MALLA REDDY UNIVERSITY
Telangana State Private Universities (Establishment and Regulations) (Amendment) Act No.13 of 2020
G.O.No.Ms.14, Higher Education (UE) Department, Telangana State
Maisammaguda, Kompally, Hyderabad – 500 100

MR20-1CS0181 - DATABASE MANAGEMENT SYSTEM LABORATORY

MANUAL & RECORD

B.TECH: II YEAR – I SEMESTER
(2022-2023)

SCHOOL OF ENGINEERING
(CSE)

MALLA REDDY UNIVERSITY

Telangana State Private Universities (Establishment and Regulations) (Amendment) Act No.13 of 2020
G.O.No.Ms.14, Higher Education (UE) Department, Telangana State
Maisammaguda, Kompally, Hyderabad – 500 100

LABORATORY MANUAL & RECORD

Name: .....

Roll No. .....

Year ..... Sem .....

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and
G.O.Ms.No.14, Higher Education (UE) Department)

Certificate

Certified that this is the bonafide record of the work done by

Mr./Ms..... Roll. No..... of

B.Tech ..... year ..... Semester for Academic year 20..... - 20..... in

..... Laboratory.

Date: Staff Incharge HOD

PROGRAM OUTCOMES (POs)

A **B.Tech** – Graduate should possess the following Program Outcomes.

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), those who come after 5 minutes will not be allowed into the lab.

2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.

3. Student should enter into the laboratory with:

a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.

b. Laboratory Record updated up to the last session experiments and other utensils (if any) needed in the lab.

c. Proper Dress code and Identity card.

4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.

5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.

6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.

7. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.

8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.

9. Students must take the permission of the faculty in case of any urgency to go out ; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.

10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

MR20-1CS0181 - DATABASE MANAGEMENT SYSTEM LABORATORY

Objectives:

- To understand the practical applicability of database management system concepts.
- Working on existing database systems, designing of database, creating relational database, analysis of table design.
- To engage themselves in lifelong learning of Database management systems theories and technologies this enables them to purse higher studies.

Course Outcomes:

- Students get practical knowledge on designing and creating relational database systems.
- Understand various advanced queries execution such as relational constraints, joins, set operations, aggregate functions, views etc.
- Students will be able to demonstrate their skill in drawing ER diagrams.
- Able to convert the entity-relationship diagrams into relational tables.
- To develop appropriate databases to given problem that integrates ethical, social, legal, and economic concerns.

NDEX

S. No	Name of the Experiment	Page No.	Date of Conduction	Date of Submission	Grade	Signature of Faculty
1	Installation of My MYSQL & Familiarization of E-R Diagrams					
2	Road-Way Travels E-R Diagrams					
3	DDL Commands					
4	DML Commands					
5	KEY Constraints					
6	Aggregate Functions and Mathematical Functions					
7	Nested Queries					
8	Correlated Queries					
9	Views					
10	Joins					
11	TCL Commands					
12	DCL Commands					

Exp. No: 1(WEEK-1)	INSTALLATION OF MYSQL

AIM: Installation of Mysql.

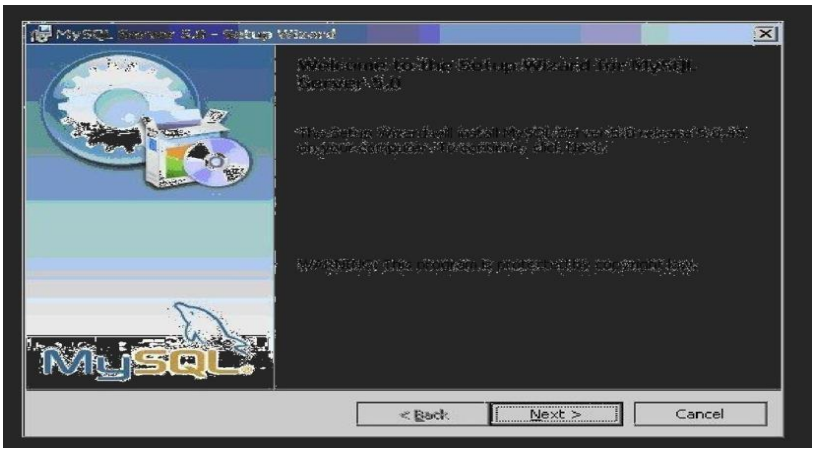
1. Steps for installing Mysql

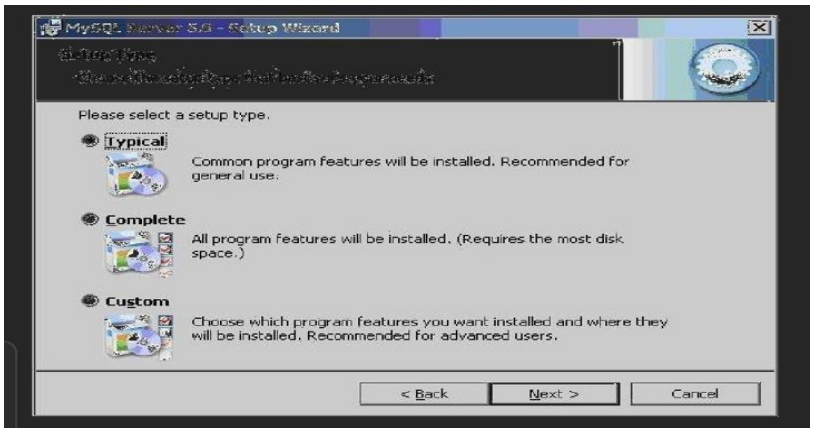
Step 1:

Make sure you already downloaded the **Mysql essential 5.0.45 win32.msi file**. Double click on the .msi file.

Step 2:

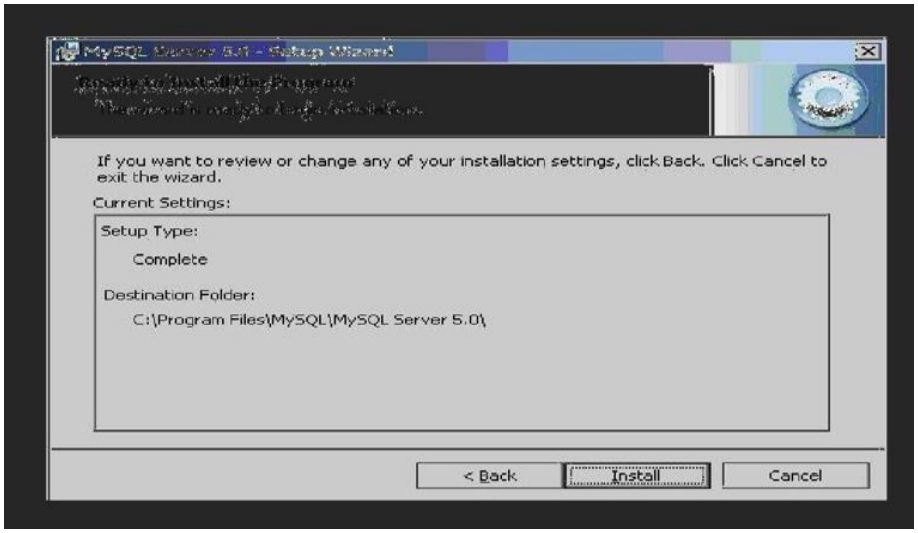
This is Mysql Server 5.0 setup wizard. The setup wizard will install Mysql Server 5.0release 5.0.45 on your computer. To continue, click **next.**





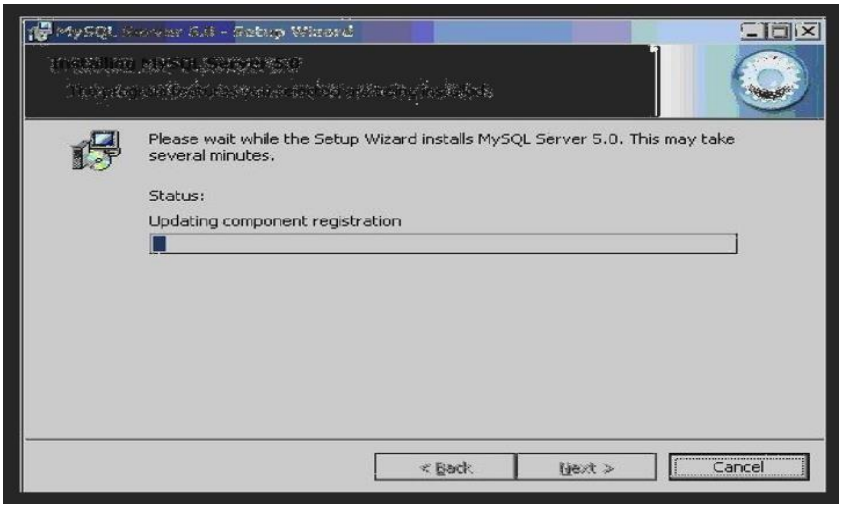
Step 3:

Choose the setup type that best suits your needs. For common program features select ***Typical*** and it's recommended for general use. To continue, click **next**.



Step 4:

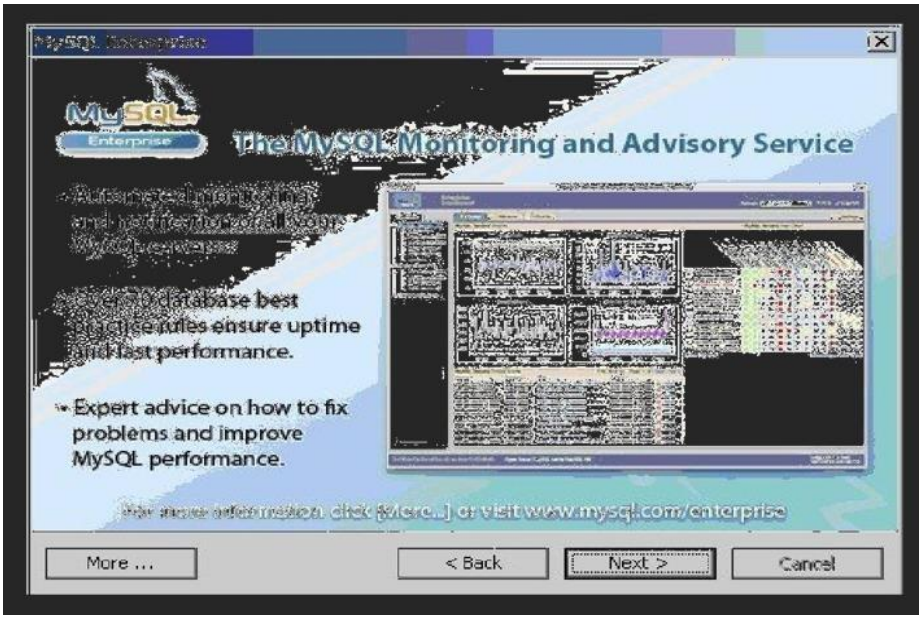
This wizard is ready to begin installation. Destination folder will be in **C:\ProgramFiles\Mysql\Mysql Server 5.0**. To continue, click **next**.



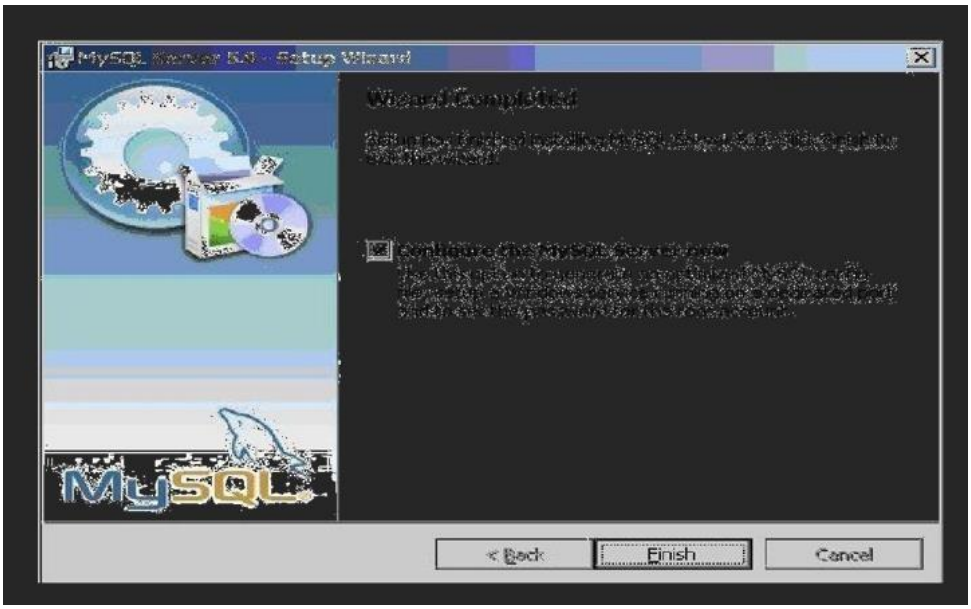
Step 5:

The program features you selected are being installed. Please wait while the setup wizard installs Mysql 5.0. This may take several minutes.

Step 6:
To continue, click **next**.

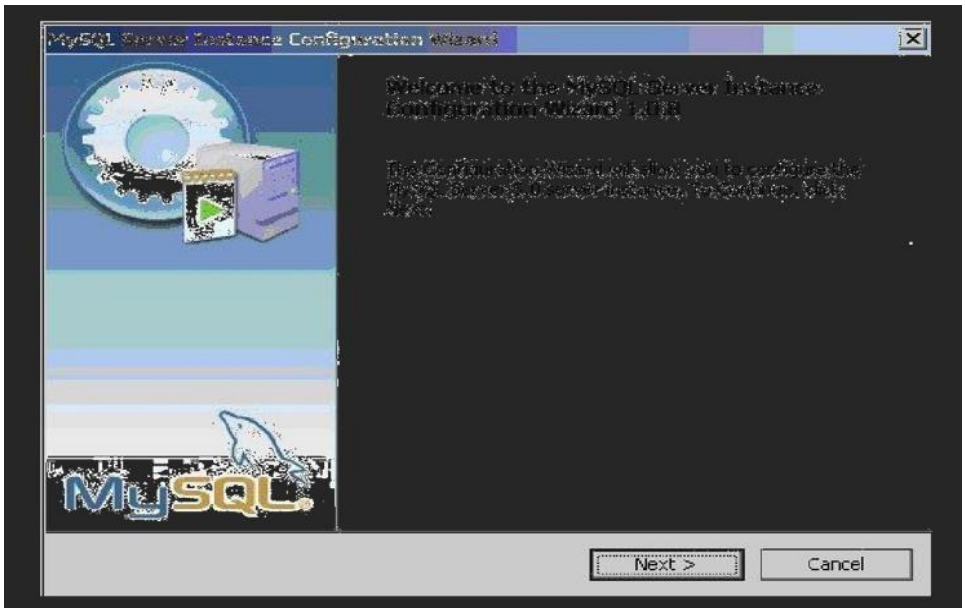


Step 7:
To continue, click **next**.



Step 8:

Wizard Completed. Setup has finished installing Mysql 5.0. **Check** the configure the Mysqlserver now to continue. Click **Finish** to exit the wizard



Step 9:

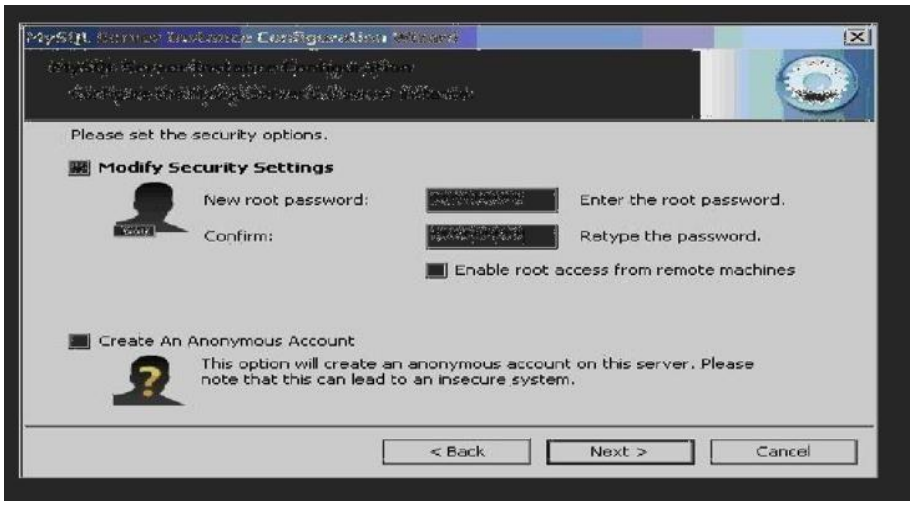
The configuration wizard will allow you to configure the Mysql Server 5.0 server instance. To continue, click **next**.

Step 10:

Select a **standard configuration** and this will use a general purpose configuration for theserver that can be tuned manually. To continue, click **next**.

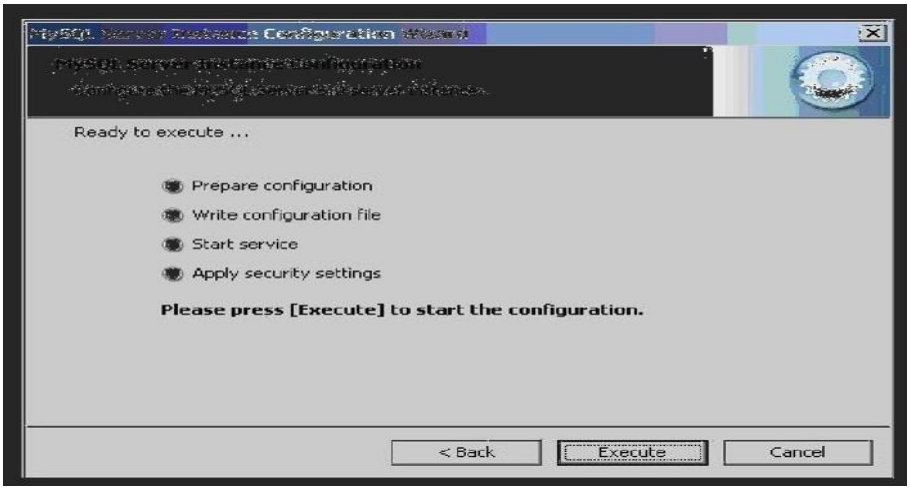
Step 11:

Check on the **install as windows service** and **include bin directory in windows path**. To continue, click **next**.



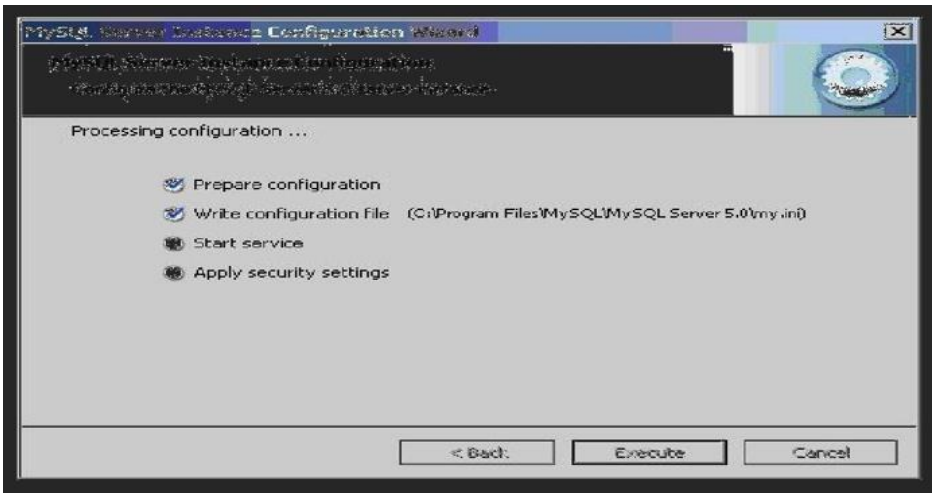
Step 12:

Please set the security options by entering the root password and confirm retype the password. Continue, click next



Step 13:

Ready to execute? Click **execute** to continue.



Step 14:

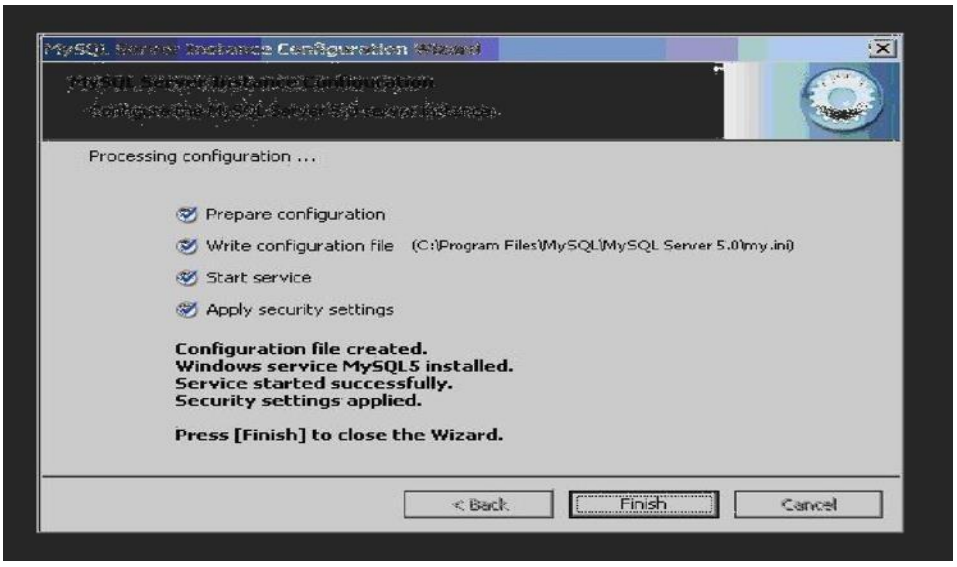
Processing configuration in progress

Step15

Configuration file created. Windows service Mysql5 installed.

Press **finish** to

close the wizard

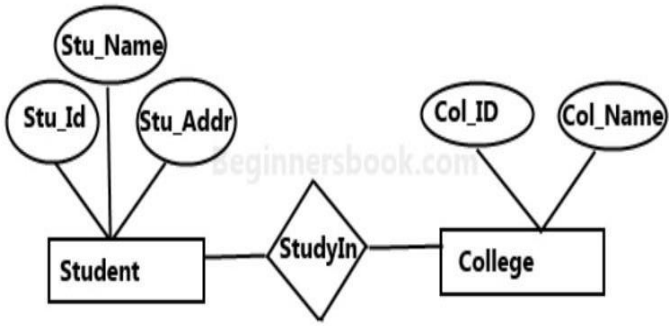


Entity Relationship Diagram – ER Diagram

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.

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

A simple ER Diagram:



In the following diagram we have two entities Student and College and their relationship. The relationship between Student and College is many to one as a college can have many students however a student cannot study in multiple colleges at the same time. Student entity has attributes such as Stu_Id, Stu_Name & Stu_Addr and College entity has attributes such as Col_ID & Col_Name.

Here are the geometric shapes and their meaning in an E-R Diagram. We will discuss these terms in detail in the next section(Components of a ER Diagram) of this guide so don't worry too much about these terms now, just go through them once.

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

Exp. No: 2(WEEK-2)	ROADWAY TRAVELS- ER DIAGRAM

ILLUSTRATION: ROADWAY TRAVELS

ER model is a logical representation of an enterprise data. ER model is a diagrammatic representation of logical structure of database. ER model describes relationship among entities and attributes.

"Roadway Travels" is in business since 1977 with several buses connecting different places in India. Its main office is located in Hyderabad. The company wants to computerize its operations in the following areas:

Reservations :

Reservations are directly handled by booking office. Reservations can be made 60 days in advance in either cash or credit. In case the ticket is not available, a wait listed ticket is issued to the customer. This ticket is confirmed against the cancellation.

Cancellation and modification:

Cancellations are also directly handed at the booking office. Cancellation charges will be charged. Wait listed tickets that do not get confirmed are fully refunded.

AIM: Analyze the problem and come with the entities in it. Identify what Data has to be persisted in the databases.

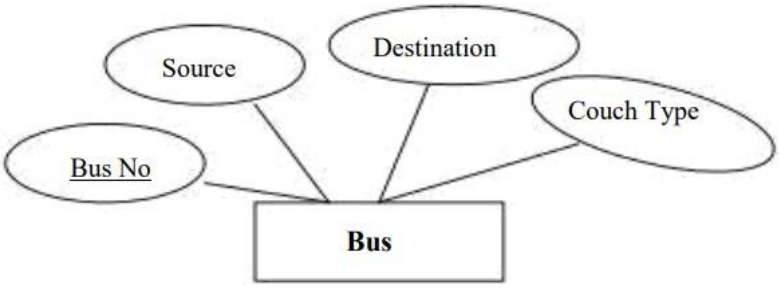
The Following are the entities:

1 .Bus

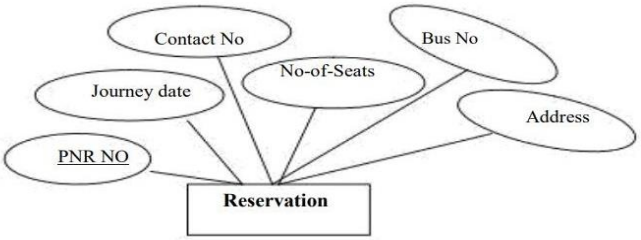
2. Reservation
3. Ticket
4. Passenger
5. Cancellation

The attributes in the Entities:

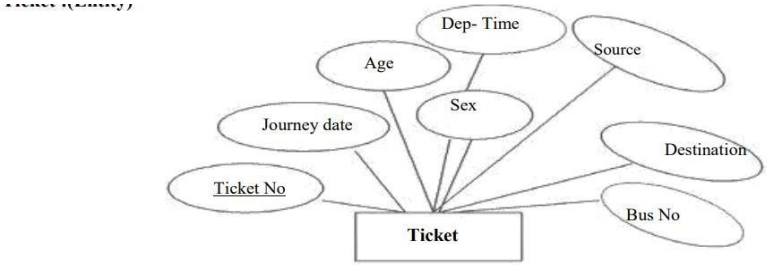
Bus :(Entity)



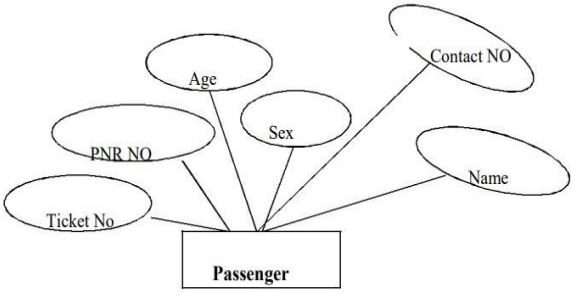
Reservation :(Entity)



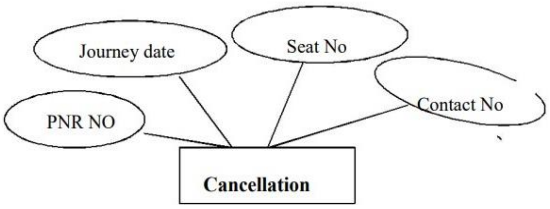
Ticket :(Entity)



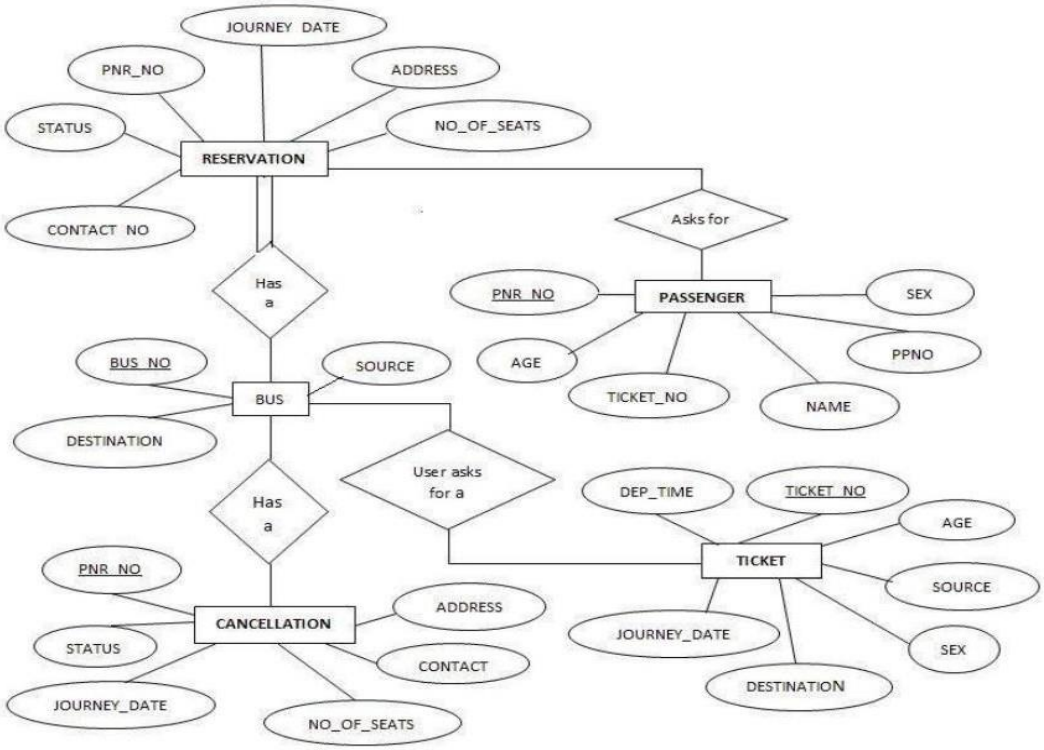
Passenger :(Entity)



Cancellation: (Entity)



Concept design with E-R Model:



Exercise 1:

Consider the following information about a university database:

- Professors have an SSN, a name, an age, a rank, and a research specialty.
- Projects have a project number, a sponsor name (e.g., NSF), a starting date,and ending date, and a budget.
- Graduate students have an SSN, a name, an age, and a degree program (e.g., M.S. or Ph.D.).
- Each project is managed by one professor (known as the project'sprincipal investigator).
- Each project is worked on by one or more professors (known as the project'sco-investigators).
- Professors can manage and/or work on multiple projects.
- Each project is worked on by one or more graduate students (known as the project's research assistants).
- When graduate students work on a project, a professor must supervise their work on the project. Graduate students can work on multiple projects, in whichcase they will have a (potentially different) supervisor for each one.
- Departments have a department number, a department name, and a main office.
- Departments have a professor (known as the chairman) who runs the department.
- Professor's work in one or more departments and for each department that they work in, a time percentage is associated with their job.
- Graduate students have one major department in which they are working ontheir degree.
- Each graduate student has another, more senior graduate student (knownas a student advisor) who advises him or her on what courses to take.

WORKSHEET

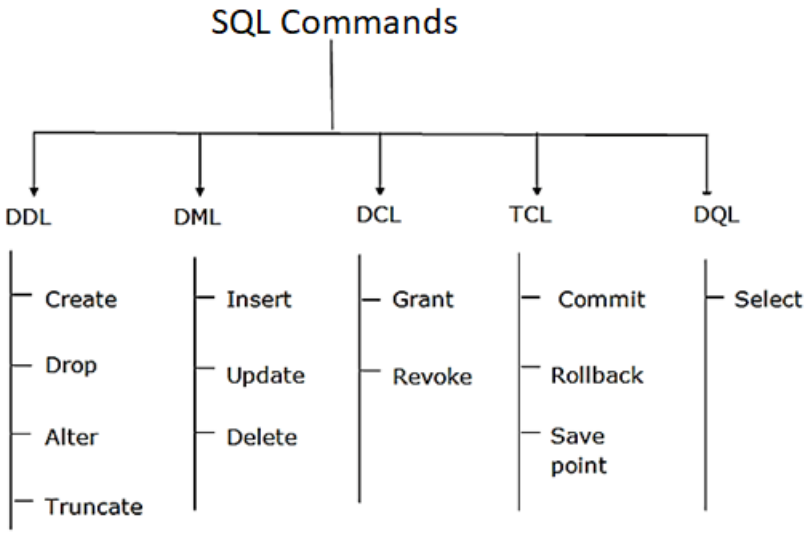
To communicate with Mysql supports the following categories of commands:

1. **Data Definition Language (DDL)**
   Create, Alter, Drop and Truncate

2. **Data Manipulation Language (DML)**
   Insert,Update, Delete and Select

3. **Transaction control language(TCL)**
   Commit,Rollback and Save point

4. **Data Control Language (DCL)**
   Grant and Revoke



Exp. No: 3(WEEK-3)	DDL COMMANDS

Data Definition Language (DDL) Statements

- DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
- All the command of DDL are auto-committed that means it permanently save all the changes in the database.

The Data Definition Languages used for table definition can be classified into following:

1. Create table command
    - Describe Command
2. Alter table command
3. Rename table command
4. Truncate table command
5. Drop table command

1. CREATE TABLE:

My MYSQL - CREATE TABLE:

Table is a primary object of database, used to store data in form of rows and columns. It is created using following command:

Syntax: CREATE TABLE tablename (column_name data_ type constraints, …)

```
mysql>CREATE TABLE SAILORS(
SID int(10) PRIMARY KEY,
SNAME VARCHAR (10),
RATING int (10),
AGE int (10));
```

Table Created.

- **Desc command**

The DESCRIBE command is used to view the structure of a table as follows.

```
mysql>DESC SAILORS;
```

TEST RESULT

Example 1: Create a RESERVES table with fields (SID, BID, DAY) and display using DESCRIBE command.

Example 2: Create a BOATS table with Fields (BID, BNAME, COLOR) and display using DESCRIBE command

2. ALTER TABLE:

(a) To ADD a column:

SYNTAX: ALTER TABLE <TABLE NAME>

ADD (

<NEW COLUMN NAME><DATA TYPE>(<SIZE>),

<NEW COLUMN NAME><DATATYPE>(<SIZE>));

```
mysql> ALTER TABLE SAILORS ADD (SNO INT (10));
```

TEST OUTPUT

(b) To DROP a column:

SYNTAX: ALTER TABLE <TABLE NAME>

DROP

<COLUMN NAME>;

```
mysql> ALTER TABLE SAILORS DROP SNO;
```

TEST OUTPUT

(c) To MODIFY a column:

SYNTAX: ALTER TABLE <TABLE NAME>

MODIFY(<COLUMN NAME>

<NEW DATATYPE>(<NEW SIZE>));

```
mysql> ALTER TABLE SAILORS MODIFY SNAME VARCHAR(20);
```

TEST OUTPUT

3. RENAME A TABLE

Rename command is used to give new names for existing tables.

mysql> **RENAME TABLE**

<old table name>

TO

<new table name>;

```
mysql> RENAME TABLE SAILORS TO SAILORS1;
mysql> RENAME TABLE SAILORS1 TO SAILORS;
```

TEST OUTPUT

4. TRUNCATE A TABLE

Truncate command is used to delete all records from a table.

mysql> TRUNCATE
TABLE <table name>;

```
mysql> TRUNCATE TABLE SAILORS;
```

TEST OUTPUT

5. DROP A TABLE

Drop command is used to remove an existing table permanently from database.

mysql> DROP TABLE <table name>;

```
mysql> DROP TABLE SAILORS;
```

TEST OUTPUT:

LAB EXERCISES

Apply all the DDL commands on the table.

1. Create a SALES table using fields Sales_Id, First_name, Last_Name, Dept_No
2. Create a PRODUCT table using fields Model, Cost, Colour.
3. Create a PURCHASE table with fields First_name, Last_Name, Id_No, Model, Sales_id.

Exp. No: 4(WEEK-4)	DML COMMANDS

Data Manipulation Language (DML) Statements

- DML commands are used to modify the database. It is responsible for all form of changes in the database.
- The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.

DML resembles simple English language and enhances efficient user interaction with the system. The functional capability of DML is organized in manipulation commands like SELECT, UPDATE, INSERT INTO and DELETE FROM, as described below:

1. **Inserting Data Into Table (INSERT)**

Insert command is used to insert rows into the table.

SYNTAX:
mysql> INSERT

INTO <tablename>

(columnname1, columnname2,….columnname n)

Example:

```
mysql> INSERT INTO SAILORS VALUES (22,'DUSTIN', 7, 45.0);
```
1 row created

```
mysql> INSERT INTO SAILORS VALUES (29,'BRUTUS', 1, 33.0);
```
1 row created

INSERTION of Data can also be done by the following Syntax:

SYNTAX

mysql> INSERT
INTO tablename
(columnname1, columnname2,….columnname n)
VALUES

(Value1,Value2,..Value n);

Example:

```
mysql> INSERT
       INTO SAILORS
       SID,SNAME,RATING,AGE)
       VALUES
       (31,'LUBBER', 8, 55.5);
```

Example1: INSERT data into RESERVES table. (Insert at least 5 rows)

TEST OUTPUT:

Example 2: INSERT data into BOATS table. (Insert at least 5 rows)

TEST OUTPUT:

2. To Retrieve / Display Data from Tables (SELECT)

a. Select command is used to select and display the values from table

SYNTAX

mysql> SELECT * FROM <TABLENAME>;

Example:

```
mysql> SELECT * FROM SAILORS;
```

TEST OUTPUT:

b. Retrieving specific columns from a table

mysq> SELECT columnname 1, columnname 2,…. columnname n

FROM <table name>;

Example:

```
mysql> SELECT SNAME, AGE FROM SAILORS;
```

TEST OUTPUT

c. Retrieving distinct rows from a table

mysql> SELECT DISTINCT

columnname 1, columnname 2,…. columnname n

FROM<table name>;

Example:

```
mysql> SELECT DISTINCT RATING FROM SAILORS;
```

TEST OUTPUT

d. Selecting data from table based on some condition

mysql> **SELECT**

columnname 1, columnname 2,…. columnname n

FROM <table name>

WHERE <search condition>;

Example:

```
mysql> SELECT SID, SNAME, RATING FROM SAILORS WHERE (RATING>7);
```

TEST OUTPUT

Example1: Display Data from RESERVES Table

Example2: Display Data from BOATS Table

3. UPDATE

This MYSQL command is used to modify the values in an existing table.

mysql>**UPDATE** <table name>

SET column1= expression1, column2= expression 2,...

WHERE < column= value>;

An expression consists of a constant (new value), arithmetic or string operation or an MYSQL query. Note that the new value to assign to <column> must match data type.

An update statement used without a where clause results in changing respective attributes of all tuples in the specified table.

Example

```
mysql> UPDATE SAILORS SET AGE=21 WHERE SID=31;
```

TEST OUTPUT

4. **DELETE :**

In order to delete rows from a table we use this command

mysql>DELETE

FROM <table name>

WHERE <condition>;

Based on the condition specified the rows gets fetched from the table and gets deleted in table. Here the WHERE clause is optional.

Example1:

```
MYSQL> DELETE FROM SAILORS WHERE SNAME ="BRUTUS";
```

TEST OUTPUT

Example2:

```
mysql> DELETE
       FROM SAILORS
       WHERE SID=29;
```

TEST OUTPUT

LAB EXERCISES

Perform all DML operations on SALES table, PRODUCT table and PURCHASE table.

Exp. No: 5(WEEK 5)	KEY CONSTRAINTS

KEY CONSTRAINTS

1. PRIMARY KEY & NOT NULL

Example:

```
mysql> CREATE TABLE STUDENT (
       SID INT,
       SNAME VARCHAR(32) ,
       CGPA REAL NOT NULL,
       AGE INT(3), PRIMARY KEY (SID));
```

Test Output:

Imposing Integrity Constraint using ALTER

Example:

```
mysql> ALTER TABLE STUDENT
       MODIFY
       SNAME VARCHAR(32) NOT NULL;
```

Test Output

Example: Alter the integrity constraint of SNAME column to NOT NULL in STUDENT table

Test Output:

2. DEFAULT

```
mysql> CREATE TABLE STUDENT1 (
       SID INT,
       SNAME VARCHAR(128) ,
       CGPA REAL NOT NULL,
       AGE INT(3)DEFAULT 18, PRIMARY KEY (SID));
```

TEST OUTPUT:

Example:

```
mysql> INSERT INTO STUDENT1 VALUES (100, 'RAHUL', 8.5, DEFAULT);
```

TEST OUTPUT

3. UNIQUE

```
mysql> CREATE TABLE STUDENT2 (
      SID INT,
      SNAME VARCHAR(128) UNIQUE,
      CGPA REAL NOT NULL,
      AGE INT(3)DEFAULT 18, PRIMARY KEY (SID));
```

Example:

```
mysql> INSERT INTO STUDENT2 VALUES (100, 'RAHUL', 8.5, 18);

mysql> INSERT INTO STUDENT2 VALUES (101, 'RAHUL', 8.5, 18);
```

TEST OUTPUT:

4. FOREIGN KEY

```
mysql> CREATE TABLE LIBRARY (
      SID INT,
      LID INT(4),
      BNAME VARCHAR(60),
      BID INT(5),
      PRIMARY KEY (LID),
      CONSTRAINT FOREIGN KEY (SID) REFERENCES STUDENT(SID)
      );
```

TEST OUTPUT

Adding Foreign Key to an existing Table

```
mysql>  ALTER TABLE
        ADD
        FOREIGN KEY (SID REFERENCES SAILORS (SID));
```

TEST OUTPUT

LAB EXERCISES

1. Apply primary key constraint on Sales_Id and UNIQUE key on Dept_No from SALES table.
2. Apply primary key constraint on Model and NOT NULL on Cost from PRODUCT table.
3. Apply Foreign keys on Sales_id and Model on PURCHASE table.

Exp. No: 6(WEEK 6)	AGGREGATE FUNCTIONS & NUMERIC FUNCTIONS

I) AGGREGATE FUNCTIONS

Create a table **employee** with following information and insert appropriate rows.

Table Name: employee		
S. no	**Fileld Name**	**Data Type**
1	eid	Integer
2	ename	Varchar
3	Salary	Float
4	desig	Varchar

1. COUNT:

SYNTAX:

Select count ([<distinct>/<ALL]<expr>)

Query: Count number of different Employee names?

```
mysql>SELECT COUNT (distinct ename) FROM employee;
```

TEST RESULT:

Query: Count number of different Designation names?

```
mysql>SELECT COUNT (distinct desig) FROM employee;
```

TEST RESULT:

2. **SUM:**

SYNTAX:

Select SUM ([<distinct>/<ALL]<n>)

MYSQL>

Query: Find the sum of salary of all employees?

```
Mysql>SELECT SUM(salary) FROM employee;
```

TEST RESULT:

3. **AVG:**

SYNTAX:

Select AVG ([<distinct>/<ALL]<n>)

Query: Find the average salary of all employees?

```
Mysql>SELECT AVG(salary)FROM employee;
```

TEST RESULT:

4. **MIN:**

SYNTAX:

Select MIN ([<distinct>/<ALL]<n>)

Query: Find the lowest salary from all employees?

```
Mysql>SELECT MIN(salary)FROM employee;
```

TEST RESULT:

5. **MAX:**

SYNTAX:

Select MAX ([<distinct>/<ALL]<n>)

Query: Find the Highest salary from all employees?

```
Mysql>SELECT MAX(salary)FROM employee;
```

TEST RESULT:

II) NUMERIC FUNCTIONS

SYNTAX: ceil ()

Ex: `mysql> SELECT CEIL (9.5);`

Test output:

SYNTAX: floor ()

Ex: `mysql>SELECT FLOOR(10.5);`

Test output:

SYNTAX mod ()

Ex: `mysql>SELECT MOD(17,5);`

Test output

SYNTAX: power (n,m)

Ex: `mysql>SELECT POWER(2,2);`

Test output:

SYNTAX: round (n,m)

Ex: `mysql>SELECT ROUND(10.586,2);`

Test output:

SYNTAX: truncate (n,m)

Ex: `mysql>SELECT TRUNCATE(1.223,1);`

Test output:

SYNTAX: sign ()

Ex: `mysql>SELECT SIGN(-5);`

Test output:

SYNTAX: sort ()

Ex: `mysql>SELECT SQRT(25);`

Test output:

SYNTAX: abs ()

Ex: `mysql>SELECT ABS(-9);`

Test output:

II) SPECIAL OPERATORS IN MYSQL

a) BETWEEN & AND

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

Query: Select the list of employees whose salary between 5000 and 8000

Example:

```
MYSQL> Select * From employee WHERE salary BETWEEN 5000 AND 8000;
```

Test Output:

b) IN Operator:

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

Query: Select the list of employees whose salary between 5000 and 8000

```
MYSQL>SELECT * FROM employee WHERE Desig IN(MANAGER,CLERK);
```

Test Output:

c) LIKE Operator:

Syntax:

```
SELECT column1,column2,...
FROM table_name
WHERE columnN LIKE pattern;
```

Query: Select the list of employees whose name starts with "s%"

```
MYSQL>SELECT*FROM employee WHERE ename like‘s%’;
```

Test Output:

d) Logical operator:

AND Operator:

Query: Select list of all employees whose salary is greater than 5000 and designation is manager.

```
MYSQL> select * from employee where salary > 5000 and desig =”Manager”;
```

Test Output:

OR Operator:

Query: Select list of all employees whose salary is greater than 5000 or designation is clerk.

```
MYSQL>select * from employee where salary > 5000 or desig = “clerk”;
```

Test Output:

NOT Operator:

Query: Select list of all employees who is not a clerk.

```
MYSQL>select*from employee where not desig =”clerk”;
```

Test Output:

III) String Functions

1) **Lower:** This String function will convert input string in to lower case.

Syntax: lower(string)

Example:

```
MYSQL> select lower(‘DATABASE MANAGEMENT SYSTEMS ’) ;
```

Test Output:

2) **Upper:** This string function will convert input string in to upper case.

Syntax: Upper(string)

Example:

```
MYSQL> select upper (‘database management systems ’);
```

Test Output:

3) Ltrim (Left Trim):

Syntax: Ltrim(“ string”)

Example:

```
MYSQL> SELECT LTRIM("     DBMS");
```

Test Output:

4) Rtrim (Right Trim):

Syntax: Rtrim(“string ”)

Example:

```
MYSQL> SELECT RTRIM("DBMS    ");
```

Test Output:

5) Replace:

Syntax: Replace(string, searchstring, replacestring)

Example:

```
MYSQL> SELECT REPLACE(“GOOD MORNING STUDENTS”,”MORNING”,”AFTERNOON”);
```

Test Output:

6) Substr:

Syntax: Substr (string, starts [, count])

Example:

```
     MYSQL>select substr(“GOOD MORNING”,6,7);
```

Test Output:

7) Char:

Syntax: Char (number)

Example:

```
MYSQL>select char(65);
```

Test Output:

8) Lpad (Left Pad):

Syntax: Lpad(String, length, pattern)

Example:

```
Mysql>select lpad("Welcome",15,'*');
```

Test Output

9) Rpad (Right Pad):

Syntax: Rpad(String, length, pattern)

Example:

```
mysql> select rpad("Welcome",15,'*') ;
```

Test Output:

10) Length:

Syntax: Length (string)

Example:

```
mysql> select length("DBMS");
```

Test Output:

LAB EXERCISES

1. Apply aggregate functions on PRODUCT table.
2. Apply string functions of SALES table.

Exp. No: 7(WEEK-7)	NESTED QUERIES

NESTED QUERIES

NESTED QUERIES:

Nested Query is a query that has another query embedded within it.

IN- is an operator which allows us to check whether a value is present in a given set of elements.

Create following Tables with at least 5 rows in orders and 10 rows in Customers

```
Orders (OrderID, CustomerID, OrderDate);
Customers (CustomerID, CustomerName, ContactName, Country);
```

Example1:

Query: Find all the customers who have placed at least one order.

```
SELECT * FROM Customers
WHERE
CustomerID IN (SELECT CustomerID FROM Orders);
```

Test Output:

Exp. No: 8(WEEK-8)	CORRELATED QUERIES & MYSQL CLAUSES

CORRELATED QUERIES:

- A correlated subquery in MySQL is a subquery that depends on the outer query. It uses the data from the outer query or contains a reference to a parent query that also appears in the outer query.
- MySQL evaluates it once from each row in the outer query.

Create table employees with following fields and insert appropriate rows.

employees(emp_id, emp_name, emp_age, city, income)

Query: Select an employee name and city whose income is higher than the average income of all employees in each city

```
SELECT emp_name, city, income
FROM employees emp
WHERE income > (SELECT AVG(income)
FROM employees WHERE city = emp.city);
```

Test output:

MySQL Clauses

S. NO	Name of the Clause	Description
1	MySQL "ORDER BY" clause	It helps to show data in ascending or descending order.
2	MySQL "GROUP BY" clause	This clause displays information in a particular group.
3	MySQL "HAVING" clause	This clause applies after the "group by" clause.

MySQL "ORDER BY" clause.

Syntax:

SELECT expressions

FROM tables

[WHERE conditions]

ORDER BY expression [ASC | DESC];

Query:

List the employee's details residing in Bangalore city order by Employee name.

Example:

```
SELECT * FROM employees
WHERE city = 'Bangalore'
ORDER BY emp_name;
```

Test Output:

MySQL "GROUP BY" clause.

Syntax:

SELECT expression1, expression2, ... expression_n,
aggregate_function (expression)
FROM tables
[**WHERE** conditions]
GROUP BY expression1, expression2, ... expression_n;

Query: list the number of employees working group by city wise

```
Example:
     mysql> SELECT city, count(city)
            From employees
            Group by city;
```

Test output:

MySQL "HAVING" clause.

Syntax:

SELECT column1, column2
FROM table name
GROUP BY column1, column2
HAVING condition;

Query: Select the employees group by city where income is greater than 5000.

Example:

```
     mysql> select * from employees
            GROUP BY city
            HAVING income >5000;
```

Test output:

LAB EXERCISES
NESTED AND CORRELATED QUERIES

1. Find the Sales_Id of sales men who sold a model like %CR%
2. Find the sales men who sold all models.
3. Find the sales men who sold highest cost model.
4. For each model, find the average cost of the model.
5. Find the Sales_id of sales man who sold more costly product than every sales man

Exp. No: 9 (WEEK-9)	VIEWS

VIEWS

- A view is a database object that has no values. Its contents are based on the base table.
- It contains rows and columns similar to the real table.
- In MySQL, the View is a **virtual table** created by a query by joining one or more tables.
- If any changes occur in the underlying table, the same changes reflected in the View also.

Create a table emp with following fields and insert at least 10 rows.

emp (empno, empname, salary, deptno)

I. Create a view in MySQL

1. Syntax: Create View <View_Name> As Select statement;

Example:

```
MYSQL>Create View EmpView As Select * from emp;
```

Test Output:

2. Syntax: Select columnname1, columnname2 from <View_Name>;

Example:

```
MYSQL> Select empno, ename, Salary
       from EmpView
       where deptno  in(10,30);
```

Test Output:

II. UPDATABLE VIEWS:

Syntax for creating an Updatable View:

```
Create View Emp_vw As
Select Empno, Ename, Deptno
from Emp;
```

View created.

Query: insert row into view

```
MYSQL>Insert into Emp_vw values(1126,'Brijesh',20);
```

Test Output:

Query: update department number is 30 whose employee number is 1125

```
MYSQL>Update Emp_vw set Deptno=30 where Empno=1125;
```

Test Output:

Query: delete a row from the view whose employee number is 1122

```
MYSQL>Delete from Emp_vw where Empno=1122;
```

Test Output:

Query: update the view to set salary 4300 whose employee number is 1125

```
MYSQL>Update EmpDept_Vw set salary=4300 where Empno=1125;
```

Test Output:

Query: delete a row from the view whose employee number is 1123

```
MYSQL>Delete From EmpDept_Vw where Empno=1123;
```

Test Output

III. DESTROYING A VIEW:

Syntax: Drop View <View_Name>;

Query: drop the view

Example:

```
MYSQL>Drop View Emp_Vw;
```

Test Output:

LAB EXERCISES

create a table salesman with following fields and insert the values

salesman_id	name	city	commission
5001	Kumar	New York	0.15
5002	Rohit	Paris	0.13
5005	Santhosh	London	0.11
5006	Nagaraj	Paris	0.14
5007	Ramya	Rome	0.13
5003	Jaideep	San Jose	0.12

Write and execute the following queries

1. From the above table, create a view for those salespeople who belong to the city of New York.
2. From the above table, create a view for all salespersons. Return salesperson ID, name, and city.
3. From the above table update commission 0.20 whose salesman id is 5007.
4. From the above table update city Delhi whose salesman id is 5005.
5. Drop the View

Exp. No: 10(WEEK-10)	JOINS

JOINS

MYSQL joins are used to query data from two or more tables, based on a relationshipbetween certain columns in these tables.

ORDER TABLE		
OrderID	**CustomerID**	**OrderDate**
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

Customers Table			
CustomerID	**CustomerName**	**ContactName**	**Country**
1	Jaideep	Nagaraj	India
2	Rohith	Santhosh	London
3	Akshay	Lingamoorthy	China

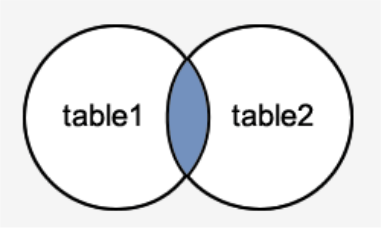


Fig a. Inner Join (Simple Join)

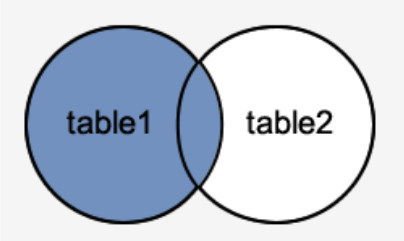


Fig b. Left outer join (Left join)

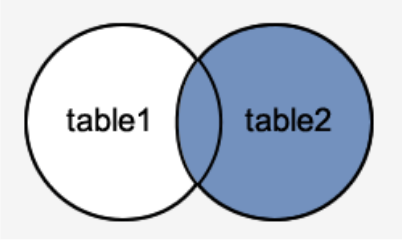


Fig c. Right Outer Join (Right join)

INNER JOIN

The **INNER JOIN** keyword return rows when there is at least one match in both tables.

Example 1:

```
Mysql>SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
      FROM Orders
      INNER JOIN Customers
      ON Orders.CustomerID = Customers.CustomerID;
```

Test Output

Example 2:

```
Mysql>SELECT Customers.CustomerName,  Orders.OrderID
     FROM Customers
     INNER JOIN Orders
     ON Customers.CustomerID=Orders.CustomerID
     ORDER BY Customers.CustomerName;
```

TEST OUTPUT

LEFT JOIN

The LEFT JOIN keyword returns all rows from the left table (table_name1), even if there are no matches in the right table (table_name2).

```
mysql>  SELECT Customers.CustomerName, Orders.OrderID
        FROM Customers
        LEFT JOIN Orders
        ON Customers.CustomerID=Orders.CustomerID
        ORDER BY Customers.CustomerName;
```

TEST OUTPUT

RIGHT JOIN

The RIGHT JOIN keyword Return all rows from the right table (table_name2), even if there are no matches in the left table (table_name1).

This is another table named employee.here they have to give field accordingly.

```
mysql>  SELECT Customers.CustomerName, Orders.OrderID
        FROM Customers
        RIGHT JOIN Orders
        ON Customers.CustomerID=Orders.CustomerID
        ORDER BY Customers.CustomerName;
```

TEST OUTPUT

UNION

Suppose our database has the following tables: "Student1" and "Student2" that contains the data below:

Table: student1

stud_id	stud_name	subject	marks
1	Mark	English	68
2	Joseph	Physics	70
3	John	Maths	70
4	Barack	Maths	90
5	Rinky	Maths	85
6	Adam	Science	92
7	Andrew	Science	83
8	Brayan	Science	85

Table: Student2

stud_id	stud_name	subject	marks
1	Donald	History	85
2	Joseph	Physics	70
3	Stephen	Geography	82
4	Abraham	Java	75
5	John	Maths	70

The following statement produces output that contains all student names and subjects by combining both tables.

```
mySql> SELECT stud_name, subject FROM student1
       UNION
       SELECT stud_name, subject FROM student2;
```

TEST OUTPUT

LAB EXERCISES

Perform the following operations on above tables

a) inner join b) left join c) right join d) Union

Table: company

COMPANY_ID	COMPANY_NAME	COMPANY_CITY
18	Order All	Boston
15	Jack Hill Ltd	London
16	Akas Foods	Delhi
17	Foodies.	London
19	sip-n-Bite.	New York

Table: foods

ITEM_ID	ITEM_NAME	ITEM_UNIT	COMPANY_ID
1	Chex Mix	Pcs	16
6	Cheez-It	Pcs	15
2	BN Biscuit	Pcs	15
3	Mighty Munch	Pcs	17
4	Pot Rice	Pcs	15
5	Jaffa Cakes	Pcs	18

Exp. No: 11(WEEK-11)	TRIGGER

- A trigger in MySQL is a set of SQL statements that reside in a system catalog.
- **It is a special type of stored procedure that is invoked automatically in response to an event**. Each trigger is associated with a table, which is activated on any DML statement such as **INSERT, UPDATE**, or **DELETE**.
- A trigger is called a special procedure because it cannot be called directly like a stored procedure.
- The main difference between the trigger and procedure is that a trigger is called automatically when a data modification event is made against a table.

Types of Triggers

We can define the maximum six types of actions or events in the form of triggers:

Before Insert

It is activated before the insertion of data into the table.

After Insert

It is activated after the insertion of data into the table.

Before Update

It is activated before the update of data in the table.

After Update

It is activated after the update of the data in the table.

Before Delete

It is activated before the data is removed from the table.

After Delete

It is activated after the deletion of data from the table.

Create triggers in MySQL:

Syntax:

CREATE TRIGGER <trigger_name >
(**AFTER** | BEFORE) (**INSERT** | **UPDATE** | **DELETE**)
ON table_name **FOR** EACH ROW
BEGIN
--variable declarations
--trigger code
END;

Example:

1. Create a Following Table
Query :

```
CREATE TABLE employee(
    name varchar(45) NOT NULL,
    occupation varchar(35) NOT NULL,
    working_date date,
    working_hours varchar(10)
);
```

2. Insert the following Values into the table

Query :

```
INSERT INTO employee VALUES
('Robin', 'Scientist', '2020-10-04', 12),
('Warner', 'Engineer', '2020-10-04', 10),
('Peter', 'Actor', '2020-10-04', 13),
('Marco', 'Doctor', '2020-10-04', 14),
('Brayden', 'Teacher', '2020-10-04', 12),
('Antonio', 'Business', '2020-10-04', 11);
```

3. We will create a **BEFORE INSERT trigger**. This trigger is invoked automatically insert the **working_hours = 0** if someone tries to insert **working_hours < 0**.

Query :

```
mysql> DELIMITER //
mysql> Create Trigger before_insert_empworkinghours
BEFORE INSERT ON employee FOR EACH ROW
BEGIN
IF NEW.working_hours < 0 THEN SET NEW.working_hours = 0;
END IF;
END //
```

Test result:

Lab Exercises

Exp. No: 12(WEEK-12)	DCL COMMANDS & TCL COMMANDS

DCL (DATA CONTROL LANGUAGE):

- All user activity occurs inside a transaction. If autocommit mode is enabled, each SQL statement forms a single transaction on its own.
- By default, MySQL starts the session for each new connection with autocommit enabled, so MySQL does a commit after each SQL statement if that statement did not return an error. If a statement returns an error, the commit or rollback behavior depends on the error.
- A session that has autocommit enabled can perform a multiple-statement transaction by starting it with an explicit START TRANSACTION or BEGIN statement and ending it with a COMMIT or ROLLBACK statement.
- If autocommit mode is disabled within a session with SET autocommit = 0, the session always has a transaction open.
- A COMMIT or ROLLBACK statement ends the current transaction and a new one starts.
- If a session that has autocommit disabled ends without explicitly committing the final transaction, MySQL rolls back that transaction.

The TCL commands are:

1. COMMIT
2. ROLLBACK
3. SAVEPOINT

1. COMMIT:

This command is used to save the data permanently.

2. ROLLBACK:

This command is used to get the data or restore the data to the last savepoint or last committed state.

3. SAVEPOINT:

This command is used to save the data at a particular point temporarily, so that whenever needed can be rollback to that particular point.

Example:

```
mysql> CREATE TABLE customer (a INT, b CHAR (20));


mysql> -- Do a transaction with autocommit turned on.

mysql> START TRANSACTION;
TEST RESULT:


mysql> INSERT INTO customer VALUES (10, 'Sarvesh');
TEST RESULT:


mysql> COMMIT;
TEST RESULT :


mysql> -- Do another transaction with autocommit turned off.

mysql> SET autocommit=0;
TEST RESULT:



mysql> INSERT INTO customer VALUES (15, 'John');

mysql> INSERT INTO customer VALUES (20, 'Yadhvika');

mysql> DELETE FROM customer WHERE b = 'Sarvesh';
TEST RESULT:

mysql> -- Now we undo those last 2 inserts and the delete.
mysql> ROLLBACK;


TEST RESULT:
```

```
mysql> SELECT * FROM customer;
```

TEST RESULT:

Data Control Language (DCL)

- Data Control Language (DCL) is used to control privileges in Database. To perform any operation in the database, such as for creating tables, sequences or views, a user needs privileges. Privileges are of two types,
- **System:** This includes permissions for creating session, table, etc and all types of other system privileges.
- **Object:** This includes permissions for any command or query to perform any operation on the database tables.
- In DCL we have two commands,
- **GRANT:** Used to provide any user access privileges or other priviliges for the database.
- **REVOKE:** Used to take back permissions from any user.

SQL GRANT Command

Syntax:

```
GRANT privilege_name
ON object_name
TO {user_name |PUBLIC |role_name}
[WITH GRANT OPTION];
```

SQL REVOKE Command:

Syntax:

```
Mysql> REVOKE privilege_name
ON object_name
FROM {user_name |PUBLIC |role_name}
```

MySQL SHOW GRANTS statement examples

A) Using MySQL SHOW GRANTS to display the privileges granted for the current user

The following statement uses the SHOW GRANTS statement to display the privileges granted for the current user:

```
Mysql> SHOW GRANTS;
        or
Mysql> SHOW GRANTS FOR CURRENT_USER;
        Or
Mysql> SHOW GRANTS FOR CURRENT_USER();
```

Test Result :

B) Using MySQL SHOW GRANTS to display the privileges granted for a user

First, create a new database named vehicles:

```
Mysql> CREATE DATABASE vehicles;
```

Second, select the database vehicles:

```
Mysql> USE vehicles;
```

Third, create a new table called cars in the vehicles database:

```
MySQL> CREATE TABLE cars (
       id INT AUTO_INCREMENT,
       make VARCHAR(100) NOT NULL,
       model VARCHAR(100) NOT NULL,
       PRIMARY KEY (id));
```

Fourth, create a new user called musk@localhost:

```
Mysql> CREATE USER CSE@localhost
       IDENTIFIED BY 'mruh@';
```

fifth, show the default privileges granted to the user CSE@localhost:

```
mysql> SHOW GRANTS
       FOR CSE@localhost;
```

Test Result:

C) Using MySQL GRANTS to apply privileges to a user

Grant SELECT, INSERT, UPDATE, and DELETE privileges on the vehicles database to the CSE@localhost:

```
mysql>GRANT
      SELECT,INSERT,UPDATE,DELETE
      ON vehicles.*
      TO CSE@localhost;
```

Test Result:

D) Using MySQL REVOKE to extract privileges from a user

Revoke the UPDATE and INSERT privileges from CSE@localhost:

```
Mysql> REVOKE INSERT, UPDATE
       ON vehicles.*
       FROM CSE@localhost;
```

Test Result:

```
mysql> SHOW GRANTS
       FOR CSE@localhost;
```

Test Result: