Ex.No:1a	**CREATION OF A DATABASE AND WRITING SQL QUERIES**

AIM:

To create a database and execute the SQL Commands for retrieving information from the database.

PROCEDURE:

STEP 1: Start

STEP 2: Create a database.

STEP 2: Create the table with its essential attributes.

STEP 3: Execute different Commands and extract information from the table.

STEP 4: Stop

Syntax for creation of database:

CREATE DATABASE DATABASENAME;

USE DATABASENAME;

Syntax for creation of a table:

SQL> CREATE <OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1 <DATATYPE> (SIZE), COLUMN NAME.2 <DATATYPE> (SIZE) ……);

(or)

SQL> CREATE TABLE <TARGET TABLE NAME> SELECT EMPNO, ENAME FROM <SOURCE TABLE NAME>;

(or)

SQL> CREATE TABLE <TARGET TABLE NAME> AS SELECT * FROM <SOURCE TABLE NAME> WHERE <FALSE CONDITION>;

Syntax for describing a table:

SQL> DESC <TABLE NAME>;

Syntax :

SQL > INSERT INTO <TABLE NAME> VALUES< VAL1, 'VAL2',…..);

Syntax:

SQL> SELECT * FROM <TABLE NAME>;

mysql> create database university;

```
mysql> create database university;
Query OK, 1 row affected (0.09 sec)
```

mysql> use university;

```
mysql> use university;
Database changed
```

mysql> CREATE TABLE Student (StudentID int, LastName varchar(255), FirstName varchar(255), Address varchar(255), City varchar(255));

```
mysql> CREATE TABLE Student (StudentID int, LastName varchar(255), FirstName varchar(255), Address varchar(255),    Cit
y varchar(255));
Query OK, 0 rows affected (1.70 sec)
```

mysql> Desc student;

```
mysql> desc student;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| StudentID | int          | YES  |     | NULL    |       |
| LastName  | varchar(255) | YES  |     | NULL    |       |
| FirstName | varchar(255) | YES  |     | NULL    |       |
| Address   | varchar(255) | YES  |     | NULL    |       |
| City      | varchar(255) | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
5 rows in set (0.24 sec)
```

mysql> insert into student values (1001, 'Krishnan', 'Amal', '254 First Street', 'Chennai');

```
mysql> insert into student values (1001, 'Krishnan', 'Amal', '254 First Street', 'Chennai');
Query OK, 1 row affected (0.18 sec)
```

mysql> insert into student (StudentID, LastName, FirstName, Address, City) values (1002, 'Kumar', 'Mahesh', '274 First Street', 'Chennai');

```
mysql> insert into student (StudentID, LastName, FirstName, Address, City) values (1002, 'Kumar', 'Mahesh', '274 First S
treet', 'Chennai');
Query OK, 1 row affected (0.13 sec)
```

mysql> insert into student (StudentID, FirstName, Address, City) values (1003, 'Mahesh', '294 First Street', 'Chennai');

```
mysql> insert into student (StudentID, FirstName, Address, City) values (1003, 'Mahesh', '294 First Street', 'Chennai');
Query OK, 1 row affected (0.12 sec)
```

mysql> select * from student;

```
mysql> select * from student;
+-----------+----------+-----------+------------------+---------+
| StudentID | LastName | FirstName | Address          | City    |
+-----------+----------+-----------+------------------+---------+
|      1001 | Krishnan | Amal      | 254 First Street | Chennai |
|      1002 | Kumar    | Mahesh    | 274 First Street | Chennai |
|      1003 | NULL     | Mahesh    | 294 First Street | Chennai |
+-----------+----------+-----------+------------------+---------+
3 rows in set (0.00 sec)
```

mysql> select StudentID, City from student;

```
mysql> select StudentID, City from student;
+-----------+---------+
| StudentID | City    |
+-----------+---------+
|      1001 | Chennai |
|      1002 | Chennai |
|      1003 | Chennai |
+-----------+---------+
3 rows in set (0.00 sec)
```

Result:

Thus the SQL commands to create and retrieve information from the database are executed successfully.

Ex.No:1b	**INSERTION, DELETION, MODIFYING, ALTERING, UPDATING AND VIEWING RECORDS**

AIM:

To execute and verify the Data Manipulation Language (DML) and Data Definition Language (DDL) commands.

DDL (DATA DEFINITION LANGUAGE)

- DROP
- TRUNCATE
- ALTER
- RENAME

DML (DATA MANIPULATION LANGUAGE)

- INSERT
- SELECT
- UPDATE
- DELETE

PROCEDURE

STEP 1: Start

STEP 2: Create the table with its essential attributes.

STEP 3: Insert the record into table

STEP 4: Update the existing records into the table

STEP 5: Delete the records from the table

STEP 6: Viewing records from database.

STEP 7: Alter and Modify the table.

STEP 8: Stop

SQL COMMANDS:

Syntax:

SQL > INSERT INTO <TABLE NAME> VALUES< VAL1, 'VAL2',…..);

(or)

SQL> INSERT INTO <TABLE NAME> VALUES< '&column name', '&column name 2',…..);

Syntax:

SQL> SELECT * FROM <TABLE NAME>;

Syntax:

SQL> UPDATE <<TABLE NAME> SET <COLUMNANE>=<VALUE> WHERE <COLUMN NAME=<VALUE>;

Syntax for updating multiple Records from the table:

SQL> UPDATE <<TABLE NAME> SET <COLUMNANE>=<VALUE> WHERE <COLUMN NAME=<VALUE>;

Syntax:

SQL> DELETE <TABLE NAME> WHERE <COLUMN NAME>=<VALUE>;

Syntax to Alter & Modify on a Single Column:

SQL > ALTER <TABLE NAME> MODIFY <COLUMN NAME><DATATYPE> (SIZE);

Syntax to alter multiple column:

SQL > ALTER <TABLE NAME> MODIFY <COLUMN NAME1><DATATYPE> (SIZE), MODIFY <COLUMN NAME2><DATATYPE> (SIZE)…….;

Syntax to add a new column:

SQL> ALTER TABLE <TABLE NAME> ADD (<COLUMN NAME><DATA TYPE><SIZE>);

Syntax for add multiple new columns:

SQL> ALTER TABLE <TABLE NAME> ADD (<COLUMN NAME1><DATA TYPE><SIZE>,(<COLUMN NAME2><DATA TYPE><SIZE>,……);

Syntax :

SQL> ALTER TABLE <TABLE NAME> DROP COLUMN <COLUMN NAME>;

Syntax:

SQL> ALTER TABLE <TABLE NAME> DROP <COLUMN NAME1>,<COLUMN NAME2>,…….. ;

Syntax:

SQL> ALTER TABLE RENAME <OLD NAME> TO <NEW NAME>;

mysql> insert into student values (1001, 'Krishnan', 'Amal', '254 First Street', 'Chennai');

```
mysql> insert into student values (1001, 'Krishnan', 'Amal', '254 First Street', 'Chennai');
Query OK, 1 row affected (0.18 sec)
```

mysql> insert into student (StudentID, LastName, FirstName, Address, City) values (1002, 'Kumar', 'Mahesh', '274 First Street', 'Chennai');

```
mysql> insert into student (StudentID, LastName, FirstName, Address, City) values (1002, 'Kumar', 'Mahesh', '274 First S
treet', 'Chennai');
Query OK, 1 row affected (0.13 sec)
```

mysql> insert into student (StudentID, FirstName, Address, City) values (1003, 'Mahesh', '294 First Street', 'Chennai');

```
mysql> insert into student (StudentID, FirstName, Address, City) values (1003, 'Mahesh', '294 First Street', 'Chennai');

Query OK, 1 row affected (0.12 sec)
```

mysql> select * from student;

```
mysql> select * from student;
+-----------+----------+-----------+-------------------+---------+
| StudentID | LastName | FirstName | Address           | City    |
+-----------+----------+-----------+-------------------+---------+
|      1001 | Krishnan | Amal      | 254 First Street  | Chennai |
|      1002 | Kumar    | Mahesh    | 274 First Street  | Chennai |
|      1003 | NULL     | Mahesh    | 294 First Street  | Chennai |
+-----------+----------+-----------+-------------------+---------+
3 rows in set (0.00 sec)
```

mysql> update student SET City = 'Madurai' where StudentID = 1002;

```
mysql> update student SET City = 'Madurai' where StudentID = 1002;
Query OK, 1 row affected (0.16 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

mysql> update student SET City = 'Trichy', Address= '204 Second Street' WHERE StudentID = 1001;

```
mysql> update student SET City = 'Trichy', Address= '204 Second Street' WHERE StudentID = 1001;
Query OK, 1 row affected (0.12 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

mysql> select * from student;

```
mysql> select * from student;
+-----------+----------+-----------+-------------------+---------+
| StudentID | LastName | FirstName | Address           | City    |
+-----------+----------+-----------+-------------------+---------+
|      1001 | Krishnan | Amal      | 204 Second Street | Trichy  |
|      1002 | Kumar    | Mahesh    | 274 First Street  | Madurai |
|      1003 | NULL     | Mahesh    | 294 First Street  | Chennai |
+-----------+----------+-----------+-------------------+---------+
3 rows in set (0.00 sec)
```

mysql> delete from student where StudentID=1003;

```
mysql> delete from student where StudentID=1003;
Query OK, 1 row affected (0.20 sec)
```

mysql> select * from student;

```
mysql> select * from student;
+-----------+----------+-----------+-------------------+---------+
| StudentID | LastName | FirstName | Address           | City    |
+-----------+----------+-----------+-------------------+---------+
|      1001 | Krishnan | Amal      | 204 Second Street | Trichy  |
|      1002 | Kumar    | Mahesh    | 274 First Street  | Madurai |
+-----------+----------+-----------+-------------------+---------+
2 rows in set (0.06 sec)
```

mysql> alter table student modify column StudentID varchar(255);

```
mysql> ALTER TABLE student MODIFY COLUMN StudentID varchar(255);
Query OK, 2 rows affected (2.83 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

mysql> desc student;

```
mysql> desc student;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| StudentID | varchar(255) | YES  |     | NULL    |       |
| LastName  | varchar(255) | YES  |     | NULL    |       |
| FirstName | varchar(255) | YES  |     | NULL    |       |
| Address   | varchar(255) | YES  |     | NULL    |       |
| City      | varchar(255) | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
5 rows in set (0.02 sec)
```

mysql> alter table student add phone varchar(255);

```
mysql> ALTER TABLE student ADD phone varchar(255);
Query OK, 0 rows affected (0.37 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

mysql> desc student;

```
mysql> desc student;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| StudentID | varchar(255) | YES  |     | NULL    |       |
| LastName  | varchar(255) | YES  |     | NULL    |       |
| FirstName | varchar(255) | YES  |     | NULL    |       |
| Address   | varchar(255) | YES  |     | NULL    |       |
| City      | varchar(255) | YES  |     | NULL    |       |
| phone     | varchar(255) | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
6 rows in set (0.11 sec)
```

mysql> alter table student modify column StudentID int, modify phone int;

```
mysql> ALTER TABLE student MODIFY COLUMN StudentID int, modify phone int;
Query OK, 0 rows affected (0.15 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

mysql> desc student;

```
mysql> desc student;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| StudentID | int          | YES  |     | NULL    |       |
| LastName  | varchar(255) | YES  |     | NULL    |       |
| FirstName | varchar(255) | YES  |     | NULL    |       |
| Address   | varchar(255) | YES  |     | NULL    |       |
| City      | varchar(255) | YES  |     | NULL    |       |
| phone     | int          | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
6 rows in set (0.08 sec)
```

mysql> alter table student add DateOfBirth date, add email varchar(255);

```
mysql> ALTER TABLE student ADD DateOfBirth date, add email varchar(255);
Query OK, 0 rows affected (0.90 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

mysql> desc student;

```
mysql> desc student;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| StudentID   | int          | YES  |     | NULL    |       |
| LastName    | varchar(255) | YES  |     | NULL    |       |
| FirstName   | varchar(255) | YES  |     | NULL    |       |
| Address     | varchar(255) | YES  |     | NULL    |       |
| City        | varchar(255) | YES  |     | NULL    |       |
| phone       | int          | YES  |     | NULL    |       |
| DateOfBirth | date         | YES  |     | NULL    |       |
| email       | varchar(255) | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
8 rows in set (0.07 sec)
```

Mysql> ALTER TABLE student DROP COLUMN email;

```
mysql> ALTER TABLE student DROP COLUMN email;
Query OK, 0 rows affected (2.56 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

mysql> desc student;

```
mysql> desc student;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| StudentID   | int          | YES  |     | NULL    |       |
| LastName    | varchar(255) | YES  |     | NULL    |       |
| FirstName   | varchar(255) | YES  |     | NULL    |       |
| Address     | varchar(255) | YES  |     | NULL    |       |
| City        | varchar(255) | YES  |     | NULL    |       |
| phone       | int          | YES  |     | NULL    |       |
| DateOfBirth | date         | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
7 rows in set (0.01 sec)
```

mysql> alter table student drop DateOfBirth, drop phone;

```
mysql> ALTER TABLE student DROP DateOfBirth, drop phone;
Query OK, 0 rows affected (1.68 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

mysql> desc student;

```
mysql> desc student;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| StudentID | int          | YES  |     | NULL    |       |
| LastName  | varchar(255) | YES  |     | NULL    |       |
| FirstName | varchar(255) | YES  |     | NULL    |       |
| Address   | varchar(255) | YES  |     | NULL    |       |
| City      | varchar(255) | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
5 rows in set (0.09 sec)
```

mysql> ALTER TABLE student RENAME students;

```
mysql> ALTER TABLE student RENAME students;
Query OK, 0 rows affected (0.90 sec)
```

mysql> desc student;

```
mysql> desc student;
ERROR 1146 (42S02): Table 'university.student' doesn't exist
```

mysql> desc students;

```
mysql> desc students;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| StudentID | int          | YES  |     | NULL    |       |
| LastName  | varchar(255) | YES  |     | NULL    |       |
| FirstName | varchar(255) | YES  |     | NULL    |       |
| Address   | varchar(255) | YES  |     | NULL    |       |
| City      | varchar(255) | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
5 rows in set (0.08 sec)
```

Result:

Thus the SQL commands to perform insertion, deletion, modifying, altering, updating and viewing records based on conditions are executed successfully.

Ex.No:1c	COMMIT, ROLLBACK AND SAVEPOINT SQL COMMANDS

AIM:

To execute and verify the Commit, Rollback and Savepoint commands.

SQL COMMANDS:

COMMIT command:

COMMIT command is used to permanently save any transaction into the database.

To avoid that, we use the COMMIT command to mark the changes as permanent.

Syntax:

COMMIT;

SAVEPOINT command:

SAVEPOINT command is used to temporarily save a transaction so that you can rollback to that point whenever required.

Syntax:

SAVEPOINT savepoint_name;

ROLLBACK command

This command restores the database to last commited state. It is also used with SAVEPOINT command to jump to a savepoint in an ongoing transaction.

Syntax:

ROLLBACK TO savepoint_name;

PROCEDURE

STEP 1: Start

STEP 2: Create the table with its essential attributes.

STEP 3: Commit

STEP 4: Savepoint

STEP 5: Rollback

STEP 6: Viewing records from database.

STEP 7: Stop

mysql> Create table class (id int, name varchar(255));

```
mysql> Create table class( id int, name varchar(255));
Query OK, 0 rows affected (1.46 sec)
```

mysql> START TRANSACTION;

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)
```

mysql> insert into class values (5, 'Rahul');

```
mysql> INSERT INTO class VALUES(5, 'Rahul');
Query OK, 1 row affected (0.25 sec)
```

mysql> commit;

```
mysql> commit;
Query OK, 0 rows affected (0.00 sec)
```

mysql> SET autocommit=0;

```
mysql> SET autocommit=0;
Query OK, 0 rows affected (0.00 sec)
```

mysql> update class set name = 'Abhijit' where id = '5';

```
mysql> UPDATE class SET name = 'Abhijit' WHERE id = '5';
Query OK, 1 row affected (0.13 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

mysql> insert into class values (6, 'Chris');

```
mysql> INSERT INTO class VALUES(6, 'Chris');
Query OK, 1 row affected (0.14 sec)
```

mysql> select * from class;

```
mysql> select * from class;
+------+---------+
| id   | name    |
+------+---------+
|    5 | Abhijit |
|    6 | Chris   |
+------+---------+
2 rows in set (0.06 sec)
```

mysql> rollback;

```
mysql> rollback;
Query OK, 0 rows affected (0.08 sec)
```

mysql> select * from class;

```
mysql> select * from class;
+------+-------+
| id   | name  |
+------+-------+
|    5 | Rahul |
+------+-------+
1 row in set (0.00 sec)
```

mysql> update class set name = 'Abhijit' where id = '5';

```
mysql> UPDATE class SET name = 'Abhijit' WHERE id = '5';
Query OK, 1 row affected (0.13 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

mysql> savepoint A;

```
mysql> savepoint A;
Query OK, 0 rows affected (0.00 sec)
```

mysql> insert into class values (6, 'Chris');

```
mysql> INSERT INTO class VALUES(6, 'Chris');
Query OK, 1 row affected (0.14 sec)
```

mysql> savepoint B;

```
mysql> savepoint B;
Query OK, 0 rows affected (0.00 sec)
```

mysql> select * from class;

```
mysql> select * from class;
+------+---------+
| id   | name    |
+------+---------+
|    5 | Abhijit |
|    6 | Chris   |
+------+---------+
2 rows in set (0.06 sec)
```

mysql> Rollback to A;

```
mysql> Rollback to A;
Query OK, 0 rows affected (0.00 sec)
```

mysql> select * from class;

```
mysql> select * from class;
+------+---------+
| id   | name    |
+------+---------+
|    5 | Abhijit |
+------+---------+
1 row in set (0.00 sec)
```

Result:

Thus the SQL commands to perform commit, savepoint and rollback are executed successfully.

Ex.No:2	**DATABASE QUERYING – SIMPLE QUERIES, NESTED QUERIES, SUB QUERIES AND JOINS**

Aim:

To Create Table and Apply Simple Queries Nested Queries, Sub Queries and Joins.

PROCEDURE

STEP 1: Start

STEP 2: Create two different tables with its essential attributes.

STEP 3: Insert attribute values into the table.

STEP 4: Create the Nested query from the above created table.

STEP 5: Execute Command and extract information from the tables.

STEP 6: Stop

SQL COMMANDS FOR NESTED QUERIES

1. Command Name: SELECT

Command Description: SELECT command is used to select records from the table.

2. Command Name: WHERE

Command Description: WHERE command is used to identify particular elements.

3. Command Name: HAVING

Command Description: HAVING command is used to identify particular elements.

4. Command Name: MIN (SAL)

Command Description: MIN (SAL) command is used to find minimum salary.

GENERAL SYNTAX FOR NESTED QUERY:

SELECT "COLUMN_NAME1"FROM "TABLE_NAME1"WHERE "COLUMN_NAME2"
[COMPARISON OPERATOR](SELECT "COLUMN_NAME3"FROM "TABLE_NAME2"
WHERE [CONDITION])

SYNTAX FOR NESTED QUERY STATEMENT:

SQL> SELECT <COLUMN_NAME> FROM <TABLE _1> WHERE
<COLUMN_NAME><RELATIONAL _OPERATION> ‘VALUE’
(SELECT (AGGRECATE FUNCTION) FROM <TABLE_1> WHERE

<COLUMN NAME> = ‘VALUE’

(SELECT <COLUMN_NAME> FROM <TABLE_2> WHERE

<COLUMN_NAME= ‘VALUE’));

SQL COMMANDS FOR JOIN QUERIES

1.Command Name: INNER JOIN

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

2.Command Name: LEFT JOIN

Command Description: 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).

3.Command Name : RIGHT JOIN

Command Description: 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).

4.Command Name : FULL JOIN

Command Description: The FULL JOIN keyword return rows when there is a match in one of the tables.

Syntax for left outer join:

SQL> SELECT column_name(s) FROM table_name1LEFT JOIN table_name2 ON table_name1.column_name = table_name2 .column_name

Syntax for full outer join:

SQL>SELECT column_name(s)FROM table_name1 FULL JOIN table_name2ON table_name1.column_name=table_name2 column_name.

Syntax for right outer join:

SQL>SELECT column_name(s)FROM table_name1 RIGHT JOIN table_name2 ON table_name1.column_name=table_name2

Syntax for inner join:

SQL>SELECT column_name(s) FROM table_name1 INNER JOIN table_name2 ON table_name1.column_name=table_name2.column_name

mysql> create table customers (Id int, Name varchar(255), Age int, Address varchar(255), Salary double);

mysql> insert into customers values (1, 'Ramesh', 32, 'Ahmedabad', 20000.00);

mysql> insert into customers values (2, 'Khilan', 25, 'Delhi', 15000.00);

mysql> insert into customers values (3, 'kaushik', 23, 'Kota', 20000.00);

mysql> insert into customers values (4, 'Chaitali', 25, 'Mumbai', 65000.00);

mysql> insert into customers values (5, 'Hardik', 27, 'Bhopal', 85000.00);

mysql> insert into customers values (6, 'Komal', 22, 'MP', 45000.00);

mysql> insert into customers values (7, 'Muffy', 24, 'Indore', 100000.00);

mysql> select * from customers;

```
mysql> select * from customers;
+------+----------+------+-----------+--------+
| Id   | Name     | Age  | Address   | Salary |
+------+----------+------+-----------+--------+
|    1 | Ramesh   |   32 | Ahmedabad |  20000 |
|    2 | Khilan   |   25 | Delhi     |  15000 |
|    3 | kaushik  |   23 | Kota      |  20000 |
|    4 | Chaitali |   25 | Mumbai    |  65000 |
|    5 | Hardik   |   27 | Bhopal    |  85000 |
|    6 | Komal    |   22 | MP        |  45000 |
|    7 | Muffy    |   24 | Indore    | 100000 |
+------+----------+------+-----------+--------+
7 rows in set (0.00 sec)
```

mysql> select id, name, salary from customers;

```
mysql> SELECT ID, NAME, SALARY FROM CUSTOMERS;
+------+----------+--------+
| ID   | NAME     | SALARY |
+------+----------+--------+
|    1 | Ramesh   |  20000 |
|    2 | Khilan   |  15000 |
|    3 | kaushik  |  20000 |
|    4 | Chaitali |  65000 |
|    5 | Hardik   |  85000 |
|    6 | Komal    |  45000 |
|    7 | Muffy    | 100000 |
+------+----------+--------+
7 rows in set (0.00 sec)
```

mysql> create table student (Stud_ID int, Stud_Name varchar(255), Age int, Email varchar(255), Course_Id int);

mysql> insert into student values (1,'Anjali',26,'anjali@gmail.com',10);

mysql> insert into student values (2,'Shweta',27,'shweta@gmail.com',30);

mysql> insert into student values (3,'Sapna',25,'sapna@gmail.com',30);

mysql> insert into student values (4,'Doli',26,'doli@gmail.com',10);

mysql> select * from student;

```
mysql> select * from student;
+---------+-----------+------+------------------+-----------+
| Stud_ID | Stud_Name | Age  | Email            | Course_Id |
+---------+-----------+------+------------------+-----------+
|       1 | Anjali    |   26 | anjali@gmail.com |        10 |
|       2 | Shweta    |   27 | shweta@gmail.com |        30 |
|       3 | Sapna     |   25 | sapna@gmail.com  |        30 |
|       4 | Doli      |   26 | doli@gmail.com   |        10 |
+---------+-----------+------+------------------+-----------+
4 rows in set (0.00 sec)
```

mysql> create table Subject (Course_Id int, Course_Name varchar(255), Faculty varchar(255));

mysql> insert into Subject values (10,'Oracle','Amit');

mysql> insert into Subject values (20,'Automata','Amit');

mysql> insert into Subject values (30,'Network','Kumar');

mysql> insert into Subject values (40,'Unix','Kumar');

mysql> select * from Subject;

```
mysql> select * from Subject;
+-----------+-------------+---------+
| Course_Id | Course_Name | Faculty |
+-----------+-------------+---------+
|        10 | Oracle      | Amit    |
|        20 | Automata    | Amit    |
|        30 | Network     | Kumar   |
|        40 | Unix        | Kumar   |
+-----------+-------------+---------+
4 rows in set (0.00 sec)
```

1. Sub Query using WHERE Clause

mysql> SELECT * FROM student WHERE course_id in (SELECT course_id FROM subject WHERE course_name = ‘Oracle');

```
mysql> SELECT * FROM student WHERE course_id in (SELECT course_id FROM subject WHERE course_name ='Oracle');
+---------+-----------+------+------------------+-----------+
| Stud_ID | Stud_Name | Age  | Email            | Course_Id |
+---------+-----------+------+------------------+-----------+
|       1 | Anjali    |   26 | anjali@gmail.com |        10 |
|       4 | Doli      |   26 | doli@gmail.com   |        10 |
+---------+-----------+------+------------------+-----------+
2 rows in set (0.06 sec)
```

2. Sub Query using FROM Clause

mysql> SELECT a.course_name, b.Avg_Age FROM subject a, (SELECT course_id, Avg(Age) as Avg_Age FROM student GROUP BY course_id) b WHERE b.course_id = a.course_id;

```
mysql> SELECT a.course_name, b.Avg_Age FROM subject a, (SELECT course_id, Avg(Age) as Avg_Age FROM student GROUP BY cour
se_id) b WHERE b.course_id = a.course_id;
+-------------+---------+
| course_name | Avg_Age |
+-------------+---------+
| Oracle      | 26.0000 |
| Network     | 26.0000 |
+-------------+---------+
2 rows in set (0.06 sec)
```

3. Sub Query using SELECT Clause

mysql> SELECT course_id, course_name, (SELECT count(course_id) as num_of_student FROM student a WHERE a.course_id = b.course_id GROUP BY course_id) No_of_Students FROM subject b;

```
mysql> SELECT course_id, course_name, (SELECT count(course_id) as num_of_student FROM student a WHERE  a.course_id = b.c
ourse_id GROUP BY course_id) No_of_Students FROM subject b;
+-----------+-------------+-----------------+
| course_id | course_name | No_of_Students  |
+-----------+-------------+-----------------+
|        10 | Oracle      |               2 |
|        20 | Automata    |            NULL |
|        30 | Network     |               2 |
|        40 | Unix        |            NULL |
+-----------+-------------+-----------------+
4 rows in set (0.00 sec)
```

```
mysql> CREATE TABLE employees (
        employee_id INT (11) UNSIGNED NOT NULL,
        first_name VARCHAR(20),
        last_name VARCHAR(25) NOT NULL,
        email VARCHAR(25) NOT NULL,
        phone_number VARCHAR(20),
        hire_date DATE NOT NULL,
        job_id VARCHAR(10) NOT NULL,
        salary DECIMAL(8, 2) NOT NULL,
        commission_pct DECIMAL(2, 2),
        manager_id INT (11) UNSIGNED,
        department_id INT (11) UNSIGNED,
        PRIMARY KEY (employee_id)
        );
mysql> CREATE TABLE departments (
        department_id INT (11) UNSIGNED NOT NULL,
        department_name VARCHAR(30) NOT NULL,
        manager_id INT (11) UNSIGNED,
        location_id INT (11) UNSIGNED,
```

PRIMARY KEY (department_id)

);

mysql> insert into employees values (100, 'Steven', 'King', 'SKING', 515.123.4567', STR_TO_DATE ('17-JUN-1987', '%d-%M-%Y'), 'AD_PRES', 24000, NULL, NULL, 90);

mysql> insert into employees values (101,'Neena','Kochhar','NKOCHHAR','515.123.4568',STR_TO_DATE('21-SEP-1989', '%d-%M-%Y'),'AD_VP',17000,NULL,100,90);

mysql> INSERT INTO employees VALUES (102,'Lex','De Haan','LDEHAAN','515.123.4569',STR_TO_DATE('13-JAN-1993', '%d-%M-%Y'), 'AD_VP',17000,NULL,100,90);

mysql> INSERT INTO employees VALUES (103,'Alexander','Hunold','AHUNOLD','590.423.4567',STR_TO_DATE('03-JAN-1990', '%d-%M-%Y'),'IT_PROG',9000,NULL,102,60);

mysql> INSERT INTO employees VALUES (104,'Bruce','Ernst','BERNST', '590.423.4568',STR_TO_DATE('21-MAY-1991', '%d-%M-%Y'),'IT_PROG',6000,NULL,103,60);

mysql> INSERT INTO employees VALUES (105,'David','Austin','DAUSTIN','590.423.4569',STR_TO_DATE('25-JUN-1997', '%d-%M-%Y'), 'IT_PROG',4800,NULL,103,60);

mysql> select * from employees;

```
mysql> select * from employees;
+-------------+------------+-----------+----------+--------------+------------+---------+----------+----------------+---
---------+---------------+
| employee_id | first_name | last_name | email    | phone_number | hire_date  | job_id  | salary   | commission_pct | ma
nager_id | department_id |
+-------------+------------+-----------+----------+--------------+------------+---------+----------+----------------+---
---------+---------------+
|         100 | Steven     | King      | SKING    | 515.123.4567 | 1987-06-17 | AD_PRES | 24000.00 |           NULL |
   NULL |            90 |
|         101 | Neena      | Kochhar   | NKOCHHAR | 515.123.4568 | 1989-09-21 | AD_VP   | 17000.00 |           NULL |
    100 |            90 |
|         102 | Lex        | De Haan   | LDEHAAN  | 515.123.4569 | 1993-01-13 | AD_VP   | 17000.00 |           NULL |
    100 |            90 |
|         103 | Alexander  | Hunold    | AHUNOLD  | 590.423.4567 | 1990-01-03 | IT_PROG |  9000.00 |           NULL |
    102 |            60 |
|         104 | Bruce      | Ernst     | BERNST   | 590.423.4568 | 1991-05-21 | IT_PROG |  6000.00 |           NULL |
    103 |            60 |
|         105 | David      | Austin    | DAUSTIN  | 590.423.4569 | 1997-06-25 | IT_PROG |  4800.00 |           NULL |
    103 |            60 |
+-------------+------------+-----------+----------+--------------+------------+---------+----------+----------------+---
---------+---------------+
6 rows in set (0.06 sec)
```

mysql> INSERT INTO departments VALUES (10,'Administration',200,1700);

mysql> INSERT INTO departments VALUES (20,'Marketing',201,1800);

mysql> INSERT INTO departments VALUES (30,'Purchasing',114,1700);

mysql> INSERT INTO departments VALUES (40,'Human Resources',203,2400);

mysql> INSERT INTO departments VALUES (50,'Shipping',121,1500);

mysql> INSERT INTO departments VALUES (60,'IT',103,1400);

mysql> INSERT INTO departments VALUES (70,'Public Relations',204,2700);

mysql> INSERT INTO departments VALUES (80,'Sales',145,2500);

mysql> select * from departments;

```
mysql> select * from departments;
+---------------+--------------------+------------+-------------+
| department_id | department_name    | manager_id | location_id |
+---------------+--------------------+------------+-------------+
|            10 | Administration     |        200 |        1700 |
|            20 | Marketing          |        201 |        1800 |
|            30 | Purchasing         |        114 |        1700 |
|            40 | Human Resources    |        203 |        2400 |
|            50 | Shipping           |        121 |        1500 |
|            60 | IT                 |        103 |        1400 |
|            70 | Public Relations   |        204 |        2700 |
|            80 | Sales              |        145 |        2500 |
+---------------+--------------------+------------+-------------+
8 rows in set (0.00 sec)
```

mysql> select * from employees where salary = (select min(salary) from employees);

```
mysql> SELECT * FROM employees WHERE salary = (SELECT MIN(salary) FROM employees);
+-------------+------------+-----------+--------+--------------+------------+--------+----------------+-----
-------+---------------+
| employee_id | first_name | last_name | email  | phone_number | hire_date  | job_id | salary  | commission_pct | mana
ger_id | department_id |
+-------------+------------+-----------+--------+--------------+------------+--------+---------+----------------+-----
-------+---------------+
|         105 | David      | Austin    | DAUSTIN | 590.423.4569 | 1997-06-25 | IT_PROG | 4800.00 |           NULL |
   103 |            60 |
+-------------+------------+-----------+--------+--------------+------------+--------+---------+----------------+-----
-------+---------------+
1 row in set (0.05 sec)
```

mysql> SELECT e.first_name, e.salary FROM employees e WHERE salary IN (SELECT MIN(e.salary) FROM employees e GROUP BY e.department_id);

```
mysql> SELECT e.first_name, e.salary FROM employees e WHERE salary IN ( SELECT MIN(e.salary) FROM employees e GROUP BY e
.department_id);
+------------+----------+
| first_name | salary   |
+------------+----------+
| Neena      | 17000.00 |
| Lex        | 17000.00 |
| David      |  4800.00 |
+------------+----------+
3 rows in set (0.05 sec)
```

mysql> SELECT e.first_name, e.salary, e.department_id, b.salary_avg FROM employees e, (SELECT e1.department_id, AVg(e1.salary) salary_avg FROM employees e1 GROUP BY e1.department_id) b WHERE e.department_id = b.department_id AND e.salary > b.salary_avg;

```
mysql> SELECT e.first_name, e.salary, e.department_id, b.salary_avg FROM employees e, (SELECT e1.department_id, AVg(e1.s
alary) salary_avg FROM employees e1 GROUP BY e1.department_id) b WHERE e.department_id = b.department_id AND e.salary >
b.salary_avg;
+------------+----------+---------------+-------------+
| first_name | salary   | department_id | salary_avg  |
+------------+----------+---------------+-------------+
| Steven     | 24000.00 |            90 | 19333.333333 |
| Alexander  |  9000.00 |            60 |  6600.000000 |
+------------+----------+---------------+-------------+
2 rows in set (0.00 sec)
```

Joins:

mysql> CREATE TABLE Customer (Cust_id int NOT NULL, Cust_name varchar(20), Country varchar(20), Receipt_no int,Order_id int NOT NULL);

mysql> CREATE TABLE Orders (Order_id int, Item_ordered varchar(20),Order_date date);

mysql> insert into customer values(111,'PPPP','USA',113,1);

mysql> insert into customer values(112,'AAAA','UK',115,2);

mysql> insert into customer values(113,'BBBB','Australia',116,5);

mysql> insert into customer values(114,'CCCC','England',112,1);

mysql> insert into customer values(115,'DDDD','Germany',111,4);

mysql> insert into customer values(116,'EEEE','Dubai',114,7);

mysql> select * from customer;

```
mysql> select * from customer;
+---------+-----------+-----------+------------+----------+
| Cust_id | Cust_name | Country   | Receipt_no | Order_id |
+---------+-----------+-----------+------------+----------+
|     111 | PPPP      | USA       |        113 |        1 |
|     112 | AAAA      | UK        |        115 |        2 |
|     113 | BBBB      | Australia |        116 |        5 |
|     114 | CCCC      | England   |        112 |        1 |
|     115 | DDDD      | Germany   |        111 |        4 |
|     116 | EEEE      | Dubai     |        114 |        7 |
+---------+-----------+-----------+------------+----------+
6 rows in set (0.00 sec)
```

mysql> insert into Orders values(1,'talc','2007-12-24');

mysql> insert into Orders values(2,'Soap','2001-08-13');

mysql> insert into Orders values(3,'Deo Spray','2005-03-19');

mysql> insert into Orders values(4,'hair Oil','2012-11-05');

mysql> select * from Orders;

```
mysql> select * from Orders;
+----------+--------------+------------+
| Order_id | Item_ordered | Order_date |
+----------+--------------+------------+
|        1 | talc         | 2007-12-24 |
|        2 | Soap         | 2001-08-13 |
|        3 | Deo Spray    | 2005-03-19 |
|        4 | hair Oil     | 2012-11-05 |
+----------+--------------+------------+
4 rows in set (0.00 sec)
```

1. Using Clause

To join a table using the USING Clause we write the following command.

mysql> SELECT Cust_id, Cust_name, Country, item_Ordered, Order_date FROM Customer C JOIN Orders O USING (Order_id);

```
mysql> SELECT Cust_id, Cust_name, Country, item_Ordered, Order_date FROM Customer C JOIN Orders O USING (Order_id);
+---------+-----------+---------+--------------+------------+
| Cust_id | Cust_name | Country | item_Ordered | Order_date |
+---------+-----------+---------+--------------+------------+
|     111 | PPPP      | USA     | talc         | 2007-12-24 |
|     112 | AAAA      | UK      | Soap         | 2001-08-13 |
|     114 | CCCC      | England | talc         | 2007-12-24 |
|     115 | DDDD      | Germany | hair Oil     | 2012-11-05 |
+---------+-----------+---------+--------------+------------+
4 rows in set (0.00 sec)
```

2. On Clause

To join a table using an ON Clause we write the following command.

mysql> SELECT Cust_id, Cust_name, Country, item_Ordered, Order_date FROM Customer C JOIN Orders O on (C.Order_id = O.Order_id);

```
mysql> SELECT Cust_id, Cust_name, Country, item_Ordered, Order_date FROM Customer C JOIN Orders O on (C.Order_id = O.Ord
er_id);
+---------+-----------+---------+--------------+------------+
| Cust_id | Cust_name | Country | item_Ordered | Order_date |
+---------+-----------+---------+--------------+------------+
|     111 | PPPP      | USA     | talc         | 2007-12-24 |
|     112 | AAAA      | UK      | Soap         | 2001-08-13 |
|     114 | CCCC      | England | talc         | 2007-12-24 |
|     115 | DDDD      | Germany | hair Oil     | 2012-11-05 |
+---------+-----------+---------+--------------+------------+
4 rows in set (0.00 sec)
```

Equi Join

An Equi join is used to get the data from multiple tables where the names are common and the columns are specified. It includes the equal ("=") operator.

mysql> SELECT Cust_id, Cust_name, item_Ordered, Order_date FROM Customer C, Orders O WHERE C.Order_id = O.Order_id;

```
mysql> SELECT Cust_id, Cust_name, item_Ordered, Order_date FROM Customer C, Orders O WHERE C.Order_id = O.Order_id;
+---------+-----------+--------------+------------+
| Cust_id | Cust_name | item_Ordered | Order_date |
+---------+-----------+--------------+------------+
|     111 | PPPP      | talc         | 2007-12-24 |
|     112 | AAAA      | Soap         | 2001-08-13 |
|     114 | CCCC      | talc         | 2007-12-24 |
|     115 | DDDD      | hair Oil     | 2012-11-05 |
+---------+-----------+--------------+------------+
4 rows in set (0.00 sec)
```

Inner Join

An Inner Join retrieves the matching records; in other words, it retrieves all the rows where there is at least one match in the tables.

mysql> SELECT Cust_id, Cust_name, Country, item_ordered, Order_date FROM Customer INNER JOIN Orders USING (Order_id);

```
mysql> SELECT Cust_id, Cust_name, Country, item_ordered, Order_date FROM Customer INNER JOIN Orders USING (Order_id);
+---------+-----------+---------+--------------+------------+
| Cust_id | Cust_name | Country | item_ordered | Order_date |
+---------+-----------+---------+--------------+------------+
|     111 | PPPP      | USA     | talc         | 2007-12-24 |
|     112 | AAAA      | UK      | Soap         | 2001-08-13 |
|     114 | CCCC      | England | talc         | 2007-12-24 |
|     115 | DDDD      | Germany | hair Oil     | 2012-11-05 |
+---------+-----------+---------+--------------+------------+
4 rows in set (0.00 sec)
```

Outer Join

The records that don't match will be retrieved by the Outer join. It is of the following three types:

1. Left Outer Join

2. Right Outer Join

3. Full Outer Join

1. Left Outer Join

A Left outer join retrieves all records from the left hand side of the table with all the matched records.

mysql> SELECT Cust_id, Cust_name, Country, item_ordered, Order_date FROM customer C LEFT OUTER JOIN Orders O ON (C. Order_id = O.Order_id);

```
mysql> SELECT Cust_id, Cust_name, Country, item_ordered, Order_date FROM customer C LEFT OUTER JOIN Orders O ON (C. Orde
r_id = O.Order_id);
+---------+-----------+-----------+--------------+------------+
| Cust_id | Cust_name | Country   | item_ordered | Order_date |
+---------+-----------+-----------+--------------+------------+
|     111 | PPPP      | USA       | talc         | 2007-12-24 |
|     112 | AAAA      | UK        | Soap         | 2001-08-13 |
|     113 | BBBB      | Australia | NULL         | NULL       |
|     114 | CCCC      | England   | talc         | 2007-12-24 |
|     115 | DDDD      | Germany   | hair Oil     | 2012-11-05 |
|     116 | EEEE      | Dubai     | NULL         | NULL       |
+---------+-----------+-----------+--------------+------------+
6 rows in set (0.00 sec)
```

2. Right Outer Join

A Right Outer Join retrieves the records from the right hand side columns.

mysql> SELECT Cust_id, Cust_name, Country, item_ordered, Order_date FROM customer C RIGHT OUTER JOIN Orders O ON (C. Order_id = O.Order_id) ;

```
mysql> SELECT Cust_id, Cust_name, Country, item_ordered, Order_date FROM customer C RIGHT OUTER JOIN Orders O ON (C. Ord
er_id = O.Order_id) ;
+---------+-----------+---------+--------------+------------+
| Cust_id | Cust_name | Country | Item_ordered | Order_date |
+---------+-----------+---------+--------------+------------+
|     111 | PPPP      | USA     | talc         | 2007-12-24 |
|     114 | CCCC      | England | talc         | 2007-12-24 |
|     112 | AAAA      | UK      | Soap         | 2001-08-13 |
|    NULL | NULL      | NULL    | Deo Spray    | 2005-03-19 |
|     115 | DDDD      | Germany | hair Oil     | 2012-11-05 |
+---------+-----------+---------+--------------+------------+
5 rows in set (0.00 sec)
```

3. Full Outer Join

To retrieve all the records, both matching and unmatched from all the tables then use the FULL OUTER JOIN.

mysql> SELECT Cust_id, Cust_name, Country, item_ordered, Order_date FROM customer C FULL OUTER JOIN Orders O where (C. Order_id = O.Order_id);

Non-Equi Join

A Non-Equi join is based on a condition using an operator other than equal to "=".

mysql> SELECT Cust_id, Cust_name, Country, Item_ordered, Order_date FROM Customer C, Orders O WHERE C. Order_id > O.Order_id;

```
mysql> SELECT Cust_id, Cust_name, Country, Item_ordered, Order_date FROM Customer C, Orders O WHERE C. Order_id > O.Or
r_id;
+---------+-----------+-----------+--------------+------------+
| Cust_id | Cust_name | Country   | Item_ordered | Order_date |
+---------+-----------+-----------+--------------+------------+
|     112 | AAAA      | UK        | talc         | 2007-12-24 |
|     113 | BBBB      | Australia | talc         | 2007-12-24 |
|     113 | BBBB      | Australia | Soap         | 2001-08-13 |
|     113 | BBBB      | Australia | Deo Spray    | 2005-03-19 |
|     113 | BBBB      | Australia | hair Oil     | 2012-11-05 |
|     115 | DDDD      | Germany   | talc         | 2007-12-24 |
|     115 | DDDD      | Germany   | Soap         | 2001-08-13 |
|     115 | DDDD      | Germany   | Deo Spray    | 2005-03-19 |
|     116 | EEEE      | Dubai     | talc         | 2007-12-24 |
|     116 | EEEE      | Dubai     | Soap         | 2001-08-13 |
|     116 | EEEE      | Dubai     | Deo Spray    | 2005-03-19 |
|     116 | EEEE      | Dubai     | hair Oil     | 2012-11-05 |
+---------+-----------+-----------+--------------+------------+
12 rows in set (0.00 sec)
```

Natural Join

A natural join is just like an equi-join since it compares the common columns of both tables.

mysql> SELECT Cust_id, Cust_name, Country, Item_ordered, Order_date FROM Customer NATURAL JOIN Orders;

```
mysql> SELECT Cust_id, Cust_name, Country, Item_ordered, Order_date FROM Customer NATURAL JOIN Orders;
+---------+-----------+---------+--------------+------------+
| Cust_id | Cust_name | Country | Item_ordered | Order_date |
+---------+-----------+---------+--------------+------------+
|     111 | PPPP      | USA     | talc         | 2007-12-24 |
|     112 | AAAA      | UK      | Soap         | 2001-08-13 |
|     114 | CCCC      | England | talc         | 2007-12-24 |
|     115 | DDDD      | Germany | hair Oil     | 2012-11-05 |
+---------+-----------+---------+--------------+------------+
4 rows in set (0.00 sec)
```

Cross Join

This join is a little bit different from the other joins since it generates the Cartesian product of two tables.

mysql> SELECT Cust_id, Cust_name, Country, Item_ordered, Order_date FROM Customer CROSS JOIN Orders;

```
mysql> SELECT Cust_id, Cust_name, Country, Item_ordered, Order_date FROM Customer CROSS JOIN Orders;
+---------+-----------+-----------+--------------+------------+
| Cust_id | Cust_name | Country   | Item_ordered | Order_date |
+---------+-----------+-----------+--------------+------------+
|     111 | PPPP      | USA       | talc         | 2007-12-24 |
|     111 | PPPP      | USA       | Soap         | 2001-08-13 |
|     111 | PPPP      | USA       | Deo Spray    | 2005-03-19 |
|     111 | PPPP      | USA       | hair Oil     | 2012-11-05 |
|     112 | AAAA      | UK        | talc         | 2007-12-24 |
|     112 | AAAA      | UK        | Soap         | 2001-08-13 |
|     112 | AAAA      | UK        | Deo Spray    | 2005-03-19 |
|     112 | AAAA      | UK        | hair Oil     | 2012-11-05 |
|     113 | BBBB      | Australia | talc         | 2007-12-24 |
|     113 | BBBB      | Australia | Soap         | 2001-08-13 |
|     113 | BBBB      | Australia | Deo Spray    | 2005-03-19 |
|     113 | BBBB      | Australia | hair Oil     | 2012-11-05 |
|     114 | CCCC      | England   | talc         | 2007-12-24 |
|     114 | CCCC      | England   | Soap         | 2001-08-13 |
|     114 | CCCC      | England   | Deo Spray    | 2005-03-19 |
|     114 | CCCC      | England   | hair Oil     | 2012-11-05 |
|     115 | DDDD      | Germany   | talc         | 2007-12-24 |
|     115 | DDDD      | Germany   | Soap         | 2001-08-13 |
|     115 | DDDD      | Germany   | Deo Spray    | 2005-03-19 |
|     115 | DDDD      | Germany   | hair Oil     | 2012-11-05 |
|     116 | EEEE      | Dubai     | talc         | 2007-12-24 |
|     116 | EEEE      | Dubai     | Soap         | 2001-08-13 |
|     116 | EEEE      | Dubai     | Deo Spray    | 2005-03-19 |
|     116 | EEEE      | Dubai     | hair Oil     | 2012-11-05 |
+---------+-----------+-----------+--------------+------------+
24 rows in set (0.00 sec)
```

Result:

Thus the SQL commands to establish the relationship between the databases has been executed successfully.

Ex.No:3	**CREATION OF VIEWS, SYNONYMS, SEQUENCE, INDEXES.**

AIM:

To execute and verify the SQL commands for Views, Synonyms, Sequence and Indexes.

PROCEDURE:

STEP 1: Start

STEP 2: Create the table with its essential attributes.

STEP 3: Insert attribute values into the table.

STEP 4: Create the view from the above created table.

STEP 5: Execute different Commands and extract information from the View.

STEP 6: Use synonyms to create an alias name for the table.

STEP 7: Use sequences to order the numbering of records.

STEP 8: Use indexes to speed up the performance.

STEP 9: Stop.

VIEWS

1. Command Name: CREATE VIEW

Command Description: CREATE VIEW command is used to define a view.

2. Command Name: INSERT IN VIEW

Command Description: INSERT command is used to insert a new row into the view.

3. Command Name: DELETE IN VIEW

Command Description: DELETE command is used to delete a row from the view.

4. Command Name: UPDATE OF VIEW

Command Description: UPDATE command is used to change a value in a tuple without changing all values in the tuple.

a) Command Name: DROP OF VIEW

Command Description: DROP command is used to drop the view table

Syntax for creating a view:

SQL> CREATE <VIEW><VIEW NAME> AS SELECT

<COLUMN_NAME_1>, <COLUMN_NAME_2> FROM <TABLE NAME>;

(or)

SQL>CREATE [OR REPLACE] VIEW <VIEW NAME>AS SELECT <COLUMN NAME1>…..FROM <TABLE ANME>;

Syntax for inserting into view:

SQL> INSERT INTO <VIEW_NAME> (COLUMN NAME1,……)

VALUES(VALUE1,….);

Syntax for deleting a column from a view:

SQL> DELETE <VIEW_NAME>WHERE <COLUMN NAME> =’VALUE’;

Syntax for updating a view:

SQL>UPDATE <VIEW_NAME> SET< COLUMN NAME> = <COLUMN NAME> +<VIEW> WHERE <COLUMNNAME>=VALUE;

Syntax for deleting a view:

SQL> DROP VIEW <VIEW_NAME>

Syntax for creating a synonym:

SQL>CREATE SYNONYM synonym-name FOR table-name;

Syntax for creating a sequence:

CREATE SEQUENCE sequence_name

MINVALUE value

MAXVALUE value

START WITH value

INCREMENT BY value

CACHE value;

Syntax for creating an index:

CREATE INDEX index_name

ON table_name (column_name)

mysql> select * from customer

```
mysql> select * from customers;
+------+----------+------+-----------+--------+
| Id   | Name     | Age  | Address   | Salary |
+------+----------+------+-----------+--------+
|    1 | Ramesh   |   32 | Ahmedabad |  20000 |
|    2 | Khilan   |   25 | Delhi     |  15000 |
|    3 | kaushik  |   23 | Kota      |  20000 |
|    4 | Chaitali |   25 | Mumbai    |  65000 |
|    5 | Hardik   |   27 | Bhopal    |  85000 |
|    6 | Komal    |   22 | MP        |  45000 |
|    7 | Muffy    |   24 | Indore    | 100000 |
+------+----------+------+-----------+--------+
7 rows in set (0.07 sec)
```

mysql> create view customers_view as select name, age from customers;

```
mysql> create view customers_view as select name, age from customers;
Query OK, 0 rows affected (0.40 sec)
```

mysql> select * from customers_view;

```
mysql> select * from customers_view;
+----------+------+
| name     | age  |
+----------+------+
| Ramesh   |   32 |
| Khilan   |   25 |
| kaushik  |   23 |
| Chaitali |   25 |
| Hardik   |   27 |
| Komal    |   22 |
| Muffy    |   24 |
+----------+------+
7 rows in set (0.09 sec)
```

mysql> UPDATE CUSTOMERS_VIEW SET AGE = 35 WHERE name = 'Ramesh';

```
mysql>  UPDATE CUSTOMERS_VIEW SET AGE = 35 WHERE name = 'Ramesh';
Query OK, 1 row affected (0.14 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

mysql> select * from customers_view;

```
mysql> select * from customers_view;
+----------+------+
| name     | age  |
+----------+------+
| Ramesh   |   35 |
| Khilan   |   25 |
| kaushik  |   23 |
| Chaitali |   25 |
| Hardik   |   27 |
| Komal    |   22 |
| Muffy    |   24 |
+----------+------+
7 rows in set (0.00 sec)
```

mysql> delete from customers_view where age = 22;

```
mysql> delete from customers_view where age = 22;
Query OK, 1 row affected (0.21 sec)
```

mysql> select * from customers_view;

```
mysql> select * from customers_view;
+----------+------+
| name     | age  |
+----------+------+
| Ramesh   |   35 |
| Khilan   |   25 |
| kaushik  |   23 |
| Chaitali |   25 |
| Hardik   |   27 |
| Muffy    |   24 |
+----------+------+
6 rows in set (0.01 sec)
```

mysql> drop view customers_view;

```
mysql> drop view customers_view;
Query OK, 0 rows affected (0.68 sec)
```

mysql> select * from customers_view;

```
mysql> select * from customers_view;
ERROR 1146 (42S02): Table 'university.customers_view' doesn't exist
```

mysql> CREATE TABLE employee1 (emp_no INT AUTO_INCREMENT PRIMARY KEY, first_name VARCHAR(50), last_name VARCHAR(50));

mysql> INSERT INTO employee1(first_name,last_name) VALUES('John','Doe');

mysql> INSERT INTO employee1(first_name,last_name) VALUES ('Mary','Jane');

mysql> SELECT * FROM employee1;

```
mysql> SELECT * FROM employee1;
+--------+------------+-----------+
| emp_no | first_name | last_name |
+--------+------------+-----------+
|      1 | John       | Doe       |
|      2 | Mary       | Jane      |
+--------+------------+-----------+
2 rows in set (0.00 sec)
```

mysql> DELETE FROM employee1 WHERE emp_no = 2;

```
mysql> DELETE FROM employee1 WHERE emp_no = 2;
Query OK, 1 row affected (0.16 sec)
```

mysql> SELECT * FROM employee1;

```
mysql> SELECT * FROM employee1;
+--------+------------+-----------+
| emp_no | first_name | last_name |
+--------+------------+-----------+
|      1 | John       | Doe       |
+--------+------------+-----------+
1 row in set (0.00 sec)
```

mysql> INSERT INTO employee1(first_name,last_name) VALUES ('Jack','Lee');

mysql> SELECT * FROM employee1;

```
mysql> SELECT * FROM employee1;
+--------+------------+-----------+
| emp_no | first_name | last_name |
+--------+------------+-----------+
|      1 | John       | Doe       |
|      3 | Jack       | Lee       |
+--------+------------+-----------+
2 rows in set (0.06 sec)
```

mysql> UPDATE employee1 SET first_name = 'Joe', emp_no = 10 WHERE emp_no = 3;

mysql> SELECT * FROM employee1;

```
mysql> SELECT * FROM employee1;
+--------+------------+-----------+
| emp_no | first_name | last_name |
+--------+------------+-----------+
|      1 | John       | Doe       |
|     10 | Joe        | Lee       |
+--------+------------+-----------+
2 rows in set (0.00 sec)
```

mysql> create index myIndex on student(Stud_Name);

```
mysql> create index myIndex on student(Stud_Name);
Query OK, 0 rows affected (1.62 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

mysql> create index myCompIndex on student(Age,Email);

```
mysql> create index myCompIndex on student(Age,Email);
Query OK, 0 rows affected (0.43 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

mysql> create unique index myIndex1 on student(Stud_Name);

```
mysql> create unique index myIndex on student(Stud_Name);
ERROR 1061 (42000): Duplicate key name 'myIndex'
mysql> create unique index myIndex1 on student(Stud_Name);
Query OK, 0 rows affected (0.69 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

mysql> drop index myIndex1 on student;

```
mysql> drop index myIndex1 on student;
Query OK, 0 rows affected (1.30 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

mysql> CREATE TABLE student_table(Reg_No int,NAME varchar(5),MARK int);

mysql> insert into student_table values(10001,'ram',85);

mysql> insert into student_table values(10002,'sam',75);

mysql> insert into student_table values(10003,'samu',95);

mysql> select * from STUDENT_TABLE;

```
mysql> select * from STUDENT_TABLE;
+--------+------+------+
| Reg_No | NAME | MARK |
+--------+------+------+
|  10001 | ram  |   85 |
|  10002 | sam  |   75 |
|  10003 | samu |   95 |
+--------+------+------+
3 rows in set (0.00 sec)
```

sql> CREATE SYNONYM STUDENT_SYNONYM FOR STUDENT_TABLE;

Synonym created.

sql> select * from STUDENT_SYNONYM;

```
+--------+------+------+
| Reg_No | NAME | MARK |
+--------+------+------+
|  10001 | ram  |   85 |
|  10002 | sam  |   75 |
|  10003 | samu |   95 |
+--------+------+------+
```

sql> insert into student_SYNONYM values(10006,'RAJA',80);

1 row created.

sql> select * from STUDENT_TABLE;

```
+--------+------+------+
| Reg_No | NAME | MARK |
+--------+------+------+
|  10001 | ram  |   85 |
|  10002 | sam  |   75 |
|  10003 | samu |   95 |
|  10006 | RAJA |   80 |
+--------+------+------+
```

sql> select * from STUDENT_SYNONYM;

Reg_No	NAME	MARK
10001	ram	85
10002	sam	75
10003	samu	95
10006	RAJA	80

sql>UPDATE STUDENT_SYNONYM SET MARK=100 WHERE REG_NO=10006;

1 row updated.

sql>select * from STUDENT_SYNONYM;

Reg_No	NAME	MARK
10001	ram	85
10002	sam	75
10003	samu	95
10006	RAJA	100

sql>select * from STUDENT_TABLE;

Reg_No	NAME	MARK
10001	ram	85
10002	sam	75
10003	samu	95
10006	RAJA	100

sql>DROP SYNONYM STUDENT_SYNONYM;

Synonym dropped.

sql> select * from STUDENT_TABLE;

Reg_No	NAME	MARK
10001	ram	85
10002	sam	75
10003	samu	95
10006	RAJA	100

Result:

Thus the SQL commands for Views, synonyms, sequence and indexes has been verified and executed successfully.

Ex.No:4	**CREATING AN EMPLOYEE DATABASE TO SET VARIOUS CONSTRAINTS IN RDBMS**

AIM:

Write an sql query:

1.To differentiate between self referential constraints and foreign key constraint.

2.To refer a field of a given table or another table by using foreign key.

3.To apply check constraint & default constraint in an effective manner.

ALGORITHM:

STEP 1: Start the DMBS.

STEP 2: Connect to the existing database (DB)

STEP 3: Create the table with its essential constraint.

STEP 4: Insert record values into the table and then check the constraint.

STEP 5: disable the constraints and insert the values into the table.

STEP 6: if you want to re-enable the constraint then enable you can do.

STEP 7: Stop the DBMS.

CONSTRAINTS

Constraints are part of the table definition that limits and restriction on the value entered into its columns.

INTEGRITY CONSTRAINT

An integrity constraint is a mechanism used by oracle to prevent invalid data entry into the table. It has enforcing the rules for the columns in a table.

The types of the integrity constraints are:

a) Domain Integrity

b) Entity Integrity

c) Referential Integrity

TYPES OF CONSTRAINTS:

1) Primary key

2) Foreign key/references

3) Check

4) Unique

5) Not null

6) Null

7) Default

CONSTRAINTS CAN BE CREATED IN THREE WAYS:

1) Column level constraints

2) Table level constraints

3) Using DDL statements-alter table command

OPERATION ON CONSTRAINT:

i) ENABLE

ii) DISABLE

iii) DROP

PRIMARY KEY CONSTRAINTS

A primary key avoids duplication of rows and does not allow null values. It can be defined on one or more columns in a table and is used to uniquely identify each row in a table. These values should never be changed and should never be null. A table should have only one primary key. If a primary key constraint is assigned to more than one column or combination of column is said to be composite primary key.

Column level constraints using primary key:

Syntax:

SQL> CREATE<OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1 <DATATYPE> (SIZE)<TYPE OF CONSTRAINTS>, COLUMN NAME.1 <DATATYPE> (SIZE) ……);

mysql> CREATE TABLE TBL_PKEY(RegNo int PRIMARY KEY, Name VARCHAR(20), ANY_SUB_MARK int);

```
Query OK, 0 rows affected (1.67 sec)
```

mysql> insert into TBL_PKEY values(10001,'raju',75);

```
Query OK, 1 row affected (0.11 sec)
```

mysql> insert into TBL_PKEY values(10002,'raj',85);

```
Query OK, 1 row affected (0.11 sec)
```

mysql> insert into TBL_PKEY values(0,'Kaj',22);

```
Query OK, 1 row affected (0.11 sec)
```

mysql> insert into TBL_PKEY values(NULL,'Kaj',22);

```
mysql> insert into TBL_PKEY values(NULL,'Kaj',22);
ERROR 1048 (23000): Column 'RegNo' cannot be null
```

mysql> insert into TBL_PKEY values(10002,'RAJAN',95);

```
ERROR 1062 (23000): Duplicate entry '10002' for key 'tbl_pkey.PRIMARY'
```

mysql> insert into TBL_PKEY values(10003,'RAJA',85);

```
Query OK, 1 row affected (0.11 sec)
```

mysql> select * from TBL_PKEY;

```
mysql> select * from TBL_PKEY;
+-------+------+--------------+
| RegNo | Name | ANY_SUB_MARK |
+-------+------+--------------+
|     0 | Kaj  |           22 |
| 10001 | raju |           75 |
| 10002 | raj  |           85 |
| 10003 | RAJA |           85 |
+-------+------+--------------+
4 rows in set (0.00 sec)
```

Table level primary key constraints:

mysql> CREATE TABLE Persons (ID int NOT NULL,LastName varchar(255) NOT NULL, FirstName varchar(255), Age int,CONSTRAINT PK_Person PRIMARY KEY (ID,LastName));

```
Query OK, 0 rows affected (1.67 sec)
```

mysql>desc person;

```
mysql> desc persons;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| ID        | int          | NO   | PRI | NULL    |       |
| LastName  | varchar(255) | NO   | PRI | NULL    |       |
| FirstName | varchar(255) | YES  |     | NULL    |       |
| Age       | int          | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
4 rows in set (0.10 sec)
```

Table level constraint with alter command (primary key):

Syntax:

SQL:>CREATE <OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1 <DATATYPE>(SIZE),

COLUMN NAME.1 <DATATYPE> (SIZE));

[OR]

SQL> ALTER TABLE <TABLE NAME> ADD CONSTRAINTS <NAME OF THECONSTRAINTS> <TYPE OF THE CONSTRAINTS><COLUMN NAME>);

mysql>CREATE TABLE EMP3(EMPNO int,ENAME VARCHAR(6),JOB VARCHAR(6),SAL int,DEPTNO int);

```
Query OK, 0 rows affected (1.67 sec)
```

mysql>desc EMP3;

```
mysql> desc EMP3;
+--------+------------+------+-----+---------+-------+
| Field  | Type       | Null | Key | Default | Extra |
+--------+------------+------+-----+---------+-------+
| EMPNO  | int        | YES  |     | NULL    |       |
| ENAME  | varchar(6) | YES  |     | NULL    |       |
| JOB    | varchar(6) | YES  |     | NULL    |       |
| SAL    | int        | YES  |     | NULL    |       |
| DEPTNO | int        | YES  |     | NULL    |       |
+--------+------------+------+-----+---------+-------+
5 rows in set (0.12 sec)
```

mysql>ALTER TABLE EMP3 ADD CONSTRAINT EMP3_EMPNO_PK PRIMARY KEY (EMPNO);

```
mysql> ALTER TABLE EMP3 ADD CONSTRAINT EMP3_EMPNO_PK PRIMARY KEY (EMPNO);
Query OK, 0 rows affected (1.65 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

mysql>desc EMP3;

```
mysql> desc EMP3;
+--------+------------+------+-----+---------+-------+
| Field  | Type       | Null | Key | Default | Extra |
+--------+------------+------+-----+---------+-------+
| EMPNO  | int        | NO   | PRI | NULL    |       |
| ENAME  | varchar(6) | YES  |     | NULL    |       |
| JOB    | varchar(6) | YES  |     | NULL    |       |
| SAL    | int        | YES  |     | NULL    |       |
| DEPTNO | int        | YES  |     | NULL    |       |
+--------+------------+------+-----+---------+-------+
5 rows in set (0.05 sec)
```

REFERENCE /FOREIGN KEY CONSTRAINT

It enforces relationship between tables. To establish parent-child relationship between 2 tables having a common column definition, we make use of this constraint. To implement this, we should define the column in the parent table as primary key and same column in the child table as foreign key referring to the corresponding parent entry.

Foreign key

A column or combination of column included in the definition of referential integrity, which would refer to a referenced key.

Referenced key

It is a unique or primary key upon which is defined on a column belonging to the parent table.

Column level foreign key constraint

Parent Table:

Syntax:

SQL:>CREATE <OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1 <DATATYPE>(SIZE)<TYPE OF CONSTRAINTS> , COLUMN NAME.1 <DATATYPE> (SIZE)……);

mysql> create table dept(deptno int primary key,dname varchar(20),location varchar(15));

```
Query OK, 0 rows affected (1.67 sec)
```

Mysql> desc dept;

```
mysql> desc dept;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| DEPTNO   | int         | NO   | PRI | NULL    |       |
| DNAME    | varchar(20) | YES  |     | NULL    |       |
| LOCATION | varchar(15) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
3 rows in set (0.14 sec)
```

Child Table:

Syntax:

SQL:>CREATE <OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1 <DATATYPE>(SIZE),

COLUMN NAME2 <DATATYPE> (SIZE) REFERENCES <TABLE NAME>(COLUMN NAME> ….);

Mysql>CREATE TABLE EMPL(EMPNO int,DEPTNO int REFERENCES DEPT(DEPTNO),DESIGN VARCHAR(10));

```
Query OK, 0 rows affected (1.67 sec)
```

Mysql> desc empl;

```
mysql> desc empl;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| EMPNO  | int         | YES  |     | NULL    |       |
| DEPTNO | int         | YES  |     | NULL    |       |
| DESIGN | varchar(10) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Mysql>insert into EMPL values(5001,101,'RAJA');

```
Query OK, 1 row affected (0.11 sec)
```

Mysql>insert into EMPL values(5003,103,'KAJA');

```
Query OK, 1 row affected (0.11 sec)
```

Column level foreign key constraint with naming conversions

Parent Table:

Syntax:

SQL :> CREATE<OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1 <DATATYPE>(SIZE)<TYPE OF CONSTRAINTS>,COLUMN NAME.1 <DATATYPE> (SIZE)…);

Child Table:

Syntax:

SQL :> CREATE<OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1 <DATATYPE>(SIZE)

COLUMN NAME2 <DATATYPE> (SIZE) CONSTRAINT <CONST.NAME>REFERENCES <TABLE NAME>(COLUMN NAME>…);

Mysql>CREATE TABLE DEPT1(DEPTNO int PRIMARY KEY,DNAME VARCHAR (20), LOCATION VARCHAR(15));

```
Query OK, 0 rows affected (1.67 sec)
```

Mysql>CREATE TABLE EMP4A (EMPNO int,DEPTNO int CONSTRAINT EMP4A_DEPTNO_FK REFERENCES DEPT1 (DEPTNO),DESIGN VARCHAR (10));

```
Query OK, 0 rows affected (1.67 sec)
```

Table level foreign key constraints:

Parent Table:

Syntax:

SQL :> CREATE<OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1 <DATATYPE>(SIZE) <TYPE OF

CONSTRAINTS>, COLUMN NAME.1 <DATATYPE> (SIZE)…);

Child Table:

Syntax:

SQL :> CREATE<OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1

<DATATYPE>(SIZE), COLUMN NAME2 <DATATYPE> (SIZE),

CONSTRAINT <CONST.NAME>REFERENCES <TABLE NAME> (COLUMN NAME>);

mysql> CREATE TABLE DEPT5(DEPTNO int PRIMARY KEY, DNAME VARCHAR(20),LOCATION VARCHAR(15));

mysql>

Ex.No:5	**DATABASE PROGRAMMING: IMPLICIT AND EXPLICIT CURSORS**

Aim:

To create a database and apply Implicit and Explicit Cursors

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

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

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

Syntax:

Declare <cursor_naem> cursor for <select statement>

open<cursor_name>;

fetch<cursor_name>into<variable1>,<variable2>;

close<cursor_name>

```
mysql> delimiter $$
mysql> create procedure curdemo(id int)
begin
declare name varchar(255);
declare cur1 cursor for select stu_name from student where stu_id=id;
 open cur1;
fetch cur1 into name;
select name;
close cur1;
end $$
```

```
Query OK, 0 rows affected (0.35 sec)
```

```
Mysql>delimiter;
Mysql> call curdemo(2);
```

```
mysql> delimiter ;
mysql> call curdemo(2);
+------+
| name |
+------+
| shah |
+------+
1 row in set (0.09 sec)

Query OK, 0 rows affected (0.10 sec)
```

Result:

Data base created and applied Implicit and Explicit Cursors in a database.

Ex.No:6	**STUDY OF PL/SQL COMMANDS**

AIM:

To study about PL/SQL and its features.

Introduction:

The PL/SQL programming language was developed by Oracle Corporation in the late 1980s as procedural extension language for SQL and the Oracle relational database.

Following are notable facts about PL/SQL:

- PL/SQL is a completely portable, high-performance transaction-processing language.
- PL/SQL provides a built-in interpreted and OS independent programming environment.
- PL/SQL can also directly be called from the command-line SQL*Plus interface.
- Direct call can also be made from external programming language calls to database.
- PL/SQL's general syntax is based on that of ADA and Pascal programming language.
- Apart from Oracle, PL/SQL is available in TimesTen in-memory database and IBM DB2.

PL/SQL stands for Procedural Language extension of SQL.

PL/SQL is a combination of SQL along with the procedural features of programming languages. It was developed by Oracle Corporation in the early 90’s to enhance the capabilities of SQL.

The PL/SQL Engine:

Oracle uses a PL/SQL engine to processes the PL/SQL statements. A PL/SQL code can be stored in the client system (client-side) or in the database (server-side).

Features of PL/SQL

- PL/SQL has the following features:
- PL/SQL is tightly integrated with SQL.
- It offers extensive error checking.
- It offers numerous data types.
- It offers a variety of programming structures.
- It supports structured programming through functions and procedures.
- It supports object-oriented programming.
- It supports developing web applications and server pages.

Advantages of PL/SQL

PL/SQL has the following advantages:

- SQL is the standard database language and PL/SQL is strongly integrated with SQL. PL/SQL supports both static and dynamic SQL. Static SQL supports DML operations and transaction control from PL/SQL block. Dynamic SQL is SQL allows embedding DDL statements in PL/SQL blocks.

- PL/SQL allows sending an entire block of statements to the database at one time. This reduces network traffic and provides high performance for the applications.
- PL/SQL gives high productivity to programmers as it can query, transform, and update data in a database.
- PL/SQL saves time on design and debugging by strong features, such as exception handling, encapsulation, data hiding, and object-oriented data types.
- Applications written in PL/SQL are fully portable.
- PL/SQL provides high security level.
- PL/SQL provides access to predefined SQL packages.
- PL/SQL provides support for Object-Oriented Programming.
- PL/SQL provides support for Developing Web Applications and Server Pages.

Result:

Thus the features and advantages of PL/SQL has been read successfully.

Ex.No:7	**PL/SQL BLOCK USING FUNCTIONS**

Aim:

To write PL/SQL block to satisfy some condition by accepting input from user.

Algorithm:

STEP 1: Start

STEP 2: Create the table with essential attributes.

STEP 3: Initialize the Function to carry out the searching procedure.

STEP 4: Frame the searching procedure for both positive and negative searching.

STEP 5: Execute the Function for both positive and negative result.

STEP 6: Stop

Program:

```
mysql>DELIMITER $$
mysql>CREATE FUNCTION F_ADD(A INTEGER, B INTEGER)
RETURNS INTEGER
BEGIN
DECLARE C INTEGER;
SET C=A+B;
RETURN C;
END;

mysql>select F_ADD(45,56);
+ ------------------ +
| F_ADD(45,56) |
+ ------------------ +
| 101 |
+ ------------------ +
1 rows
```

Function 2:

Greatest of three numbers

```
delimiter $$
CREATE FUNCTIONgreatestofthree(A INTEGER, B INTEGER, C integer)
RETURNS INTEGER
BEGIN
if A>B and A>C then
return A;
elseif B>A and B>C then
return B;
else
return C;
end if;
END;
$$
```

Output:

```
mysql> select greatestofthree(10,25,5)
+ ---------------------------- +
| greatestofthree(10,25,5) |
+ ---------------------------- +
| 25 |
+ ---------------------------- +
1 rows
```

Simple interest:

```
DELIMITER $$
CREATE FUNCTION pnr101(p float, n float, r FLOAT)
RETURNS float
BEGIN
```

```
Declare pnr float;
set pnr=p* n* r;
return pnr/100;
END;
```

Output:

```
mysql> select pnr1011(10.5,15.2,20.3)
+ -------------------------- +
| pnr1011(10.5,15.2,20.3) |
+ -------------------------- +
| 32.398799896240234 |
+ -------------------------- +
1 rows
```

Result:

Thus a PL/SQL block is written to satisfy conditions by accepting input from user was executed successfully.

Ex.No:8	**EXCEPTION HANDLING IN PL/SQL**

Aim:

To write a PL/SQL block to handle different types of exception

Program:

(1) Exception handling for 'No data found'

```
DELIMITER $$
CREATE PROCEDURE `Student_check`(IN STID INTEGER)
BEGIN
BEGIN
SELECT studid,studname,gender,percentage
FROM Student s
WHERE studid= STID;
END;
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='NO DATA FOUND';
END
```

Output:

```
CALL Student_check(21);
No data found
```

(2) Exception handling for 'No data found'

```
DELIMITER $$
CREATE PROCEDURE `Student_exist`(IN STID INTEGER)
BEGIN
Declare v_stdid integer;
BEGIN
SELECT studid
into v_stdid
FROM Student s
```

```
WHERE studid= STID;
END;
if v_stdid is null then
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='NO DATA FOUND';
else
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='DATA FOUND';
end if;
END
```

Output:

```
CALL Student_exist(21);
```

No data found

```
CALL Student_exist(201);
```

Data found

(3) Exception handling for ‘Divide by zero’

```
DELIMITER $$
CREATE PROCEDURE `div_by_zero`(IN A INTEGER,IN B INTEGER)
BEGIN
DECLARE C INTEGER;
declared iv_by_zero condition for sqlstate '22012';
declare CONTINUE handler for div_by_zero
RESIGNAL SET MESSAGE_TEXT='DIVISION BY ZER0 ERROR';
IF B=0 THEN
SIGNAL div_by_zero;
END IF;
END
```

Output:

```
mysql>CALL div_by_zero(1,1);
```

mysql>CALL div_by_zero(0,1);

Division by zero error.

(4) Exception handling for ‘reporting duplicate key values’

```
DELIMITER $$
CREATE PROCEDURE `insert_account`(in v_Accd integer,
v_Accbal integer,
v_Custid integer)
BEGIN
declare continue handler for 1062
select 'duplicate primary key found';
insert into account(Accd,Accbal,Custid) values(v_Accd,v_Accbal,v_Custid);
commit;
END
```

Output:

mysql>call insert_account(111,1000,123444);

Duplicate primary key found.

(5) Exception handling for ‘No data found’

```
DELIMITER $$
CREATE PROCEDURE `Student_exist_exception`(IN STID INTEGER)
BEGIN
declare v_stdid integer;
declare continue handler for not found
select 'NO DATA FOUND';
BEGIN
SELECT studid
intov_stdid
FROM Student s
```

```
WHERE studid= STID;
END;
END
```

Output:

```
mysql>CALL Student_exist_exception(21);
No data found
```

Result:

Thus the PL/SQL block is written to handle exceptions is written and executed successfully.

Ex.No:9	**STORED PROCEDURE IN PL/SQL**

Aim:

To create procedure in mysql

Procedure - Syntax:

Create procedure <procedure name>

Begin

<Statements>;

End;

Procedure 1:

mysql>delimiter ///

mysql>CREATE PROCEDURE Student()

BEGIN

SELECT studid,studname,gender,percentage

FROM Student s

WHERE percentage > 90;

END;

///

Output:

mysql>call student;

+ ---------- + ---------- + ---------- + ---------- +

| studid | studname | gender | percentage |

+ ---------- + ---------- + ---------- + ---------- +

| 204 | sdfds | sdfds | 95 |

+ ---------- + ---------- + ---------- + ---------- +

1 rows

Procedure 2:

Cursor:

```
mysql>delimiter $$
mysql>create procedure proname1111()
begin
declare v_empid int(10);
declare v_empname varchar(10);
declare c cursor for select empid,empname from employee WHERE departmentid=102;
OPEN C;
loop1:loop
FETCH C INTO v_empid, v_empname ;
selectv_empid , v_empname;
end loop loop1;
close c;
end;
delimiter $$;
```

output:

```
mysql> call proname1111()
+ ------------ + -------------- +
| v_empid | v_empname |
+ ------------ + -------------- +
| 4 | selvapriya |
+ ------------ + -------------- +
1 rows
```

Result:

Thus the PL/SQL block for stored procedure is written and executed successfully.

Ex.No:10	**TRIGGERS IN PL/SQL**

Aim:

To create database triggers and functions using mysql

Triggers - Syntax:

```
delimiter ///
CREATE TRIGGER <trigger_name><trigger_time><trigger_event> ON <table_name>
FOR EACH ROW
<trigger_body>
///
delimiter ;
```

Trigger1 After Insert:

```
mysql>create table foo1 (a INT, b INT, ts TIMESTAMP);
mysql>create table bar1 (a INT, b INT);
mysql>INSERT INTO foo1 (a,b) VALUES(1,1);
mysql>INSERT INTO foo1 (a,b) VALUES(2,2);
mysql>INSERT INTO foo1 (a,b) VALUES(6,6);
mysql>select * from foo1;
mysql>delimiter ///
CREATE TRIGGER ins_sum11 AFTER insert ON foo1
FOR EACH ROW
BEGIN
INSERT INTO bar1 (a, b) VALUES(NEW.a, NEW.b);
END;
///
mysql> delimiter ;
mysql>desc foo;
mysql> update foo set a=20 where b=1;
```

mysql>select * from bar1;

Functions

Syntax:

mysql>Delimiter///

Create function fn_name(param1,Param2…)

returnsdatatype

<function_body>

///

delimiter ;

Function1:

mysql>Delimiter ///

mysql>CREATE FUNCTION getBal(v_custid integer)

RETURNS FLOAT

BEGIN

DECLARE V_BAL FLOAT;

BEGIN

SELECT SUM(A.ACCBAL)

INTO V_BAL

FROM ACCOUNT A

WHERE A.CUSTID = V_CUSTID;

END;

RETURN V_BAL;

END;

///

Output:

selectgetbal(123);

+ ---------- +

| getbal(123) |

+ ---------- +

| 1000 |

+ ---------- +

1 rows

2)write a function mysql to calculate the factorial of a given no.

mysql> delimiter //

mysql> create function fact(n int)

returns integer

begin

declarefacint;

declareiint;

setfac = 1;

seti = 1;

myloop : loop

ifi> n then

leavemyloop;

else

setfac = fac * i;

seti = i + 1;

iteratemyloop;

end if;

end loop;

return (fac);

end

//

Output:

```
Query OK, 0 rows affected (0.00 sec)
mysql> select fact(3);
-> //
+---------+
| fact(3) |
+---------+
| 6 |
+---------+
```

3)write a function in mysql to calculate the sum of fibonacci series.

```
mysql> delimiter //
mysql> create function sum_fb(n int)
returnsint
begin
declare a int;
declare b int;
declare c int;
declare s int;
declareiint;
set a = 0;
set b = 1;
seti = 3;
set s = 1;
myloop : loop
ifi> n then
leave myloop;
else
set c = a + b;
set a = b;
```

```
set b = c;
set s = s + c;
seti = i + 1;
iterate myloop;
end if;
end loop;
return s;
end
//
```

Output:

```
Query OK, 0 rows affected (0.05 sec)
mysql> select sum_fb(3);
-> //
+-----------+
| sum_fb(3) |
+-----------+
| 2 |
+-----------+
1 row in set (0.03 sec)
```

4)write a function in mysql to print 'hello yourname how are you'

```
mysql> delimiter //
mysql> create function print_name(name varchar(25))
returns varchar(50)
begin
returnconcat('Hello',' ',name,' ','how are you.');
end;
//
```

Output:

```
Query OK, 0 rows affected (0.05 sec)

mysql> select print_name('janai');

-> //

+-------------------------+

| print_name('janai') |

+-------------------------+

| Hello janai how are you. |

+-------------------------+

1 row in set (0.00 sec)
```

Result:

Thus database triggers and functions are written and executed successfully.

Ex.No:11	**FRONT END TOOLS**

AIM

To design a form using different tools in Visual Basic.

PROCEDURE

STEP 1: Start

STEP 2: Create the form with essential controls in tool box.

STEP 3: Write the code for doing the appropriate functions.

STEP 4: Save the forms and project.

STEP 5: Execute the form .

STEP 6: Stop

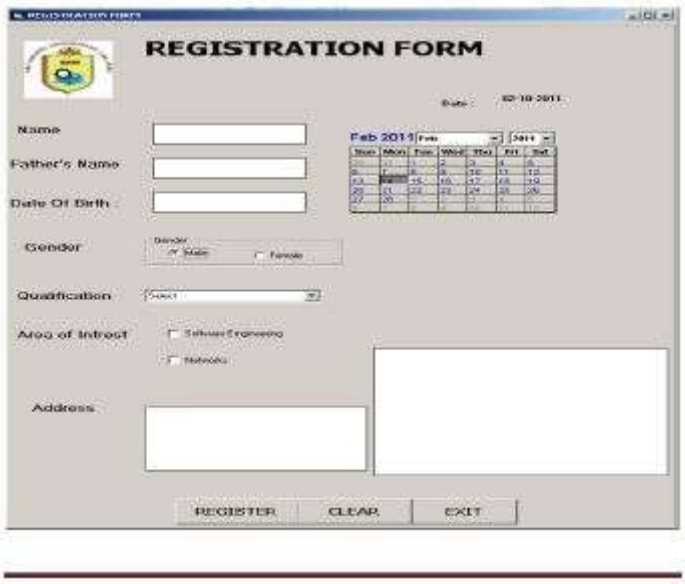
CODING:

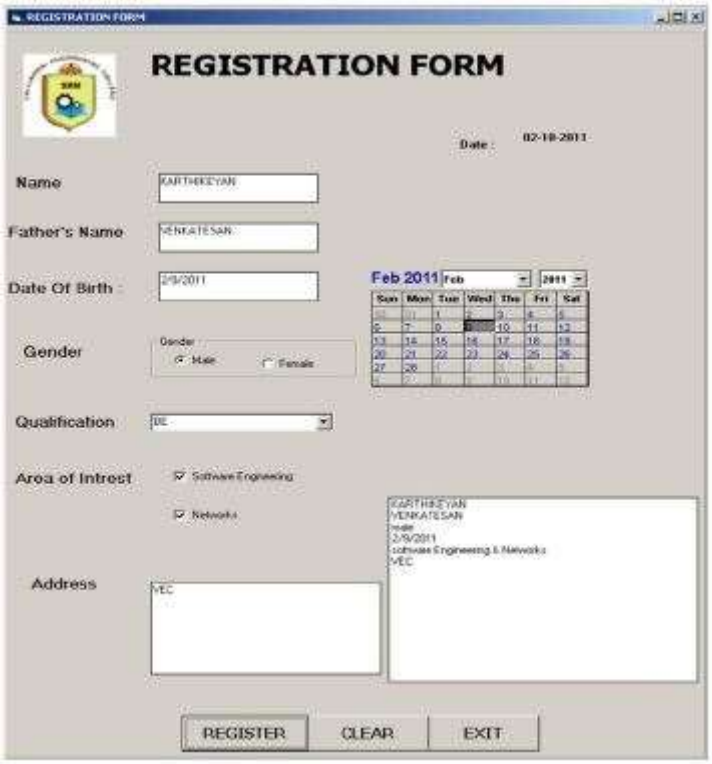
```
Private Sub Calendar1_Click()
Text3.Text = Calendar1.Value
End Sub

Private Sub Combo1_Change()
Combo1.AddItem "BSC"
Combo1.AddItem "MSC"
Combo1.AddItem "BE"
Combo1.AddItem "ME"
End Sub

Private Sub Command1_Click()
List1.AddItem Text1.Text
List1.AddItem Text2.Text
If Option1.Value = True Then
gender = "male"
```

```
End If
If Option2.Value = True Then
gender = "female"
End If
List1.AddItem gender
List1.AddItem Text3.Text
If Check1.Value = 1 And Check2.Value = 1 Then
area = "software Engineering & Networks"
End If
If Check1.Value = 0 And Check2.Value = 1 Then
area = " Networks"
End If
List1.AddItem area
List1.AddItem Text4.Text
End Sub

Private Sub Command2_Click()
End
End Sub

Private Sub Command3_Click()
If List1.ListIndex <> 0 Then
List1.RemoveItem (0)
End If
End Sub

Private Sub Form_Load()
Label10.Caption = Date$
MsgBox "Welcome to Registration"
End Sub
```

```
Private Sub Option1_Click()
If (Option1.Value = True) Then
MsgBox ("You have selected Male")
ElseIf (Option2.Value = True) Then
MsgBox ("You have selected Female")
End If
End Sub

Private Sub Option2_Click()
If (Option1.Value = True) Then
MsgBox ("You have selected Male")
ElseIf (Option2.Value = True) Then
MsgBox ("You have selected Female")
End If
End Sub
```

REGISTRATION FORM:





Result:

Thus the program has been loaded and executed successfully.