Experiment No.-01

Aim: Study of Data Base Management System (DBMS) and RDBMS.

Deffinitation of DBMS:-

A database management system (DBMS), sometimes just called a database manager, is a program that lets one or more computer users create and access data in a database. The DBMS manages user requests (and requests from other programs) so that users and other programs are free from having to understand where the data is physically located on storage media and, in a multi-user system, which else may also be accessing the data. In handling user requests, the DBMS ensures the integrity of the data (that is, making sure it continues to be accessible and is consistently organized as intended) and security (making sure only those with access privileges can access the data). The most typical DBMS is a relational database management system (RDBMS). A standard user and program interface is the Structured Query Language (SQL). A newer kind of DBMS is the object-oriented database management system (ODBMS). A DBMS can be thought of as a file manager that manages data in databases rather than files in file systems. In IBM's mainframe operating systems, the non relational data managers were (and are, because these legacy application systems are still used) known as access methods.

Features of DBMS:-

- The integration and sharing of data files minimizes the duplication and redundancy of data to a great extent.
- Integration of data files also results in a considerable saving of storage space and in data entry and data storage costs.
- Fewer application programs need to be developed for obtaining various reports due to independence of programs and data.
- The query language facility helps non-programming persons to access the database for information as needed without the help of any programmer.
- Faster preparation of information to support non-recurring tasks and changing conditions is possible.

- Updating of data becomes easier due to integration of data file. Fewer errors may when several records may be updated simultaneously.

Advantages of DBMS:-

- Controlling Data Redundancy
- Sharing of Data
- Data Consistency
- Integration of Data
- Integration Constraints
- Data Security
- Report Writers
- Control Over Concurrency
- Backup and Recovery Procedures
- Data Independence

Deffinitation of RDBMS:-

A relational database management system (RDBMS) is a program that lets you create, update, and administer a relational database. Most commercial RDBMS's use the Structured Query Language (SQL) to access the database, although SQL was invented after the development of the relational model and is not necessary for its use. The leading RDBMS products are Oracle, IBM's DB2 and Microsoft's SQL Server. Despite repeated challenges by competing technologies, as well as the claim by some experts that no current RDBMS has fully implemented relational principles, the majority of new corporate databases are still being created and managed with an RDBMS.

Features of DBMS:-

- Provides data to be stored in tables
- Persists data in the form of rows and columns
- Provides facility primary key, to uniquely identify the rows
- Creates indexes for quicker data retrieval
- Provides a virtual table creation in which sensitive data can be stored and simplified query can be applied.(views)
- Sharing a common column in two or more tables(primary key and foreign key)
- Provides multi user accessibility that can be controlled by individual users

Advantages of DBMS:-

- Data is only stored once.
- Complex queries can be carried out.
- Better security.

Experiment No.-02

Aim: To study (sql) Data Definition language Statatements and Data Manipulation Statatements.

Introduction about SQL-

SQL (Structured Questionry Language) is a nonprocedural language, you specify what you want, not how to get it. A block structured format of English key words is used in this Questionry language. It has the following components.

DDL (Data Definition Language)-

The SQL DDL provides command for defining relation schemas, deleting relations and modifying relation schema.

DML (DATA Manipulation Language)-

It includes commands to insert tuples into, delete tuples from and modify tuples in the database.

View definition-

The SQL DDL includes commands for defining views.

Transaction Control- SQL includes for specifying the beginning and ending of transactions.

Embedded SQL and Dynamic SQL-

Embedded and Dynamic SQL define how SQL statements can be embedded within general purpose programming languages, such as C, C++, JAVA, COBOL, Pascal and Fortran.

Integrity-

The SQL DDL includes commands for specifying integrity constraints that the data stored in the database must specify. Updates that violate integrity constraints are allowed.

Authorization-

The SQL DDL includes commands for specifying access rights to relations and views.

Data Definition Language-

The SQL DDL allows specification of not only a set of relations but also information about each relation, including-

- Schema for each relation
- The domain of values associated with each attribute.
- The integrity constraints.
- The set of indices to be maintained for each relation.
- The security and authorization information for each relation.
- The physical storage structure of each relation on disk.

Domain types in SQL-

The SQL standard supports a variety of built in domain types, including-

- Char (n)- A fixed length character length string with user specified length .
- Varchar (n)- A variable character length string with user specified maximum length n.
- Int- An integer.
- Small integer- A small integer.
- Numeric (p, d)-A Fixed point number with user defined precision.
- Real, double precision- Floating point and double precision floating point numbers with machine dependent precision.
- Float (n)- A floating point number, with precision of at least n digits.
- Date- A calendar date containing a (four digit) year, month and day of the month.
- Time- The time of day, in hours, minutes and seconds Eg. Time '09:30:00'.
- Number- Number is used to store numbers (fixed or floating point).
-

DDL statement for creating a table-

Syntax-

Create table tablename

(columnname datatype(size), columnname datatype(size));

Creating a table from a table-

Syntax-

```
 CREATE TABLE TABLENAME

[(columnname, columnname, ………)]

AS SELECT columnname, columnname……..FROM tablename;
```

Insertion of data into tables-

Syntax-

```
 INSERT INTO tablename

[(columnname, columnname, ………)]

Values(expression, expression);
```

Inserting data into a table from another table:

Syntax-

```
INSERT INTO tablename

SELECT columnname, columnname, …….

FROM tablename;
```

Insertion of selected data into a table from another table:

Syntax-

```
 INSERT INTO tablename

SELECT columnname, columnname……..

FROM tablename

WHERE columnname= expression;
```

Retrieving of data from the tables-

Syntax-

```
 SELECT * FROM tablename;
```

The retrieving of specific columns from a table-

Syntax-

```
 SELECT  columnname, columnname, ….

FROM tablename;
```

Elimination of duplicates from the select statement-

Syntax-

```
SELECT DISTINCT columnname, columnname

FROM tablename;
```

Selecting a data set from table data-

Syntax-

```
 SELECT columnname, columnname

FROM tablename

WHERE searchcondition;
```

DML (Data Manipulation Language) Data manipulation is

- The retrieval of information stored in the database.
- The insertion of new information into the database.
- The deletion of information from the database.

- The modification of information stored by the appropriate data model. There are basically two types.

(i) **Procedural DML**:- require a user to specify what data are needed and how to get those data.

(ii) **Non Procedural DML** : require a user to specify what data are needed without specifying how to get those data.

Updating the content of a table:

In creation situation we may wish to change a value in table without changing all values in the tuple . For this purpose the update statement can be used.

Update table name

Set columnname = experision, columnname =expression……

Where columnname = expression;

Deletion Operation:-

A delete reQuestionst is expressed in much the same way as Questionry. We can delete whole tuple (rows) we can delete values on only particulars attributes.

Deletion of all rows

Syntax:

Delete from tablename :

Deletion of specified number of rows

Syntax

Delete from table name

Where search condition;

Computation in expression lists used to select data

+ Addition	- Subtraction
* multiplication	** exponentiation
/ Division	() Enclosed operation

Renaming columns used with Expression Lists: - The default output column names can be renamed by the user if required

Syntax:

```
Select column name          result_columnname,
    Columnname              result_columnname,
From table name;
```

Experiment No.-03

Aim: Create a Table Student with a numbers of attributes and INSERT five records onto a table and perform ALTER and UPDATE operation on a table.

Create command syntax:

Create table table_name (column_name1 datatype size(),column_name2 datatype size(),column_name3 datatype size(),……………………..column_name ndatatype size());

Insert command syntax:

Insert into table_name values (column_1_value, column_2_value,…………….. column_n_value);

Alter command syntax:

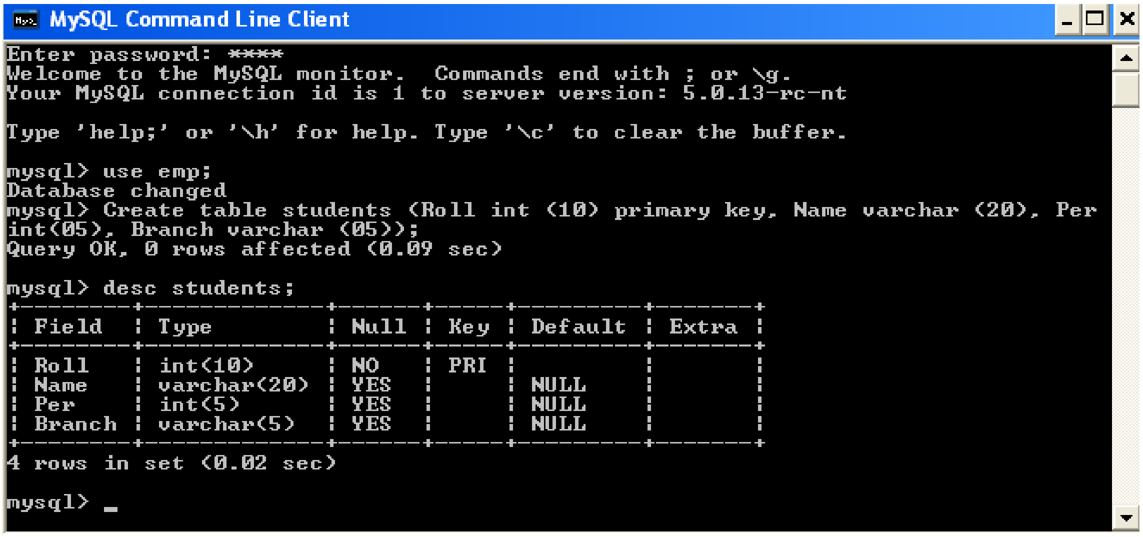
Alter table table_name Add/Modify/Drop (column_name1 datatype size(),column_name2 datatype size(),column_name3 datatype size(),……………………..column_name ndatatype size());

Update command syntax:

Update table Table_name set column_name1=Value1, column_name2=Value2,………….. column_namen=Valuen where condition;

Commands:-

Create table students (Roll int (10) primary key, Name varchar (20), Per int(05), Branch varchar (05));



```
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 5.0.13-rc-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use emp;
Database changed
mysql> Create table students (Roll int (10) primary key, Name varchar (20), Per
int(05), Branch varchar (05));
Query OK, 0 rows affected (0.09 sec)

mysql> desc students;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| Roll   | int(10)     | NO   | PRI |         |       |
| Name   | varchar(20) | YES  |     | NULL    |       |
| Per    | int(5)      | YES  |     | NULL    |       |
| Branch | varchar(5)  | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
4 rows in set (0.02 sec)

mysql> _
```

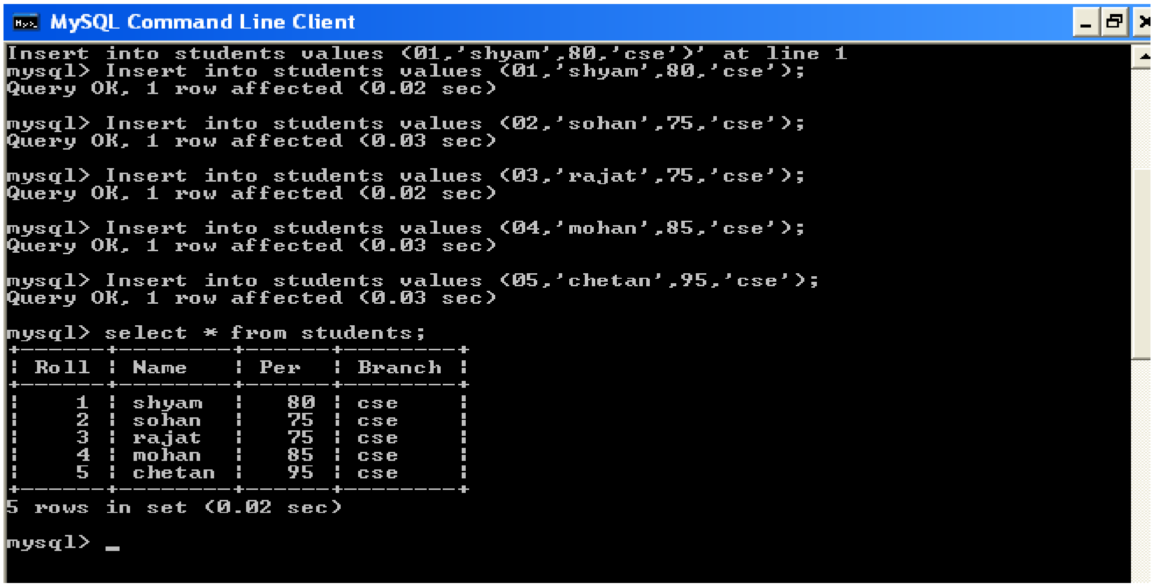
SQL> Insert into students values (01,’shyam’,80,’cse’);

SQL> Insert into students values (02,’sohan’,75,’cse’);

SQL> Insert into students values (03,’rajat’,75,’cse’);

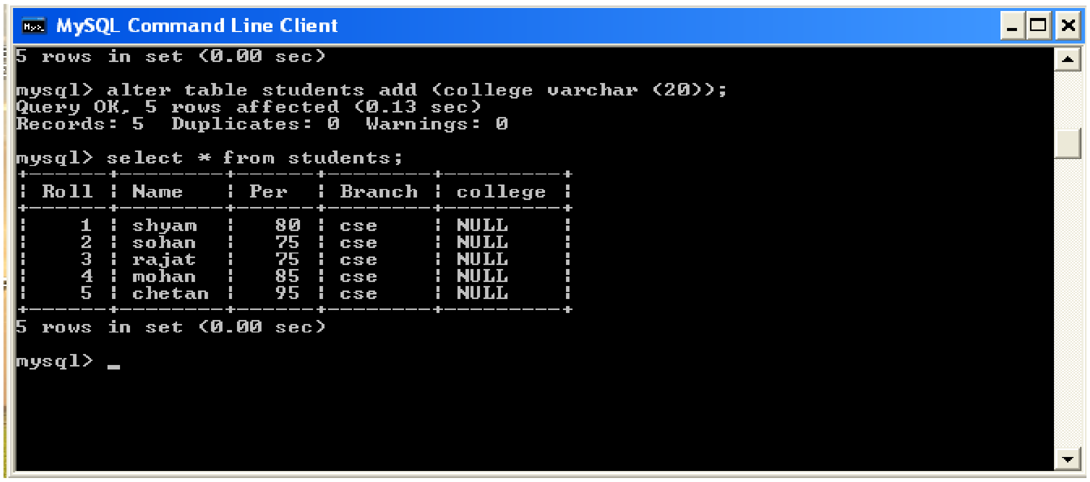
SQL> Insert into students values (04,’mohan’,85,’cse’);

SQL> Insert into students values (05,’chetan’,95,’cse’);



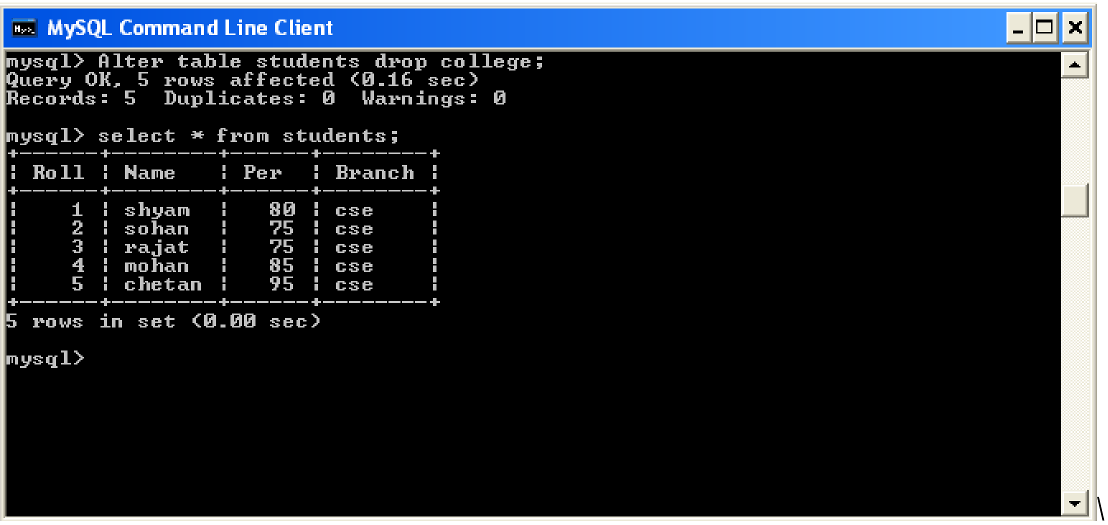
```
Insert into students values (01,'shyam',80,'cse')' at line 1
mysql> Insert into students values (01,'shyam',80,'cse');
Query OK, 1 row affected (0.02 sec)

mysql> Insert into students values (02,'sohan',75,'cse');
Query OK, 1 row affected (0.03 sec)

mysql> Insert into students values (03,'rajat',75,'cse');
Query OK, 1 row affected (0.02 sec)

mysql> Insert into students values (04,'mohan',85,'cse');
Query OK, 1 row affected (0.03 sec)

mysql> Insert into students values (05,'chetan',95,'cse');
Query OK, 1 row affected (0.03 sec)

mysql> select * from students;
+------+--------+------+--------+
| Roll | Name   | Per  | Branch |
+------+--------+------+--------+
|    1 | shyam  |   80 | cse    |
|    2 | sohan  |   75 | cse    |
|    3 | rajat  |   75 | cse    |
|    4 | mohan  |   85 | cse    |
|    5 | chetan |   95 | cse    |
+------+--------+------+--------+
5 rows in set (0.02 sec)

mysql> _
```

SQL>Alter table students add (college varchar (20));



```
5 rows in set (0.00 sec)

mysql> alter table students add (college varchar (20));
Query OK, 5 rows affected (0.13 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from students;
+------+--------+------+--------+---------+
| Roll | Name   | Per  | Branch | college |
+------+--------+------+--------+---------+
|    1 | shyam  |   80 | cse    | NULL    |
|    2 | sohan  |   75 | cse    | NULL    |
|    3 | rajat  |   75 | cse    | NULL    |
|    4 | mohan  |   85 | cse    | NULL    |
|    5 | chetan |   95 | cse    | NULL    |
+------+--------+------+--------+---------+
5 rows in set (0.00 sec)

mysql> _
```

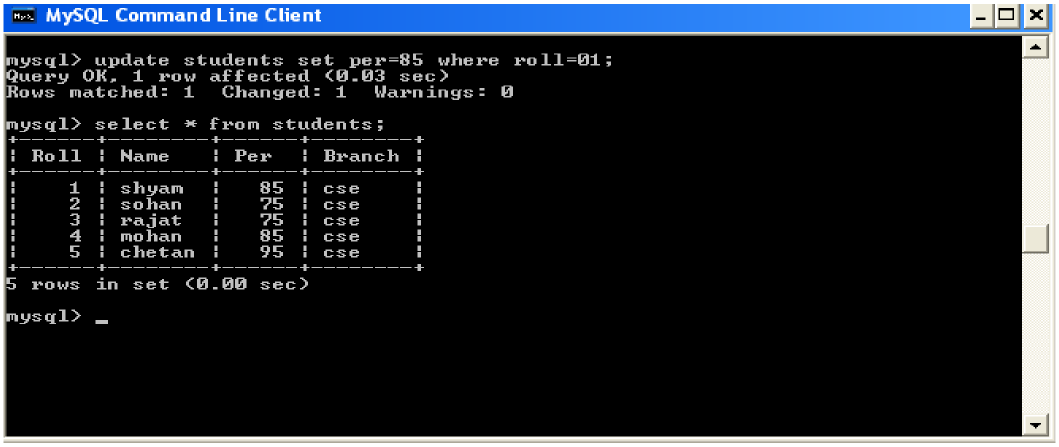
SQL>Alter table students drop college;



```
mysql> Alter table students drop college;
Query OK, 5 rows affected (0.16 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from students;
+------+--------+------+--------+
| Roll | Name   | Per  | Branch |
+------+--------+------+--------+
|    1 | shyam  |   80 | cse    |
|    2 | sohan  |   75 | cse    |
|    3 | rajat  |   75 | cse    |
|    4 | mohan  |   85 | cse    |
|    5 | chetan |   95 | cse    |
+------+--------+------+--------+
5 rows in set (0.00 sec)

mysql>
```

\

SQL>update students set per=85 where roll=1;



```
mysql> update students set per=85 where roll=01;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from students;
+------+--------+------+--------+
| Roll | Name   | Per  | Branch |
+------+--------+------+--------+
|    1 | shyam  |   85 | cse    |
|    2 | sohan  |   75 | cse    |
|    3 | rajat  |   75 | cse    |
|    4 | mohan  |   85 | cse    |
|    5 | chetan |   95 | cse    |
+------+--------+------+--------+
5 rows in set (0.00 sec)

mysql> _
```

Experiment No.-04

Aim: Study of SELECT command with different clauses and perform delete and truncate operation on existing table.

Step 1: Create table student with following attribute and its values.

Student

Roll	name	branch	sem	per	City
01	Ram	CS	5	70	Bhopal
02	Shyam	CS	5	80	Bhopal
03	Sohan	CS	5	85	Indore
04	Mohan	CS	5	95	Delhi
05	Ajay	CS	5	86	Jabalpur
06	Karan	IT	5	76	Delhi

Commands:

```
SQL> Select * from student;

SQL> Select * from student where per>80;

SQL> Select * from student where branch =’cs’ and city=’bhopal’;

SQL> Select * from student where name=’shyam’;

SQL> Select * from student where per<80 and city =’bhopal’;

SQL> Select max (per) from Student;
SQL> Select min (per) from Student;

SQL> delete * from student where per<80;

SQL> truncate Student;
```

Experiment No.-05

Aim: Perform various types of join operation on different Relation.

Step 1: Create two different table name student and student1 with following attribute and its values.

Student

Roll	name	branch	sem
01	Ram	CS	5
02	Shyam	CS	5
03	Sohan	CS	5
04	Mohan	CS	5
05	Ajay	CS	5
06	Karan	CS	5

Student1

Roll	address	per
01	Bhopal	70
02	Bhopal	80
03	Indore	75
05	Delhi	82

Join Operations:

INNER JOIN: Returns all rows when there is at least one match in BOTH tables

LEFT JOIN: Return all rows from the left table, and the matched rows from the right table

RIGHT JOIN: Return all rows from the right table, and the matched rows from the left table

FULL JOIN: Return all rows when there is a match in ONE of the tables.

Syntax:

```
SELECT column_name(s)
FROM table1
FULL JOIN table2
Using (column_name);
```

Commands:

```
SQL> Select * from student, student1;

SQL> Select * from student inner join student1;

SQL> Select * from student inner join student1 using (roll);

SQL> Select * from student right join student1 using (roll);

SQL> Select * from student left join student1 using (roll);

SQL>  Select * from student full join student1;

SQL>  Select * from student full join student1 using (roll);
```

Experiment No.-06

Aim: Study of Aggregate Function (avg, count, max, min, sum).

Definition: A function that performs a computation on a set of values rather than on a single value. For example, finding the average or mean of a list of numbers is an aggregate function. All database management and spreadsheet systems support a set of aggregate functions that can operate on a set of selected records or cells.

- **MAX Function -** The MySQL MAX aggregate function allows us to select the highest (maximum) value for a certain column.
- **MIN Function -** The MySQL MIN aggregate function allows us to select the lowest (minimum) value for a certain column.
- **COUNT Function -** The MySQL COUNT aggregate function is used to count the number of rows in a database table.
- **SUM Function -** The MySQL SUM aggregate function allows selecting the total for a numeric column.
- **AVG Function -** The MySQL AVG aggregate function selects the average value for certain table column.

Step 1: Create table student following attribute and its values.

Student

Roll	name	branch	per	Total_Fee
01	Ram	CS	80	65000
02	Shyam	CS	85	45000
03	Sohan	CS	65	25000
04	Mohan	CS	55	70000
05	Ajay	CS	75	65550
06	Karan	CS	86	65000

Commands:

```
SQL> Select max (per) from Student;

SQL> Select min (per) from Student;

SQL> Select sum (Total_fee) from Student;

SQL> Select avg (per) from Student;

SQL> Select count (*) from Student;
```

Experiment No.-07

Aim: Consider the employee database where primary key is underline. Give expression in SQL for each of the following:

Employee(employee_name,street,city)

Works(employee_name,Company_name,salary)

Company(company_name,manager_name)

Manager (employee_name,manager_name)

1. Find the name of all employees who works for “ABC”.
2. Find the name and the city of residence of all employee who works for “ABC”
3. Find the name, street and cities of residence of all employees who works for “ABC” and earn more than 10,000.
4. 1. Find the name of all employees who do not works for “ABC”.
5. Modify the database so that “ravi” now lives in “Bhopal”
6. Give all employee of “ABC” a 10% rise.
7. Delete all tuples in works relation for employee of “ABC”

Solutions:

```
SQL> Select employee_name from works where company_name=’’ABC;
```

```
SQL> Select employee_name,city from employee where employee_name in(select employee_name from works where company_name=‘ABC’);
```

```
SQL> Select * from employee where employee_name in(select employee_name from works where company_name=’ABC’ and salary>10,000);
```

```
SQL> Select employee name from works where company_name<>’ABC;’
```

```
SQL> update employee set city=’bhopal’ where employee_name=’ravi’;
```

```
SQL> update works set salary=salary*1.1 where company_name=’ABC’;
```

```
SQL> delete works where company_name=’ABC’;
```

Experiment No.-08

Aim: Perform Transaction operation on a table.

Solutions: **Account**

ID	Name	Balance
101	Ram	1000
102	Sohan	5000
103	Mohan	3000

```
SQL> Create table account (ID int (10) primary key, Name varchar (20), balance bigint (25));

SQL>Insert into account values (101,’ram’, 1000);

SQL>Insert into account values (102,’sohan’,5000);

SQL>Insert into account values (103,’mohan’, 3000);

SQL>Start transaction;

SQL> update account set balance=balance+100 where name=’ram’;

SQL> update account set balance=balance-500 where ID=102;

SQL> update account set balance=balance+800 where name=’Mohan’;

SQL>commit;      // Use to save changes.

SQL>Rollback;   //use before commit command otherwise does not revert the changes.
```

Experiment No.-09

Aim: To implement the concept of Cursor and Trigger.

Cursor– We have seen how oracle executes an SQL statement. Oracle DBA uses a work area for its internal processing. This work area is private to SQL's operation and is called a **cursor.**

The data that is stored in the cursor is called the **Active Data set.** The size of the cursor in memory is the size required to hold the number of rows in the Active Data Set.

Explicit Cursor- You can explicitly declare a cursor to process the rows individually. A cursor declared by the user is called **Explicit Cursor.** For Questionnaires that return more than one row, You must declare a cursor explicitly.

The data that is stored in the cursor is called the **Active Data set.** The size of the cursor in memory is the size required to hold the number of rows in the Active

Why use an Explicit Cursor- Cursor can be used when the user wants to process data one row at a time.

Explicit Cursor Management- The steps involved in declaring a cursor and manipulating data in the active data set are:-

- Declare a cursor that specifies the SQL select statement that you want to process.
- Open the Cursor.
- Fetch the data from the cursor one row at a time.
- Close the cursor.

Explicit Cursor Attributes- Oracle provides certain attributes/ cursor variables to control the execution of the cursor. Whenever any cursor (explicit or implicit) is opened and used Oracle creates a set of four system variables via which Oracle keeps track of the 'Current' status of the cursor. You

- Declare a cursor that specifies the SQL select statement that you want to process.
- Open the Cursor.
- Fetch the data from the cursor one row at a time.
- Close the cursor.

How to Declare the Cursor:-

The General Syntax to create any particular cursor is as follows:-

Cursor <Cursorname> is Sql Statement;

How to Open the Cursor:-

The General Syntax to Open any particular cursor is as follows:-

Open Cursorname;

Fetching a record From the Cursor:-

The fetch statement retrieves the rows from the active set to the variables one at a time. Each time a fetch is executed. The focus of the DBA cursor advances to the next row in the Active set.

One can make use of any loop structure (Loop-End Loop along with While, for) to fetch the records from the cursor into variable one row at a time.

The General Syntax to Fetch the records from the cursor is as follows:-

Fetch cursorname into variable1, variable2,______

Closing a Cursor:-

The General Syntax to Close the cursor is as follows:-

Close <cursorname>;

Database Triggers:-

Database triggers are procedures that are stored in the database and are implicitly executed(fired) when the contents of a table are changed.

Use of Database Triggers:-

Database triggers support Oracle to provide a highly customized database management system. Some of the uses to which the database triggers can be put to customize management information in Oracle are as follows:-

- A Trigger can permit DML statements against a table any if they are issued, during regular business hours or on predetermined weekdays.
- A trigger can also be used to keep an audit trail of a table along with the operation performed and the time on which the operation was performed.

- It can be used to prevent invalid transactions.
- Enforce complex security authorizations.

How to apply DataBase Triggers:-

A trigger has three basic parts:-

1. A triggering event or statement.
2. A trigger restriction
3. A trigger action.

Types of Triggers:-

Using the various options, four types of triggers can be created:-

1. **Before Statement Trigger:-** Before executing the triggering statement, the trigger action is executed.
2. **Before Row Trigger:-** Before modifying the each row affected by the triggering statement and before appropriate integrity constraints, the trigger is executed if the trigger restriction either evaluated to TRUE or was not included.’
3. **After Statement Trigger:-** After executing the triggering statement and applying any deferred integrity constraints, the trigger action is executed.
4. **After row Trigger:-** After modifying each row affected by the triggering statement and possibly applying appropriate integrity constraints, the trigger action is executed for the current row if the trigger restriction either evaluates to TRUE or was not included.

Syntex For Creating Trigger:-

The syntax for Creating the Trigger is as follows:-

Create or replace Trigger<Triggername> {before, after} {Delete, Insert, Update} On <Tablename> For Each row when Condition

Declare

<Variable declarations>;

<Constant Declarations>;

Begin

<PL/SQL> Subprogram Body;

Exception

Exception Pl/SQL block;

End;

How to Delete a Trigger:-

The syntex for Deleting the Trigger is as follows:-

Drop Trigger <Triggername>;

Experiment No.-10

Aim: To Study the Installation of Mysql.

Installing MySQL on Linux/UNIX

- **MySQL -** The MySQL database server, which manages databases and tables, controls user access, and processes SQL queries.
- **MySQL-client -** MySQL client programs, which make it possible to connect to and interact with the server.
- **MySQL-devel -** Libraries and header files that come in handy when compiling other programs that use MySQL.
- **MySQL-shared -** Shared libraries for the MySQL client.

MySQL-bench - Benchmark and performance testing tools for the MySQL database server.

The MySQL RPMs listed here are all built on a SuSE Linux system, but they'll usually work on other Linux variants with no difficulty.

Now, follow the following steps to proceed for installation:

- Login to the system using **root** user.

- Switch to the directory containing the RPMs:

- Install the MySQL database server by executing the following command. Remember to replace the filename in italics with the file name of your RPM.

```
[root@host]# rpm -i MySQL-5.0.9-0.i386.rpm
```

Above command takes care of installing MySQL server, creating a user of MySQL, creating necessary configuration and starting MySQL server automatically.

You can find all the MySQL related binaries in /usr/bin and /usr/sbin. All the tables and databases will be created in /var/lib/mysql directory.

- This is optional but recommended step to install the remaining RPMs in the same manner:

```
[root@host]# rpm -i MySQL-client-5.0.9-0.i386.rpm
[root@host]# rpm -i MySQL-devel-5.0.9-0.i386.rpm
[root@host]# rpm -i MySQL-shared-5.0.9-0.i386.rpm
[root@host]# rpm -i MySQL-bench-5.0.9-0.i386.rpm
```

Installing MySQL on Windows:

Default installation on any version of Windows is now much easier than it used to be, as MySQL now comes neatly packaged with an installer. Simply download the installer package, unzip it anywhere, and run setup.exe.

Default installer setup.exe will walk you through the trivial process and by default will install everything under C:\mysql.

Test the server by firing it up from the command prompt the first time. Go to the location of the mysqld server which is probably C:\mysql\bin, and type:

```
mysqld.exe --console
```

NOTE: If you are on NT, then you will have to use mysqld-nt.exe instead of mysqld.exe

If all went well, you will see some messages about startup and InnoDB. If not, you may have a permissions issue. Make sure that the directory that holds your data is accessible to whatever user (probably mysql) the database processes run under.

MySQL will not add itself to the start menu, and there is no particularly nice GUI way to stop the server either. Therefore, if you tend to start the server by double clicking the mysqld executable, you should remember to halt the process by hand by using mysqladmin, Task List, Task Manager, or other Windows-specific means.

Verifying MySQL Installation:

After MySQL has been successfully installed, the base tables have been initialized, and the server has been started, you can verify that all is working as it should via some simple tests.

Use the mysql admin Utility to Obtain Server Status:

Use **mysqladmin** binary to check server version. This binary would be available in /usr/bin on linux and in C:\mysql\bin on windows.

```
[root@host]# mysqladmin --version
```

It will produce the following result on Linux. It may vary depending on your installation:

```
mysqladmin  Ver 8.23 Distrib 5.0.9-0, for redhat-linux-gnu on i386
```

If you do not get such message, then there may be some problem in your installation and you would need some help to fix it.

Execute simple SQL commands using MySQL Client:

You can connect to your MySQL server by using MySQL client using **mysql** command. At this moment, you do not need to give any password as by default it will be set to blank.

So just use following command

```
[root@host]# mysql
```

It should be rewarded with a mysql> prompt. Now, you are connected to the MySQL server and you can execute all the SQL command at mysql> prompt as follows:

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| mysql    |
| test     |
+----------+
2 rows in set (0.13 sec)
```

Post-installation Steps:

MySQL ships with a blank password for the root MySQL user. As soon as you have successfully installed the database and client, you need to set a root password as follows:

```
[root@host]# mysqladmin -u root password "new_password";
```

Now to make a connection to your MySQL server, you would have to use the following command:

```
[root@host]# mysql -u root -p
Enter password:*******
```

UNIX users will also want to put your MySQL directory in your PATH, so you won't have to keep typing out the full path every time you want to use the command-line client. For bash, it would be something like:

```
export PATH=$PATH:/usr/bin:/usr/sbin
```

Running MySQL at boot time:

If you want to run MySQL server at boot time, then make sure you have following entry in /etc/rc.local file.

```
/etc/init.d/mysqld start
```

Also,you should have mysqld binary in /etc/init.d/ directory.