    Age int,  
    PRIMARY KEY (ID)  
);

To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns, use the following SQL syntax:

CREATE TABLE Persons (  
    Person\_IDint NOT NULL,  
    LastName var char(255) NOT NULL,  
    FirstName var char(255),  
    Age int,  
    CONSTRAINT PK\_Person PRIMARY KEY (Person\_ID,LastName)  
);

To create a PRIMARY KEY constraint on the "ID" column when the table is already created, use the following SQL:

ALTER TABLE Persons  
ADD PRIMARY KEY (ID);

To drop a PRIMARY KEY constraint, use the following SQL:

ALTER TABLE Persons  
DROP PRIMARY KEY;

or

ALTER TABLE Persons  
DROP CONSTRAINT PK\_Person;

**Foreign Key:**

* The Foreign key constraint is used to prevent actions that would destroy links between tables.
* A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the [PRIMARY KEY](https://www.w3schools.com/sql/sql_primarykey.asp) in another table.
* The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.
* Notice that the "PersonID" column in the "Orders" table points to the "PersonID" column in the "Persons" table.
* The "PersonID" column in the "Persons" table is the PRIMARY KEY in the "Persons" table.
* The "Person ID" column in the "Orders" table is a FOREIGN KEY in the "Orders" table.

The FOREIGN KEY constraint prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the parent table.

CREATE TABLE Orders (  
    Order IDint NOT NULL,  
    Order Numberint NOT NULL,  
    Person IDint,  
    PRIMARY KEY (Order ID),  
    FOREIGN KEY (Person ID) REFERENCES Persons(Person ID)  
);

To allow naming of a FOREIGN KEY constraint, and for defining a FOREIGN KEY constraint on multiple columns, use the following SQL syntax:

CREATE TABLE Orders (  
    OrderIDint NOT NULL,  
    OrderNumberint NOT NULL,  
    PersonIDint,  
    PRIMARY KEY (OrderID),  
    CONSTRAINT FK\_PersonOrder FOREIGN KEY (PersonID)  
    REFERENCES Persons(Person\_ID)  
);

To create a FOREIGN KEY constraint on the "PersonID" column when the "Orders" table is already created, use the following SQL:

ALTER TABLE Orders  
ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);

To drop a FOREIGN KEY constraint, use the following SQL:

ALTER TABLE Orders  
DROP FOREIGN KEY FK\_PersonOrder;

**// Screenshot of all your sql queries having primary and foreign key constraint by using all above syntax**

**Conclusion:** In this assignment, we have successfully created table with primary key and foreign key constraints.

**Assignment No: 4**

**Aim:** Perform following SQL queries on the database created in assignment 1.

• Implementation of relational operators in SQL

• Boolean operators and pattern matching

• Arithmetic operations and built in functions

• Group functions

• Processing Date and Time functions

• Complex queries and set operators

**Operator in SQL:**

SQL stands for Structured Query Language. SQL lets you access and manipulate databases. An operator is a reserved word or a character used primarily in an SQL statement's WHERE clause to perform operation(s), such as comparisons and arithmetic operations. These Operators are used to specify conditions in an SQL statement and to serve as conjunctions for multiple conditions in a statement.

**Relational Operator:**

Relational operators are used for comparing numbers and strings. If a string is compared to a number, MySQL will try to convert the string to a number.

Relational operators in SQL are as follows:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Called as** | **Example query** |
| = | Equals to | SELECT \* FROM ProductsWHERE Price = 18; |
| > | Greater than | SELECT \* FROM Products WHERE Price > 18; |
| < | Less than | SELECT \* FROM Products WHERE Price < 18; |
| >= | Greater than or equal to | SELECT \* FROM Products WHERE Price >= 18; |
| <= | Less than or equal to | SELECT \* FROM Products WHERE Price <= 18; |
| <> | Not equal to | SELECT \* FROM Products WHERE Price <> 18; |

**// give any 3 example query for arithematic operator related to assignment no 1(Screenshot of query with output)**

**Boolean Operator:**

Following is the list of Boolean operator along with example query:

|  |  |
| --- | --- |
| **Operator** | **Example query** |
| AND | SELECT c1,c2 FROM tbl1 WHERE c1= x AND c2=y |
| OR | SELECT c1,c2 FROM tbl1 WHERE c1= x OR c2=y |
| NOT | SELECT c1 FROM tbl1 WHERE NOT c1<100 |
| ANY | SELECT column\_name(s) FROM table\_name WHERE column\_name operator ANY   (SELECT column\_name   FROM table\_name   WHERE condition); |
| ALL | SELECT column\_name(s) FROM table\_name WHERE column\_name operator ALL (SELECT column\_name   FROM table\_name   WHERE condition); |
| EXISTS | SELECT column\_name(s) FROM table\_name WHERE EXISTS (SELECT column\_name FROM table\_name WHERE condition); |
| IN | SELECT column\_name(s) FROM table\_name WHERE column\_name IN (value1, value2, ...); |
| BETWEEN | SELECT column\_name(s) FROM table\_name WHERE column\_name BETWEEN value1 AND value2; |

**// give example of any 3 booleansql query for Boolean operator (Screenshot of query with output)**

**Pattern matching:**

SQL pattern matching enables you to use \_ to match any single character and % to match an arbitrary number of characters (including zero characters). In MySQL, SQL patterns are case-insensitive by default. Some examples are shown here. Do not use = or <> when you use SQL patterns.

Example Query- SELECT\*FROM tbl1 WHEREnameLIKE'b%';

**// give one example of pattern matching (Screenshot of query and output)**

**Arithmetic operations and built in functions**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Called as** | **Example query** |
| + | Add | SELECT 10 + 20; |
| - | Subtract | SELECT 10 - 20; |
| \* | Multiply | SELECT 10 \* 20; |
| / | Divide | SELECT 10 /20; |
| % | Modulo | SELECT 10 %20; |

**// give any 2 example query of arithematic operator**

**Built in Functions:**

**Numeric Functions:**

|  |  |  |
| --- | --- | --- |
| **Function** | **Input Argument** | **Value Returned** |
| ABS ( m ) | m = value | Absolute value of m |
| MOD ( m, n ) | m = value, n = divisor | Remainder of m divided by n |
| POWER ( m, n ) | m = value, n = exponent | m raised to the nth power |
| ROUND ( m [, n ] ) | m = value, n = number of decimal places, default 0 | m rounded to the nth decimal place |
| TRUNC ( m [, n ] ) | m = value, n = number of decimal places, default 0 | m truncated to the nth decimal place |
| SQRT ( n ) | n = value | positive square root of n |
| EXP ( n ) | n = value | e raised to the power n |
| LN ( n ) | n > 0 | natural logarithm of n |
| LOG ( n2, n1 ) | base n2 any positive value other than 0 or 1, n1 any positive value | logarithm of n1, base n2 |
| CEIL ( n ) | n = value | smallest integer greater than or equal to n |
| FLOOR ( n ) | n = value | greatest integer smaller than or equal to n |
| SIGN ( n ) | n = value | -1 if n < 0, 0 if n = 0, and 1 if n > 0 |

**String Functions:**

|  |  |  |
| --- | --- | --- |
| **Function** | **Input Argument** | **Value Returned** |
| INITCAP ( s ) | s = character string | First letter of each word is changed to uppercase and all other letters are in lower case. |
| LOWER ( s ) | s = character string | All letters are changed to lowercase. |
| UPPER ( s ) | s = character string | All letters are changed to uppercase. |
| CONCAT ( s1, s2 ) | s1 and s2 are character strings | Concatenation of s1 and s2. Equivalent to *s1 || s2* |
| LPAD ( s1, n [, s2] ) | s1 and s2 are character strings and n is an integer value | Returns s1 right justified and padded left with n characters from s2; s2 defaults to space. |
| RPAD ( s1, n [, s2] ) | s1 and s2 are character strings and n is an integer value | Returns s1 left justified and padded right with n characters from s2; s2 defaults to space. |
| LTRIM ( s [, set ] ) | s is a character string and *set*is a set of characters | Returns s with characters removed up to the first character not in set; defaults to space |
| RTRIM ( s [, set ] ) | s is a character string and *set*is a set of characters | Returns s with final characters removed after the last character not in set; defaults to space |
| REPLACE ( s, search\_s [, replace\_s ] ) | s = character string, search\_s = target string, replace\_s = replacement string | Returns s with every occurrence of search\_s in s replaced by replace\_s; default removes search\_s |
| SUBSTR ( s, m [, n ] ) | s = character string, m = beginning position, n = number of characters | Returns a substring from s, beginning in position m and n characters long; default returns to end of s. |
| LENGTH ( s ) | s = character string | Returns the number of characters in s. |
| INSTR ( s1, s2 [, m [, n ] ] ) | s1 and s2 are character strings, m = beginning position, n = occurrence of s2 in s1 | Returns the position of the nth occurrence of s2 in s1, beginning at position m, both m and n default to 1. |

**// give any 3 example of built in functions.**

**Group functions**

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUPBY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)  
ORDER BY column\_name(s);

**// give any 2 example of group fuctions**

**Processing Date and Time functions:**

Date and time functions are scalar functions that perform an operation on a date and time input value and returns either a string, numeric, or date and time value.

|  |  |  |
| --- | --- | --- |
| **Function** | **Syntax** | **Description** |
| SYSDATETIME | SYSDATETIME() | Returns a datetime value that contains the date and time |
| CURRENT\_TIMESTAMP | CURRENT\_TIMESTAMP | Returns a datetime value that contains date and time. |
| DATEPART | DATEPART(datepart,date) | Returns an integer that represents the specified datepart of the specified date. |
| DAY/MONTH/YEAR | DAY(date)/MONTH(date)/YEAR(date) | Returns an integer that represents the day/month/year part of the specified date. |
| DATEDIFF | DATEDIFF(datepart, startdate, enddate) | Returns the number of date and time datepart boundaries that are crossed between two specified date. |
| DATEADD | DATEADD(datepart, number, date) | Returns a new datetime value by adding an interval to the specified datepart of the specified date. |

**// execute any 3 date and time commands and put here that query example with output**

**Complex queries and set operators**

**Complex queries:**

Queries such as find duplicate row, second highest salary, fetch first or last record form table, display first five rows are complex queries.

**// Give two example of complex query here**

**Set Operator:**

Set operators are used to join the results of two (or more) SELECT statements.The SET operators are UNION,UNION ALL,INTERSECT,andMINUS.

## UNION Operation

The UNION operator is used to combine the result-set of two or more SELECT statements.

* Every SELECT statement within UNION must have the same number of columns
* The columns must also have similar data types
* The columns in every SELECT statement must also be in the same order

## Syntax:

## SELECTcolumn\_name(s)FROM table1UNIONSELECTcolumn\_name(s)FROM table2;

## INTERSECT

The SQL INTERSECT clause/operator is used to combine two SELECT statements, but returns rows only from the first SELECT statement that are identical to a row in the second SELECT statement. This means INTERSECT returns only common rows returned by the two SELECT statements.

Just as with the UNION operator, the same rules apply when using the INTERSECT operator. MySQL does not support the INTERSECT operator.

## Syntax

The basic syntax of INTERSECT is as follows.

SELECT column1 [, column2 ]

FROM table1 [, table2 ]

[WHERE condition]

INTERSECT

SELECT column1 [, column2 ]

FROM table1 [, table2 ]

[WHERE condition]

## MINUS

The Minus Operator in SQL is used with two SELECT statements. The MINUS operator is used to subtract the result set obtained by first SELECT query from the result set obtained by second SELECT query. In simple words, we can say that MINUS operator will return only those rows which are unique in only first SELECT query and not those rows which are common to both first and second SELECT queries.

**Basic Syntax**:

SELECT column1 , column2 , ... columnN

FROM table\_name

WHERE condition

MINUS

SELECT column1 , column2 , ... columnN

FROM table\_name

WHERE condition;

**// Give 1 example of SET Operator**

Conclusion: In this way, we have studied and implemented different SQL operator on sql query for our database.

**Assignment No: 5**

**Aim:** Execute DDL/DML statements which demonstrate the use of views. Update the base table using its corresponding view. Also consider restrictions on updatable views and perform view creation from multiple tables.

**VIEW:**

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.

A view is created with the CREATE VIEW statement.

CREATE VIEW view\_nameAS  
SELECT column1*,*column2*,* ...  
FROM table\_name  
WHERE condition*;*

SQL REPLACE VIEW Syntax

REPLACE VIEW view\_nameAS  
SELECT column1*,*column2, ...  
FROM table\_name  
WHERE condition*;*

A view is deleted with the DROP VIEW statement.

DROP VIEW view\_name*;*

**Updatable Views**

Some views are updatable and references to them can be used to specify tables to be updated in data change statements. That is, you can use them in statements such as [UPDATE](https://dev.mysql.com/doc/refman/8.0/en/update.html), [DELETE](https://dev.mysql.com/doc/refman/8.0/en/delete.html), or [INSERT](https://dev.mysql.com/doc/refman/8.0/en/insert.html) to update the contents of the underlying table.

A view is not updatable if it contains any of the following:

1. The SELECT statement which is used to create the view should not include GROUP BY clause or ORDER BY clause.
2. The SELECT statement should not have the DISTINCT keyword.
3. The View should have all NOT NULL values.
4. The view should not be created using nested queries or complex queries.
5. The view should be created from a single table. If the view is created using multiple tables then we will not be allowed to update the view.

* It is not possible to create an index on a view.
* Indexes can be used for views processed using the merge algorithm. However, a view that is processed with the temptable algorithm is unable to take advantage of indexes on its underlying tables (although indexes can be used during generation of the temporary tables).

**View creation from multiple tables**

To create a View from multiple tables we can simply include multiple tables in the SELECT statement.

CREATE VIEW v AS

SELECT tbl1.NAME, tbl1.ADDRESS, tbl2.MARKS

FROM tbl1, tbl2

WHERE tbl1.id = tbl2.id;

**// Give Your Output Here**

**Conclusion:** In this way we have demonstrated the use of views with view creation from multiple tables.

# Group C: PL/SQL

# Assignment No :- 1

**Aim :-** . Write and execute PL/SQL stored procedure and function to perform a suitable task on the database. Demonstrate its use.

Objective:

* To study and implement PL/SQL procedure.
* To study and implement PL/SQL function.

**Theory :** A subprogram is a program unit/module that performs a particular task. These subprograms are combined to form larger programs. This is basically called the 'Modular design'. A subprogram can be invoked by another subprogram or program which is called the calling program.

A subprogram can be created –

* At the schema level
* Inside a package
* Inside a PL/SQL block

At the schema level, subprogram is a standalone subprogram. It is created with the CREATE PROCEDURE or the CREATE FUNCTION statement. It is stored in the database and can be deleted with the DROP PROCEDURE or DROP FUNCTION statement.

PL/SQL subprograms are named PL/SQL blocks that can be invoked with a set of parameters. PL/SQL provides two kinds of subprograms –

* **Functions −**these subprograms return a single value; mainly used to compute and return a value.
* **Procedures −**these subprograms do not return a value directly; mainly used to perform an action.

Parts of a PL/SQL Subprogram- Each PL/SQL subprogram has a name, and may also have a parameter list. Like anonymous PL/SQL blocks, the named blocks will also have the following three parts –

1. **Declarative Part -** It is an optional part. However, the declarative part for a subprogram does not start with the DECLARE keyword. It contains declarations of types, cursors, constants, variables, exceptions, and nested subprograms. These items are local to the subprogram and cease to exist when the subprogram completes execution.
2. **Executable Part -** This is a mandatory part and contains statements that perform the designated action.
3. **Exception -** handling this is again an optional part. It contains the code that handles run-time errors

**Creating a Procedure**

A procedure is created with the CREATE OR REPLACE PROCEDURE statement. The simplified syntax for the CREATE OR REPLACE PROCEDURE statement is as follows –

CREATE [OR REPLACE] PROCEDURE procedure\_name

[(parameter\_name [IN | OUT | IN OUT] type [, ...])]

{IS | AS}

BEGIN

<procedure\_body>

END procedure\_name;

Where, • procedure-name specifies the name of the procedure.

* [OR REPLACE] option allows the modification of an existing procedure.
* The optional parameter list contains name, mode and types of the parameters. IN represents the value that will be passed from outside and OUT represents the parameter that will be used to return a value outside of the procedure.
* procedure-body contains the executable part.
* The AS keyword is used instead of the IS keyword for creating a standalone procedure.

**Example**

The following example creates a simple procedure that displays the string 'Hello World!' on the screen when executed.

CREATE OR REPLACE PROCEDURE greetings AS BEGIN dbms\_output.put\_line('Hello World!'); END; /

**Deleting a Standalone Procedure**

A standalone procedure is deleted with the DROP PROCEDURE statement. Syntax for deleting a procedure is –

DROP PROCEDURE procedure-name;

You can drop the greetings procedure by using the following statement –

DROP PROCEDURE greetings;

Parameter Modes in PL/SQL Subprograms

The following table lists out the parameter modes in PL/SQL subprograms –

**IN**

An IN parameter lets you pass a value to the subprogram. It is a read-only parameter. Inside the subprogram, an IN parameter acts like a constant. It cannot be assigned a value. You can pass a constant, literal, initialized variable, or expression as an IN parameter. You can also initialize it to a default value; however, in that case, it is omitted from the subprogram call. It is the default mode of parameter passing. Parameters are passed by reference.

**OUT**

An OUT parameter returns a value to the calling program. Inside the subprogram, an OUT parameter acts like a variable. You can change its value and reference the value after assigning it. The actual parameter must be variable and it is passed by value.

**IN OUT**

An IN OUT parameter passes an initial value to a subprogram and returns an updated value to the caller. It can be assigned a value and the value can be read. The actual parameter corresponding to an IN OUT formal parameter must be a variable, not a constant or an expression. Formal parameter must be assigned a value. Actual parameter is passed by value.

**// provide other example queries and outputs (screenshots) for procedure taken into lab session**

**Functions:**

A function is same as a procedure except that it returns a value. Therefore, all the discussions of the previous chapter are true for functions too.

**Creating a Function**

A standalone function is created using the CREATE FUNCTION statement. The simplified syntax for the CREATE OR REPLACE PROCEDURE statement is as follows –

CREATE [OR REPLACE] FUNCTION function\_name

[(parameter\_name [IN | OUT | IN OUT] type [, ...])]

RETURN return\_datatype

{IS | AS}

BEGIN

<function\_body>

END [function\_name];

Where,

* function-name specifies the name of the function.
* [OR REPLACE] option allows the modification of an existing function.
* The optional parameter list contains name, mode and types of the parameters. IN represents the value that will be passed from outside and OUT represents the parameter that will be used to return a value outside of the procedure.
* The function must contain a return statement.
* The RETURN clause specifies the data type you are going to return from the function.
* function-body contains the executable part.
* The AS keyword is used instead of the IS keyword for creating a standalone function.

**//provide other example queries and outputs(screenshots) taken into lab session**

**Conclusion:-** We have studied and executed PL/SQL stored procedures and functions.

# Assignment No :- 2

**Aim :-**Write and execute suitable database triggers .Consider row level and statement level triggers.

## Objective:

* To study and implement PL/SQL triggers.

**Theory :**

Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events.

* A database manipulation (DML) statement (DELETE, INSERT, or UPDATE)
* A database definition (DDL) statement (CREATE, ALTER, or DROP).
* A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers can be defined on the table, view, schema, or database with which the event is associated.

**Benefits of Triggers**

Triggers can be written for the following purposes –

* Generating some derived column values automatically
* Enforcing referential integrity
* Event logging and storing information on table access
* Auditing
* Synchronous replication of tables
* Imposing security authorizations
* Preventing invalid transactions **Creating Triggers**

The syntax for creating a trigger is –

CREATE [OR REPLACE ] TRIGGER trigger\_name

{BEFORE | AFTER | INSTEAD OF }

{INSERT [OR] | UPDATE [OR] | DELETE}

[OF col\_name]

ON table\_name

[REFERENCING OLD AS o NEW AS n]

[FOR EACH ROW]

WHEN (condition)

DECLARE Declaration-statements BEGIN Executable-statements

EXCEPTION Exception-handling-statements END;

Where,

* CREATE [OR REPLACE] TRIGGER trigger\_name − Creates or replaces an existing trigger with the trigger\_name.
* {BEFORE | AFTER | INSTEAD OF} − This specifies when the trigger will be executed. The INSTEAD OF clause is used for creating trigger on a view.
* {INSERT [OR] | UPDATE [OR] | DELETE} − This specifies the DML operation.
* [OF col\_name] − This specifies the column name that will be updated.
* [ON table\_name] − This specifies the name of the table associated with the trigger.
* [REFERENCING OLD AS o NEW AS n] − This allows you to refer new and old values for various DML statements, such as INSERT, UPDATE, and DELETE.
* [FOR EACH ROW] − This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
* WHEN (condition) − this provides a condition for rows for which the trigger would fire. This clause is valid only for row-level triggers.

**//provide other example queries and outputs(screenshots) taken into lab session**

**Conclusion:-** We have studied and executed different types of database triggers.

# Assignment No :- 3

**Aim :-**Write a PL/SQL block to implement all types of cursor.

**Objective:**

• To study and implement PL/SQL cursors

**Theory :**

A cursor is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the active set.

You can name a cursor so that it could be referred to in a program to fetch and process the rows returned by the SQL statement, one at a time. There are two types of cursors – Implicit cursors and Explicit cursors

**Implicit Cursors**

Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement. Programmers cannot control the implicit cursors and the information in it.

Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted. For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

In PL/SQL, you can refer to the most recent implicit cursor as the SQL cursor, which always has attributes such as %FOUND, %ISOPEN, %NOTFOUND, and %ROWCOUNT. The SQL cursor has additional attributes, %BULK\_ROWCOUNT and %BULK\_EXCEPTIONS, designed for use with the FORALL statement. The following table provides the description of the most used attributes –

**%FOUND**

Returns TRUE if an INSERT, UPDATE, or DELETE statement affected one or more rows or a SELECT INTO statement returned one or more rows. Otherwise, it returns FALSE

**%NOTFOUND**

The logical opposite of %FOUND. It returns TRUE if an INSERT, UPDATE, or DELETE statement affected no rows, or a SELECT INTO statement returned no rows. Otherwise, it returns FALSE.

**%ISOPEN**

Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement.

**%ROWCOUNT**

Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO statement.

**Explicit Cursors**

Explicit cursors are programmer-defined cursors for gaining more control over the context area. An explicit cursor should be defined in the declaration section of the

PL/SQL Block. It is created on a SELECT Statement which returns more than one row.

The syntax for creating an explicit cursor is –

CURSOR cursor\_name IS select\_statement;

Working with an explicit cursor includes the following steps –

* Declaring the cursor for initializing the memory
* Opening the cursor for allocating the memory
* Fetching the cursor for retrieving the data
* Closing the cursor to release the allocated memory

Declaring the Cursor

Declaring the cursor defines the cursor with a name and the associated SELECT statement. For example

CURSOR c\_customers IS SELECT id, name, address FROM customers;

**Opening the Cursor**

Opening the cursor allocates the memory for the cursor and makes it ready for fetching the rows returned by the SQL statement into it. For example, we will open the above defined cursor as follows –

OPEN c\_customers;

**Fetching the Cursor**

Fetching the cursor involves accessing one row at a time. For example, we will fetch rows from the above-opened cursor as follows –

FETCH c\_customers INTO c\_id, c\_name, c\_addr;

**Closing the Cursor**

Closing the cursor means releasing the allocated memory. For example, we will close the above-opened cursor as follows –

CLOSE c customers;

**//provide other example queries and outputs(screenshots) taken into lab session**

**Conclusion:-** We have studied and implemented cursors in PL/SQL.