DBMS LAB

LIST OF EXPERIMENTS

1. Create a database table, add constraints (primary key, unique, check, Not null), insert rows, update and delete rows using SQL DDL and DML commands.

2. Create a set of tables, add foreign key constraints and incorporate referential integrity.

3. Query the database tables using different 'where' clause conditions and also implement aggregate functions.

4. Query the database tables and explore sub queries and simple join operations.

5. Query the database tables and explore natural, equi and outer joins.

6. Write user defined functions and stored procedures in SQL.

7. Execute complex transactions and realize DCL and TCL commands.

8. Write SQL Triggers for insert, delete, and update operations in a database table.

9. Create View and index for database tables with a large number of records.

10. Create an XML database and validate it using XML schema.

11. Create Document, column and graph based data using NOSQL database tools.

12. Develop a simple GUI based database application and incorporate all the above mentioned features 13. Case Study using any of the real life database applications from the following list

a) Inventory Management for a EMart Grocery Shop

b) Society Financial Management

c) Cop Friendly App – Eseva

d) Property Management – eMall

e) Star Small and Medium Banking and Finance

CONTENTS

EX.No :1

Data Definition Commands, Data Manipulation Commands and Constraints

AIM:

To create a database and how to perform the operations and add constraints.

ALGORITHM:

Step 1: Create a table using sql query

Step 2: Enter the input in the created table

Step 3: To retrieve a data from table use select command

Step 4: Add constraints primary key, unique, check, Notnull

Step 5: The output will be displayed based on the query.

PROGRAM:

DDL (DATA DEFINITION LANGUAGE) COMMAND:

1. CREATE
2. ALTER
3. DROP
4. TRUNCATE
5. COMMENT
6. RENAME

1. Creation of database and table:

```
mysql>create database db;
        Query OK, 0 rows affected (0.14
sec)mysql>use db;

        Database changed
```

2. Create the table:

```
SQL> CREATE TABLE STUDENT (SNO INT, NAME CHAR (11), POSITION
VARCHAR(8));

 Table created.
```

3. View the table

```
structure:SQL>

DESC EMP; Name

Null?           Type

-----------------------   -------------------------

EMPNO           INT

ENAME           CHAR(11)

POSITION        VARCHAR

(8)
```

4. Alter the Table:

```
SQL>ALTER TABLE EMP MODIFY EMPNO INT;

Table altered.

SQL> DESC
EMP;

Name Null?      Type
-----------------------   -------------------------
EMPNO           INT

ENAME           CHAR(11)

POSITION         VARCHAR(8)

                SQL> ALTER TABLE STUDENT ADD( SALARY INT);

Table altered.
SQL> DESC
EMP;

Name Null?      Type

----------------------- -------------------------

EMPNO           INT

ENAME           CHAR(11)
```

```
POSITION     VARCHAR(8)
SALARY            INT
SQL>ALTER TABLE EMP ADD (DOB DATE, DOJ DATE);
Table altered.
```

5. Remove/Drop the Column:

```
SQL> ALTER TABLE EMP DROP COLUMN QUALIFICATION;
Table altered.
SQL> ALTER TABLE EMP DROP (DOB, QUALIFICATION);
Table altered
```

6. Rename the Table:

```
SQL> RENAME TABLE EMP TO EMPLOYEE;
```

7. Truncate the Table:

```
SQL> Truncate table EMPLOYEE;
```

8. Drop the table:

```
SQL> DROP TABLE EMPLOYEE;
```

DML (DATA MANIPULATION LANGUAGE)

1. SELECT
2. INSERT
3. DELETE
4. UPDATE

1. Insert a record from an existing table:

```
SQL>INSERT INTO EMPLOYEE VALUES (101,'NAGARAJAN','LECTURER',15000);
1 row created.
SQL> INSERT INTO EMPLOYEE (EMPNO,ENAME,POSITION,SALARY) VALUES
(102,"KANNAN","PROFESSOR",50000);
```

3. Display the records from employee.

```
SQL>SELECT * FROM EMPLOYEE;

EMPNO        ENAME        DESIGNATI   SALARY
                          N
----------   ----------   ----------  ----------

101          NAGARAJA     LECTURER    15000
                N
102          KANNAN       PROFESSOR   50000
```

4. Update the Table:

```
SQL> UPDATE EMP SET SALARY=16000 WHERE EMPNO=101;

1 row updated.
```

5. Delete the row from Table:

```
SQL> DELETE EMP WHERE EMPNO=103;

1 row deleted.
```

6. Not NULL Constraint, Unique Constraint :

```
SQL> CREATE TABLE CUSTOMERS( ID   INT  NOT NULL, NAME VARCHAR (20)
NONULL, AGEINT  NOT NULL UNIQUE, ADDRESS  CHAR (25) , SALARY   DECIMAL (18, 2),

PRIMARY KEY (ID) );

  CUSTOMERS
  ID
  NAME
  AGE
  ADDRESS
  Empty


  SQL>INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
  VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00 );

  SQL>INSERT INTO CUSTOMERS
  (ID,NAME,AGE,ADDRESS,SALARY)VALUES (2, 'Khilan', 25,
  'Delhi', 1500.00 );
```

```
SQL>INSERT INTO CUSTOMERS
(ID,NAME,AGE,ADDRESS,SALARY)VALUES (3, 'kaushik', 23,
'Kota', 2000.00 );

SQL>INSERT INTO CUSTOMERS
(ID,NAME,AGE,ADDRESS,SALARY)VALUES (4, 'Chaitali', 25,
'Mumbai', 6500.00 );
```

CUSTOMERS

ID	NAME	AGE	ADDRESS
1	Ramesh	32	Ahmedabad
2	Khilan	25	Delhi
3	kaushik	23	Kota
4	Chaitali	25	Mumbai

7. Primary Key:

```
SQL>CREATE TABLE CUSTOMERS( ID   INT  NOT NULL, NAME VARCHAR (20) NOT NULL,
 AGE INT  NOT NULL UNIQUE, ADDRESS  CHAR (25) , SALARY   DECIMAL (18, 2),PRIMARY
 KEY (ID) );
SQL>ALTER TABLE CUSTOMERS DROP PRIMARY KEY ;

 ID | NAME    | AGE | ADDRESS  | SALARY |
 +....+..........+.....+............+..........+
 | 1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
 |  2 | Khilan   |  25 | Delhi     |  1500.00 |
 |  3 | kaushik  |  23 | Kota      |  2000.00 |
 |  4 | Chaitali |  25 | Mumbai    |  6500.00 |
 |  5 | Hardik   |  27 | Bhopal    |  8500.00 |
 |  6 | Komal    |  22 | MP        |  4500.00 |
 |  7 | Muffy    |  24 | Indore    | 10000.00 |
 +....+..........+.....+............+..........+
```

RESULT:

The database was created using the various commands and constraints.

EX. NO :2

Database Querying – Foreign key constraints and referentialintegrity.

AIM:

To create a database and to add foreign key constraints and referential

integrity.

ALGORITHM:

Creating a table for relating databases using constraints

Step 1: Create a table using sql query

Step 2: Enter the input in the created table

Step 3: To get related data from different tables use constraint

Step 4: The output will be displayed based on the constraint

PROGRAM:

Foreign Key constraint:

Foreign Key:

```
SQL> CREATE TABLE ORDERS (   ID    INT        NOT NULL,   DATE      DATETIME,CUSTOMER_ID

INT references CUSTOMERS(ID),   AMOUNT     double,  PRIMARY KEY (ID));


SQL>ALTER TABLE ORDERS  DROP FOREIGN KEY;

 SQL> ALTER TABLE Cancellation ADD FOREIGN KEY (PNR_NO) REFERENCES
        Reservation(PNR_N);
     Table altered.
```

Foreign Key Constraint with alter command

```
     SQL> alter table reservation add constraint fk_icode foreign key (busno) references
           bus(bus_no);
         Table altered.
```

Referential integrity:

References constraint defined at column level

```
SQL> create table Passenger(PNR_NO Numeric(9) references r eservation , Ticket_NO
         Numeric(9) references ticket, Name varchar(20), Age Number(4), Sex char(10),
         PPNOvarchar(15));
```

Table created.

RESULT:

The database was created for relating between databases using the foreign key and referential constraints.

EX.No:3

Aggregate functions

AIM:

To create a database tables using aggregate functions.

ALGORITHM:

Step 1: Create a table using sql query
Step 2: Enter the input in the created table
Step 3: To retrieve a data from table use select command
Step 4: The output will be displayed based on the query
Step 5: Query using aggregate functions.

PROGRAM:

SQL> desc emp;

Name	Null?	Type
EMPNO	NOT NULL	NUMBER (4)
ENAME	NOT NULL	VARCHAR2(30)
JOB		VARCHAR2 (10)
MGR		NUMBER (4)
HIREDATESAL		DATE NUMBER (7,2)
DEPTNO		NUMBER (2)

SQL> select * from emp;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	DEPTNO
3737	Priya	Officer	7777	07-MAR-09	51000	7
4545	Priya	Secretary	3337	09-JUN-09	21000	20
7575	Karthi	Manager	3337	07-JUL-07	75000	20
5352	Retish	Secretary	5555	09-JUN-09	20000	8
5332	Rocky	Assist	5555	08-JAN-08	19500	7
6544	Santh	Officer	3337	23-JAN-09	19000	8

6 rows selected.

```
SQL> desc emp;
 Name                          Null?  Type
 ------------------------------ ------- ---------------
 EMPNO                                 NUMBER(4)
 PH_NO                                 NUMBER(8)

SQL> select * from

emp;EMPNO  PH_NO
-----------------------------
    3737       225301
    4545       485565
    6544       789663
    7575       896652
    6555       987777
```

1) Avg (Average):

```
SQL> select avg(sal) from emp;

SAL=34250
```

2) Min (Minimum):

```
SQL>Select min(salary) from emp;
MIN=19000
```

3) Max (Maximum):

```
SQL>select max(salary) from emp;
MAX=75
```

Sum:

```
SQL>Select sum(SAL) from emp;

SUM=205500
```

4)Count:

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

COUNT-6
```

5) Where clause:

```
SQL>SELECT * FROM Customers WHERE Country='Mexico';
```

CustomerID CustomerName	ContactName Address	City PostalCode	Country
1 Alfreds Futterkiste	Maria Anders Obere Str. 57	Berlin 12209	Germany

RESULT:

The employee database was created and aggregate functions were used.

EX.No:4

Database Querying – Simple queries, Nested queries, Sub queriesand Joins

AIM:

To create a database using Nested Queries, Sub Queries and different joins

ALGORITHM:

Step 1: Create a table using sql query

Step 2: Enter the input in the created table

Step 3: To get related data from different tables use joins Query

Step4: The output will be displayed based on the joins query

PROGRAM:

```
SQL> CREATE TABLE CUSTOMERS (ID  INT  NOT NULL, NAME VARCHAR (20) NOT NULL,AGE
       INT NOTNULL, ADDRESS CHAR (25), SALARY  DECIMAL (18, 2),
       PRIMARY KEY (ID));


SQL> INSERT INTO CUSTOMERS (ID, NAME,AGE,ADDRESS,SALARY)
        VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00 );

        INSERT INTO CUSTOMERS (ID,NAME, AGE, ADDRESS,SALARY)VALUES (2,
        'Khilan', 25, 'Delhi', 1500.00 );

        INSERT INTO CUSTOMERS
        (ID,NAME,AGE,ADDRESS,SALARY)VALUES (3, 'kaushik',
        23, 'Kota', 2000.00 );

        INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)VALUES (4,
        'Chaitali', 25, 'Mumbai', 6500.00 );

        INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)VALUES (5,
        'Hardik', 27, 'Bhopal', 8500.00 );

        INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)VALUES
        (6,'Komal', 22, 'MP', 4500.00 );
```

```
ID |      NAME     |        AGE |   ADDRESS   |     SALARY  |
+.....+................+         ......+    ...............          -+............+
 | 1 |     Ramesh  |        32 |     Ahmedabad | 2000.00 |
 | 2 |     Khilan  |         25 |     Delhi     |         1500.00 |
 | 3 |     kaushik  |        23 |     Kota      |         2000.00 |
 | 4 |     Chaitali |         25 |     Mumbai    |       6500.00 |
 | 5 |     Hardik  |         27 |     Bhopal    |       8500.00 |
 | 6 |     Komal    |        22 |      MP       |         4500.00 |
```

1. Sub Queries:

```
SQL> SELECT * FROM CUSTOMERS WHERE ID IN (SELECT ID  FROM
         CUSTOMERSWHERESALARY > 4500);
```

```
ID |       NAME     |       AGE |   ADDRESS |      SALARY  |
+.....+.............+......+............+.............+
|  4 |     Chaitali |         25 |     Mumbai  |      6500.00 |
|  5 |     Hardik  |           27 |    Bhopal |         8500.00 |
```

2. Sub Queries with Update Command:

```
SQL> UPDATE CUSTOMERS SET SALARY = SALARY * 0.25 WHERE AGE IN (SELECTAGE FROM
          CUSTOMERS_BKP WHERE AGE >= 27);
```

```
+....+.............+......+..............+.............+
| ID | NAME     | AGE | ADDRESS   | SALARY  |
+.....+.............+......+..............+.............+
|  1 | Ramesh   |  35 |     Ahmedabad |  125.00 |
|  2 | Khilan     |  25 |     Delhi          |  1500.00 |
|  3 | kaushik   |  23 |     Kota           |  2000.00 |
|  4 | Chaitali   |  25 |      Mumbai      | 6500.00 |
|  5 | Hardik     |  27 |     Bhopal        |  2125.00 |
|  6 | Komal      |  22 |     MP              |  4500.00 |
```

2. Sub Queries with Delete Command:

```
SQL> DELETE FROM CUSTOMERS WHERE AGE IN (SELECT AGE FROM
          CUSTOMERS_BKPWHERE AGE >= 27 );
```

```
ID | NAME     | AGE | ADDRESS | SALARY  |
+.....+.............+......+............+.............+
|  2 | Khilan    |  25 |   Delhi        |  1500.00 |
|  3 | kaushik  |  23 |   Kota          |  2000.00 |
|  4 | Chaitali  |  25 |   Mumbai    |  6500.00 |
|  6 | Komal    |  22 |    MP            |  4500.00 |
```

3. Joins:

Table 1 – CUSTOMERS Table

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 |        Ahmedabad  |  2000.00 |
|  2 | Khilan   |  25 |        Delhi      |  1500.00 |
|  3 | kaushik  |  23 |        Kota       |  2000.00 |
|  4 | Chaitali |  25 |        Mumbai     | 6500.00 |
|  5 | Hardik   |  27 |        Bhopal     |  8500.00 |
|  6 | Komal    |  22 |        MP         |  4500.00 |
|  7 | Muffy    |  24 |        Indore     | 10000.00 |
+----+----------+-----+-----------+----------+
```

Table 2 – ORDERS Table

```
+-----+---------------------+-------------+--------+
|OID  | DATE                |       CUSTOMER_ID | AMOUNT |
+-----+---------------------+-------------+--------+
| 102 | 2009-10-08 00:00:00 |           3 |   3000 |
| 100 | 2009-10-08 00:00:00 |           3 |   1500 |
| 101 | 2009-11-20 00:00:00 |           2 |   1560 |
| 103 | 2008-05-20 00:00:00 |           4 |   2060 |
```

```
SQL> SELECT ID, NAME, AGE, AMOUNT FROM CUSTOMERS, ORDERSWHERE
        CUSTOMERS.ID = ORDERS.CUSTOMER_ID;

        ID | NAME     | AGE | AMOUNT |
        +----+----------+-----+--------+
        |  3 | kaushik  |  23 |   3000 |
        |  3 | kaushik  |  23 |   1500 |
        |  2 | Khilan  |  25 |  1560 |
        |  4 | Chaitali |  25 |  2060
```

INNER JOIN

```
        SQL> SELECT ID, NAME, AMOUNT, DATE FROM CUSTOMERS
                INNER JOIN ORDERS ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;

        ID | NAME     | AMOUNT | DATE                |
        +----+----------+--------+---------------------+
        |  3 | kaushik  |  3000 | 2009-10-08 00:00:00 |
        |  3 | kaushik  |  1500 | 2009-10-08 00:00:00 |
        |  2 | Khilan  |  1560 | 2009-11-20 00:00:00 |
        |  4 | Chaitali |  2060 | 2008-05-20 00:00:00
```

LEFT JOIN

```
SQL> SELECT ID, NAME, AMOUNT, DATE FROM CUSTOMERS
        LEFT JOIN ORDERS ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;

ID | NAME     | AMOUNT | DATE                |
+----+------------+-----------+-------------------------+
|  1 | Ramesh   |   NULL | NULL                |
|  2 | Khilan  |  1560 | 2009-11-20 00:00:00 |
|  3 | kaushik |   3000 | 2009-10-08 00:00:00 |
|  3 | kaushik |   1500 | 2009-10-08 00:00:00 |
|  4 | Chaitali |  2060 | 2008-05-20 00:00:00 |
|  5 | Hardik   |   NULL | NULL               |
|  6 | Komal    |   NULL | NULL               |
|  7 | Muffy  |  NULL | NULL
```

RIGHT JOIN:

```
SQL> SELECT ID, NAME, AMOUNT, DATE FROM CUSTOMERS
        RIGHT JOIN ORDER ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;

ID   | NAME     | AMOUNT | DATE               |
+-------+------------+-----------+-------------------------+
|   3 | kaushik  |   3000 | 2009-10-08 00:00:00 |
|   3 | kaushik  |   1500 | 2009-10-08 00:00:00 |
|  2 | Khilan  |   1560 | 2009-11-20 00:00:00 |
| 4 | Chaitali | 2060 | 2008-05-20 00:00:00
```

FULL JOINS:

```
SQL> SELECT ID, NAME, AMOUNT, DATE FROM CUSTOMERS
          FULL JOIN ORDERS ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;

ID   | NAME     | AMOUNT | DATE               |
+-------+------------+-----------+-------------------------+
|   1 | Ramesh   |   NULL | NULL               |
|  2 | Khilan  |   1560 | 2009-11-20 00:00:00 |
|   3 | kaushik  |   3000 | 2009-10-08 00:00:00 |
|   3 | kaushik  |   1500 | 2009-10-08 00:00:00 |
|  4 | Chaitali |  2060 | 2008-05-20 00:00:00 |
|   5 | Hardik   |   NULL | NULL                |
|   6 | Komal     |   NULL | NULL               |
|   7 | Muffy     |   NULL | NULL               |
|   3 | kaushik  |   3000 | 2009-10-08 00:00:00 |
|   3 | kaushik  |   1500 | 2009-10-08 00:00:00 |
|  2 | Khilan  |   1560 | 2009-11-20 00:00:00 |
|  4 | Chaitali |  2060 | 2008-05-20 00:00:0
```

SELF JOINS:

```
SQL> SELECT a.ID, b.NAME, a.SALARY FROM CUSTOMERS a, CUSTOMERS bWHERE
        a.SALARY <b.SALARY;
ID | NAME     | SALARY  |
+.....+.............+............+
|  2 | Ramesh  | 1500.00 |
|  2 | kaushik  | 1500.00 |
|  1 | Chaitali | 2000.00 |
|  2 | Chaitali | 1500.00 |
|  3 | Chaitali | 2000.00 |
|  6 | Chaitali | 4500.00 |
|  1 | Hardik   | 2000.00 |
|  2 | Hardik   | 1500.00 |
|  3 | Hardik   | 2000.00 |
|  4 | Hardik   | 6500.00 |
|  6 | Hardik   | 4500.00 |
|  1 | Komal    | 2000.00 |
|  2 | Komal    | 1500.00 |
|  3 | Komal    | 2000.00 |
|  1 | Muffy   | 2000.00 |
|  2 | Muffy   | 1500.00 |
|  3 | Muffy   | 2000.00 |
|  4 | Muffy   | 6500.00 |
|  5 | Muffy   | 8500.00 |
|  6 | Muffy   | 4500.00 |
+.....
```

RESULT:

The database was created for relating between databases using the joins and sub queries.

EX.No:5

DATABASE PROGRAMMING- VIEWS -NATURAL, EQUI AND OUTERJOINS

AIM:

To create a database tables using view command and joins.

ALGORITHM:

Step 1: Create a table using sql query
Step 2: Enter the input in the created table
Step 3: To retrieve a data from table use views and joins.
Step 4: The output will be displayed based on the query

PROGRAM:

NATURAL JOIN:

```
SQL>CREATE TABLE customer ( id INT AUTO_INCREMENT PRIMARY KEY,  customer_name
VARCHAR(55), account int, email VARCHAR(55));
Table Created

SQL>CREATE TABLE balance ( id INT AUTO_INCREMENT PRIMARY KEY,  account int,
balance FLOAT(10,2));

Table Created

SQL>INSERT INTO customer(customer_name, account, email)VALUES('Stephen', 1030,
'stephen@javat point.com'), ('Jenifer', 2035, 'jenifer@javatpoint.com'), ('Mathew', 5564,
'mathew@javatpoint.com'), ( 'Smith', 4534, 'smith@javatpoint.com'), ('David', 7648,
'david@javatpoint.com');
SQL>INSERT INTO balance(account, balance) VALUES(1030, 50000.00),
(2035, 230000.00), (5564, 1250
00.00), (4534, 80000.00),  (7648, 45000.00);
```

Customer

id	customer_name	account	email
1	Stephen	1030	stephen@javatpoint.com
2	Jenifer	2035	jenifer@javatpoint.com
3	Mathew	5564	mathew@javatpoint.com

4	Smith	4534	smith@javatpoint.com
5	David	7648	david@javatpoint.com

Balance

id	accoun t	balance
1	1030	50000
2	2035	230000
3	5564	125000
4	4534	80000
5	7648	45000

EQUI JOIN:

SQL>Select * from Student;

id	name	class	city
3	Hina3	Delhi	
4	Megha	2	Delhi
6	Gouri	2	Delhi

SQL>Select * from Record;

id	class	city
9 3	Delhi	
10	2	Delhi
12	2	Delhi

SQL>SELECT student.name, student.id, record.class, record.city FROM studentJOIN record ON student.city = record.city;

name	id	class	city
Hina	3	3	Delhi
Megha	4	3	Delhi
Gouri	6	3	Delhi
Hina	3	2	Delhi
Megha	4	2	Delhi
Gouri	6	2	Delhi
Hina	3	2	Delhi
Megha	4	2	Delhi
Gouri	6	2	Delhi

NON EQUI JOIN :

```
SQL>SELECT student.name, record.id, record.city FROM student, record WHERE
Student.id <Record.id ;
```

name	id	city
Hina	9	Delhi
Megha	9	Delhi
Gouri	9	Delhi
Hina	10	Delhi
Megha	10	Delhi
Gouri	10	Delhi
Hina	12	Delhi
Megha	12	Delhi
Gouri	12	Delhi

FULL OUTER JOIN:

```
SQL>CREATE TABLE students ( id INTEGER, name TEXT NOT NULL, gender
TEXT TNULL);
```

To insert some values

```
SQL>INSERT INTO students VALUES (1, 'Ryan', 'M');
SQL>INSERT INTO students VALUES (2, 'Joanna', 'F'); INSERT INTO students Values (3, 'Moana', 'F');

SQL>CREATE TABLE college ( id INTEGER, classTeacher TEXT NOT NULL, StrengthTEXT
NOT NULL);insert some values INSERT INTO college VALUES (1, 'Alpha', '50');INSERT
INTO college VALUES (2, 'Romeo', '60');INSERT INTO college Values (3, 'Charlie', '55');

SQL>SELECT College.classTeacher, students.id FROM College FULL OUTER JOINCollege ON
College.id=students.id ORDER BY College.classTeacher;
```

```
Class Teacher
             Id
Alpha        1
Romeo        2
Charlie      3
```

RESULT:

The database was created for relating between databases using the joins and sub queries.

EX.No:6

DATABASE PROGRAMMING- PROCEDURES AND FUNCTIONS

AIM:

To write a PL/SQL query to create a procedure and functions

Algorithm:

Step 1: Start a program by creating table
Step2: Enter the input
Step 3:The output will be displayed based on the query
Step: Stop the program

Function:

```
SQL>delimiter $$
>Create function func1 (str varchar(20))
>return char(50)
>return CONCAT('WELCOME TO', str)
>end $$

>delimiter $$
>select func1('DBMS LAB');
>$$
```

Procedure:

```
SQL> desc student;
```

Name	Null?	Type
REGNO		NUMBER(4)
NAME		VARCHAR2(20)
MARK1		NUMBER(3)
MARK2		NUMBER(3)
MARK3		NUMBER(3)
MARK4		NUMBER(3)
MARK5		NUMBER(3)

```
SQL> select * from student;
```

REGNO NAME	MARK1	MARK 2	MARK3	MARK 4	MARK 5
101 priya	78	88	77	60	89
102 karthi	99	77	69	81	99
103 karthipriya	100	90	97	89	91

```
SQL> set serverout on;

SQL> declare
  2  ave number(5,2);
  3  tot number(3);
  4  cursor c_mark is select*from student where mark1>=40 and
  mark2>=40 and5 mark3>=40 and mark4>=40 and mark5>=40;
  6  begin
  7  dbms_output.put_line('regno name mark1 mark2 mark3 mark4 mark5 total
  8       average');
  9  dbms_output.put_line('
 ......');10 for student in c_mark
 11  loop
 12  tot:=student.mark1+student.mark2+student.mark3+student.mark4+student.mark5;
 13  ave:=tot/5;
 14  dbms_output.put_line(student.regno||rpad(student.name,15)
 15  ||rpad(student.mark1,6)||rpad(student.mark2,6)||rpad(student.mark3,6)
 16  ||rpad(student.mark4,6)||rpad(student.mark5,6)||rpad(tot,8)||rpad(ave,5));
 17  end loop;
 18  end;
 19 /
```

regno	name	mark1	mark2	mark3	mark4	mark5	total	average
101	priya	78	88	77	60	89	392	78.4
102	karthi	99	77	69	81	99	425	85
103	karthipriya	100	90	97	89	91	467	93.4

PL/SQL procedure successfully completed.

RESULT:
Thus the program for creation of procedure is executed successfull

EX.No:7

Execute complex transactions and realize DCL and TCL commands

AIM:

To execute DCL and TCL Commands.

Algorithm:

```
Step 1: Start a program
Step 2: Enter the input
Step 3: Execute the TCL abd DCL commands.
Step 4: Stop the program
```

TCL Commands: Commit,
Rollback, Savepoint:

1. Write a query to end your current transaction and make permanent all changesperformed in the transaction.

```
SQL> commit;
Commit
complete.
```

2. Write a query to create a table goods with sno,itemcode,itemname,costnumber as itsattributes and assign primary key constraint for the column “itemcode”.

```
SQL> create table goods(sno number,itemcode number primary key,itemname varchar2(10),cost
number);

Table created.
SQL> insert into goods values(&sno,&itemcode,'&itemname',&cost);
Enter value for sno:1
Enter value for itemcode: 1025
Enter value for itemname: dell moniters
Enter value for cost: 5000
old 1: insert into goods values(&sno,&itemcode,'&itemname',&cost)new 1: insert into goods
values(1,1025,'dell moniters',5000)

SQL> insert into goods

values(&sno,&itemcode,'&itemname',&cost);
Enter value for sno: 1
Enter value for itemcode: 1025
Enter value for itemname: moniter
Enter value for cost: 5000
old 1: insert into goods values(&sno,&itemcode,'&itemname',&cost)
new 1: insert into goods values(1,1025,'moniter',5000)

1 row created.

 SQL> /
Enter value for sno: 2
Enter value for itemcode: 1026
Enter value for itemname: mouse
Enter value for cost: 250
old 1: insert into goods values(&sno,&itemcode,'&itemname',&cost)
new 1: insert into goods values(1026,1026,'mouse',250)

1 row created.

 SQL> /
Enter value for sno: 3
Enter value for itemcode: 1027
Enter value for item name: RAM
Enter value for cost: 1500
old 1: insert into goods values(&sno,&itemcode,'&itemname',&cost)
new 1: insert into goods values(3,1027,'RAM',1500)

1 row created.

 SQL> /
Enter value for sno: 4
Enter value for itemcode: 1028
Enter value for itemname: webcam
Enter value for cost: 350
old 1: insert into goods values(&sno,&itemcode,'&itemname',&cost)
new 1: insert into goods values(4,1028,'webcam',350)

 1 row created.

 SQL> /
Enter value for sno: 5
Enter value for itemcode: 1029
Enter value for itemname: pendrive
Enter value for cost: 500
old 1: insert into goods values(&sno,&itemcode,'&itemname',&cost)
new 1: insert into goods values(5,1029,'pendrive',500)

 1 row created.
```

```
SQL> select *from goods;
SNO      ITEMCODE      ITEMNAME     COST
---------------- ------------------------------- ------------------------------ ----------------
1        1025              moniter        5000
2        1026             mouse           250
3        1027             RAM             1500
4        1028             webcam          350
5        1029             pendrive         500

SQL> commit;
Commit complete.
```

3. Write a query to add the record into the table “goods” and set the Savepoint S1, S2 and S3 and verify it.

```
SQL> insert into goods values (6, 1030,'keyboard', 500);
1 row created.
SQL> save point s1; Save point created.
SQL> insert into goods values (7, 1031,'DVD drive', 2500);1 row created.
SQL> save point s2;Save point created.
SQL> insert into goods values (8, 1032,'UPS', 3000); 1 row created.
SQL> insert into goods values (9, 1033,'CPU', 5000); 1 row created.
SQL> save point s3; Save point created.
```

4. Write a query to Roll back to Save point S3 and verify it.

```
SQL> rollback to savepoint s3;Rollback complete.

To Verify: SQL> select *from goods;SNO ITEMCODE ITEMNAME COST
1        1025   moniter        5000
2        1026   mouse          250
3        1027   RAM            1500
4        1028   webcam         350
5        1029   pendrive       500
6        1030   keyboard       500
7        1031   DVD drive      2500
8        1032   UPS            3000
9        1033   CPU            5000

9 rows selected.
```

Write a query to Rollback to Save point S2 and verify it.

```
SQL> rollback to save point s2;
Rollback complete.

To Verify: SQL> select *from goods;

SNO    ITEMCODE     ITEMNAME   COST
------ ------------ ---------- ------
1      1025         moniter    5000
2      1026         mouse      250
3      1027         RAM        1500
4      1028         webcam     350
5      1029         pen drive  500
6      1030         keyboard   500
7      1031         DVD drive 2500
7 rows selected.
```

Write a query to Rollback completely and verify it.

```
SQL> rollback;
Rollback complete.
To Verify:
SQL> select *from goods;

SNO    ITEMCODE     ITEMNAME   COST
------ ------------ ---------- ------

1      1025         moniter    5000
2      1026         mouse      250
3      1027         RAM        1500
4      1028         webcam     350
5      1029         pendrive   500
```

DCL Commands:

Grant:

```
GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;

SQL> GRANT SELECT ON Users TO'Tom'@'localhost;
```

Revoke:

```
REVOKE privilege_nameON object_nameFROM {user_name |PUBLIC |role_name}

SQL>REVOKE SELECT, UPDATE ON student FROM BCA, MCA;
```

RESULT:

Thus the TCL and DCL commands in SQL is successfully executed.

EX.No:8

TRIGGERS

AIM:

To write a PL/SQL query to create triggers.

Algorithm:

Step 1: Start a program by creating table
Step2: Enter the input
Step 3: The output will be displayed based on the queryStep 5: Stop the program

Program:

```
select * from account;
     +      +      +
     | sum | amount |
     +      +      +
     | 10 |   20 |
     | 20 |   30 |
     | 30 |   40 |
     +      +      +

Delimiter $$
create trigger update_trigger before update on account for each row
begin
if new.sum=50 thenset new.sum=100;
else if new. Sum>50 then set new.sum=200;
end if;
end$$
```

Output function:

```
delimiter $$
MySQL> update account set sum=50
$$select * from account;
 -> $$
```

OUTPUT:

```
+-------+---------+
| sum  | amount |
+-------+---------+
|  100 |    20 |
|  100 |    30 |
|  100 |    40 |
+-------+---------+
```

RESULT:

Thus the program for creation of triggers and functions is executed successfully

EX.No:9

VIEWS AND INDEX

AIM:

To Create views and index for database tables.

PROGRAM:

SQL> desc emp;

Name	Null?	Type
EMPNO	NOT NULL	NUMBER (4)
ENAME	NOT NULL	VARCHAR2 (30)
JOB		VARCHAR2 (10)
MGR		NUMBER (4)
HIREDATESAL		DATE NUMBER (7,2)
DEPTNO		NUMBER (2)

SQL> select * from emp;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	DEPTNO
3737	Priya	Officer	7777	07-MAR-09	51000	7
4545	Priya	Secretary	3337	09-JUN-09	21000	20
7575	Karthi	Manager	3337	07-JUL-07	75000	20
5352	Retish	Secretary	5555	09-JUN-09	20000	8
5332	Rocky	Assist	5555	08-JAN-08	19500	7
6544	Santh	Officer	3337	23-JAN-09	19000	8

```
6 rows selected.
-------------------------------------------- ----------- ----------------------
EMPNO                     NUMBER(4)

PH_NO                     NUMBER(8)
```

```
SQL> select * fromemp;
```

EMPNO	PH_NO
3737	225301
4545	485565
6544	789663
7575	896652
6555	987777

<u>CREATE VIEW</u>

View created from more than one table leading to ‘Read Only’ view.

```
SQL> create view view1 as
   select emp.ename,empp.ph_no from emp,empp where emp.empno=empp.empno;View created.

SQL> select * from view1;
ENAME                PH_NO
-----------------------------
Priya                225301
priya                485565
santh                789663
Karthi               896652
```

INDEX:

```
SQL> select * from emp;

Emp_Id  Emp_Name   Emp_Salary      Emp_City        Emp_State
1001    Akshay     20000              Noida           U.P
1002    Ram        35000              Jaipur
                                      Rajasthan
1003    Shyam      25000              Gurgaon     Haryana
1004    Yatin      30000              Lucknow        U.P

SQL>CREATE INDEX index_state ON Employee (Emp_State);
SQL> CREATE INDEX index_city_State ON Employee (Emp_City, Emp_State);
   Index created
```

UNIQUE INDEX

```
    SQL> CREATE UNIQUE INDEX Index_Name ON Table_Name ( Column_Name);

            Index created
```

ALTER AN INDEX

```
    SQL>  ALTER INDEX index_Salary RENAME TO  Index_Employee_Salary;

            Index altered
```

DROP INDEX

```
    SQL> DROP INDEX index_salary;

            Index Dropped
```

RESULT:

Thus, the database views and indexes were successfully executed.

EX.No:10

XML DATABASE AND XML SCHEMA

AIM:

To create an xml database and validate it using xml schema.

ALGORITHM:

Step 1: Open an XML file in Visual Studio
Step 2: On the menu bar, choose XML > Create Schema.
Step 3: An XML Schema document is created and opened for each namespace found in the XML file
Step 4: The output will be displayed web page.
Step 5: Microsoft .NET Framework Class Library namespaces: System.Xml System.Xml.Schema

Create an XML document:

1. Start Microsoft Visual Studio or Microsoft Visual Studio .NET. Then, create a new

   XML file(on the File menu, point to New, and then click File).

2. Select the XML File type, and then click Open.

3. Add the following data to the XML document to represent a product in a catalog:

```
<Product ProductID="123">
<ProductName>Rugby jersey
</ProductName>
</Product>
```

4. Save the file as *Product.xml* in a folder that you will be able to readily access later.

Create a DTD and link to the XML document

1. In Visual Studio 2005 or in Visual Studio .NET, point to New on the File menu, and

   then clickFile.

2. Select the Text File type, and then click Open.

3. Add the following DTD declarations to the file to describe the grammar of the XML

   document:XML

```
<!ELEMENT Product (ProductName)>
<!ATTLIST Product ProductID CDATA #REQUIRED>
```

```
<!ELEMENT ProductName (#PCDATA)>
```

4. Save the file as *Product.dtd* in the same folder as your XML document.
5. Reopen *Product.xml* in Visual Studio 2005.

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE Product SYSTEM "Product.dtd">
```

6. Save the modified XML document as *ProductWithDTD.xml*.

<u>Create an XDR schema and link to the XML document</u>

1. In Visual Studio 2005 or in Visual Studio .NET, point to New on the File menu, and then click File.
2. Select the Text File type, and then click Open.
3. Add the following XDR schema definitions to the file to describe the grammar of the XML document:

```
<?xml version="1.0"?>
<Schema name="ProductSchema"
 xmlns="urn:schemas-microsoft-com:xml-data"
 xmlns:dt="urn:schemas-microsoft-
 com:datatypes">

    <AttributeType name="ProductID" dt:type="int"/>
    <ElementType name="ProductName" dt:type="string"/>
    <ElementType name="Product" content="eltOnly">
        <attribute type="ProductID" required="yes"/>
        <element type="ProductName"/>
    </ElementType>
</Schema>
```

4. Save the file as *Product.xdr* in the same folder as your XML document.
5. Reopen the original *Product.xml*, and then link it to the XDR schema, as follows:

```
<?xml version="1.0" encoding="utf-8" ?>
<Product ProductID="123" xmlns="x-schema:Product.xdr">
```

```
<ProductName>Rugby jersey</ProductName>
</Product>
```

6. Save the modified XML document as *ProductWithXDR.xm*

Create an XSD schema and link to the XML document

1. In Visual Studio .NET, point to New on the File menu, and then click File.
2. Select the Text File type, and then click Open.
3. Add the following XSD schema definition to the file to describe the grammar of the XML document:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Product">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="ProductName" type="xsd:string"/>
    </xsd:sequence>
    <xsd:attribute name="ProductID" use="required" type="xsd:int"/>
  </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

4. Save the file as Product.xsd in the same folder as your XML document.
5. Reopen the original *Product.xml*, and then link it to the XSD schema, as follows:

```
<?xml version="1.0" encoding="utf-8" ?>
<Product ProductID="123"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-
 instance"
 xsi:noNamespaceSchemaLocation="Product.xsd">
  <ProductName>Rugby jersey</ProductName>
</Product>
```

6. Save the modified XML document as *ProductWithXSD.xml*.

Use namespaces in the XSD schema

1. In Visual Studio 2005 or in Visual Studio .NET, open *ProductWithXSD.xml*. Declare a default namespace, urn:MyNamespace, in the document. Modify the XSD linkage to specify the XSD schema tovalidate content in this namespace, as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<Product ProductID="123"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-
 instance" xmlns="urn:MyNamespace"
 xsi:schemaLocation="urn:MyNamespace Product.xsd">
   <ProductName>Rugby jersey</ProductName>
</Product>
```

2. Save *ProductWithXSD.xml*.
3. Open Product.xsd, click the XML tab, and then modify the xsd:schema start tag as follows, so that theschema applies to the namespace urn:MyNamespace:

```
<xsd:schema
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="urn:MyNamespace"
 elementFormDefault="qualified">
```

4. Save *Product.xsd*.
5. Run the application to validate the XML document by using the XSD schema.

Output:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="Product">
  <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="ProductName" type="xsd:string"/>
```

```
        </xsd:sequence>
        <xsd:attribute name="ProductID" use="required" type="xsd:int"/>
    </xsd:complexType>
    </xsd:element>
</xsd:schema>

<?xml version="1.0" encoding="utf-8" ?>
<Product ProductID="123"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-
 instance"
 xsi:noNamespaceSchemaLocation="Product.xsd">
    <ProductName>Rugby jersey</ProductName>
</Product>
```

RESULT:

The xml database was successfully created by display the web page of validate xml schema

Execute complex transactions and realize DCL and TCL commands.

EX.No:11

AIM:

To Create document, column and graph based data using NOSQL database tools

Outcome: document, column and graph based data using NoSQL.

PROGRAM:

Create Database:

```
>use javatpointdb

    Swithched to db javatpointdb
 >db
```

Check the Database:

```
>show dbs
    local 0          local 0.078GB
```

Insert a document:

```
>db.movie.insert({"name":"javatpoint"})

db.javatpoint.insert(
 {
   course:
    "java",
    details: {
      duration: "6 months",
      Trainer: "Sonoo
      jaiswal"
    },
    Batch: [ { size: "Small", qty: 15 }, { size: "Medium", qty: 25 } ],
     category: "Programming language"
   }
 )
      WriteResult({ "nInserted": 1})
```

Drop Database:

```
  > db.dropDatabase()

 { "dropped": "javatpointdb", "ok":
```

1}Create collection :

```
  >db.SSSIT.insert({"name" : "seomount"})
>show
collectionsSSSIT

db.collection_name.find()

{ "_id" : ObjectId("56482d3e27e53d2dbc93cef8"), "course" : "java", "details" :
{ "duration" : "6 months", "Trainer" : "Sonoo jaiswal" },
"Batch" :[ {"size" : "Small", "qty" : 15 }, { "size" : "Medium",
"qty" : 25 } ], "category" : "Programming language" }
```

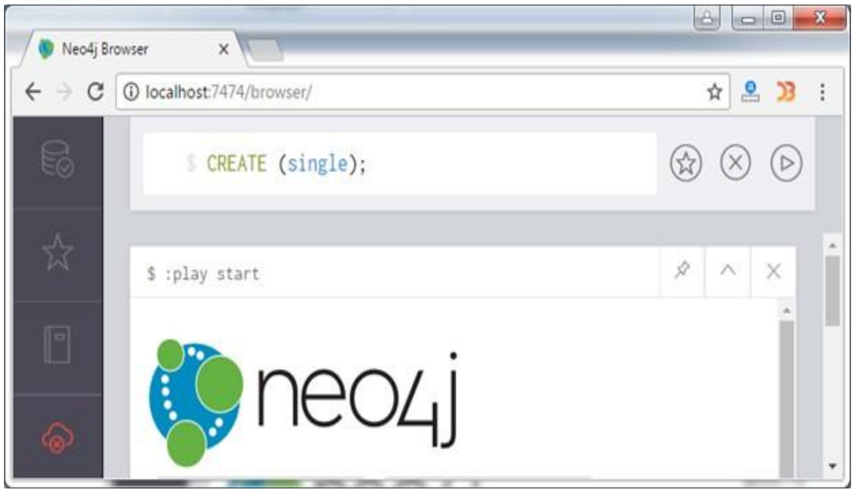
Neo4j CQL

Create

nodes:

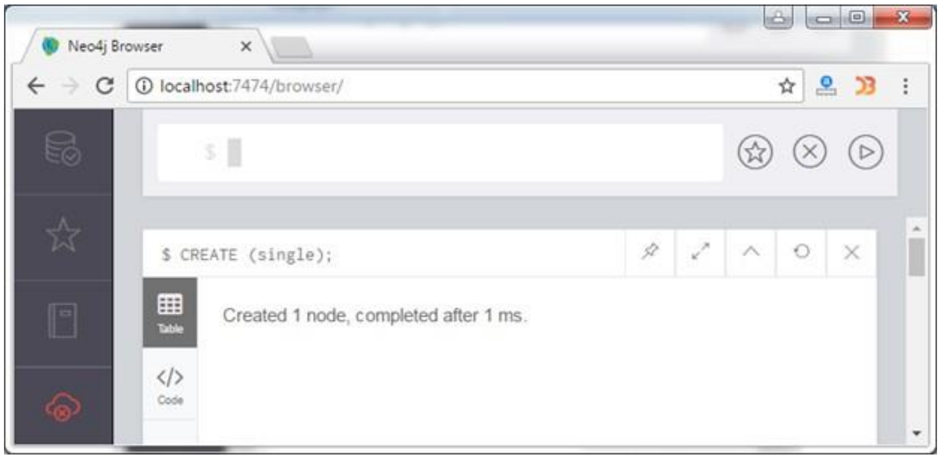
Open the localhost on the browser: http://localhost:7474/browser/ and use the following code:

```
CREATE (single);
```



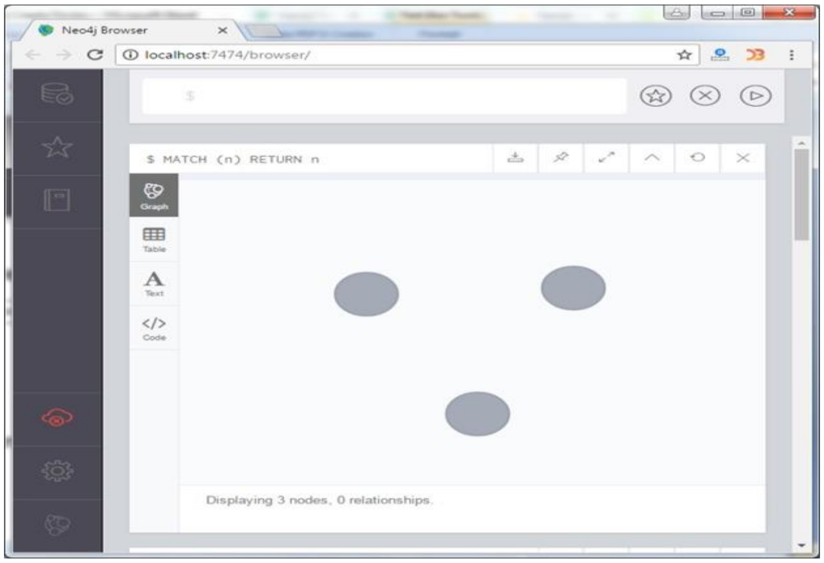
```
$ CREATE (single);
```

```
$ :play start
```



```
$ CREATE (single);
Created 1 node, completed after 1 ms.
```

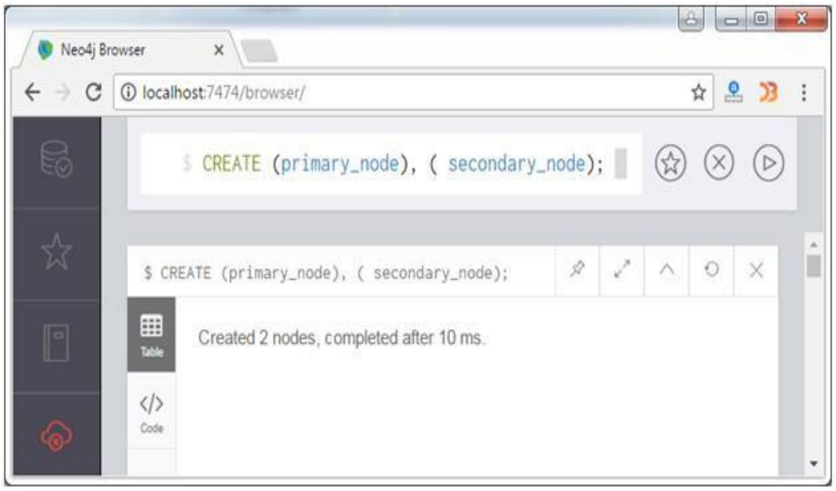
MATCH (n) RETURN (n)



```
$ MATCH (n) RETURN n
```

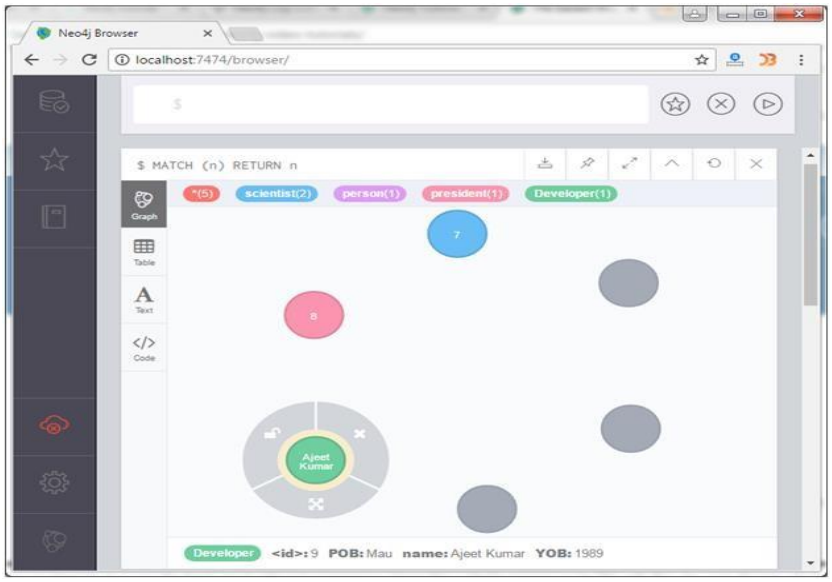
Displaying 3 nodes, 0 relationships.

CREATE (primary_node), (secondary_node);



```
$ CREATE (primary_node), ( secondary_node);
```

Created 2 nodes, completed after 10 ms.



```
$ MATCH (n) RETURN n
```

Developer <id>: 9 POB: Mau name: Ajeet Kumar YOB: 1989

RESULT:

Thus the NoSQL database tool successfully designed and executed.

EX.NO.12

DATABASE GUI APLLICATION – PAYROLL MANAGEMENT

AIM:

To design a staff payroll management System using VB with mysql as backend.

SAMPLE CODE:

```
Dim db As
 DatabaseDim rs As Recordset
Private Sub
cmdADD_Click()
 rs.AddNew DtPicker
DOJ.Enabled = True
txtEmpID.Enabled = True
txtFirstName.Enabled = True
txtLastName.Enabled = True
txtAddress.Enabled = True
 txtDOJ.Enabled = True
txtSalary.Enabled = True
 txtEmpID.SetFocus
 txtEmpID.Text = ""
txtFirstName.Text = ""
 txtLastName.Text = ""
txtAddress.Text = ""
```

```
txtDOJ.Text=Date

txtSalary.Text =""

          txtHRA.Text = ""

          txtDA.Text = ""

          txtTA.Text = ""

          txtPF.Text= ""

          txtGrossPay.Text = ""

          txtNetPay.Text = ""

          cmdEdit.Enabled=False

          cmdSubmit.Enabled = False

          cmdCalculate.Enabled=

          FalseEnd Sub


         Private Sub

         cmdCalculate_Click()If

         txtSalary.Text = "" Then

          MsgBox "SALARY FIELD IS EMPTY", vbCritical, "Employees

          project"txtSalary.SetFocusElse

          txtHRA.Text = 0.15 *

          Val(txtSalary.Text)txtDA.Text = 0.17

          * Val(txtSalary.Text) txtTA.Text =

          0.13 * Val(txtSalary.Text) txtPF.Text

          = 0.05 * Val(txtSalary.Text)

          txtGrossPay.Text = Val(txtHRA.Text) + Val(txtDA.Text) + Val(txtTA.Text) +
          Val(txtSalary.Text)txtNetPay.Text
```

```
= Val(txtGrossPay.Text) - Val(txtPF.Text)

cmdSubmit.Enabled =

TruecmdADD.Enabled =

False

End If End Sub
Private Sub cmdDelete_Click()

Dim msgboxres As VbMsgBoxResult

If rs.RecordCount = 0 Or txtEmpID.Text = "" Then

MsgBox "No records to delete", vbCritical, "Employees Project"

Else

msgboxres = MsgBox("Do you really want to Delete?", vbCritical + vbOKCancel, "Confirmation")If

msgboxres = vbOK Then

Cancel=False

rs.Delete

MsgBox "RECORD DELETED", vbCritical, "Employees

Project"ClearALL

Else

Cancel=True

End If

'rs.MoveNex

tEnd If

End Sub

 txtFirstName.Enabled = True

txtLastName.Enabled = True

txtAddress.Enabled = True
```

```
txtDOJ.Enabled = True

txtSalary.Enabled= True

rs("FIRSTNAME")

txtFirstName.Textrs("LASTNAME")

txtLastName.Text rs("ADDRESS")

txtAddress.Text

rs("DATEOFJOINING") =txtDOJ.Text

rs("SALARY") =txtSalary.Text

rs("HRA") = txtHRA.Text

rs("DA") = txtDA.Text

rs("TA") =txtTA.Text

rs("PF") = txtPF.Text rs("GROSSPAY") =txtGrossPay.Textrs
rs("NETPAY") =txtNetPay.Text
End Sub Private Sub cmdExit_Click()
 Dim msgboxres As VbMsgBoxResult
  msgboxres = MsgBox("Do you want to really exit?", vbCritical + vbOKCancel,

"Confirmation")Ifmsgboxres = vbOK Then

Cancel=False

End

Else

Cancel=True
End If

End Sub

Private Sub

cmdFirst_Cli

ck()

rs.MoveFirst
```

```
            cmdPrev.Ena

            bled = False

            cmdFirst.En

            abled = False

cmdNext.Enabled = True

cmdLast.Enabled = True

cmdEdit.Enabled = True GetData

            End Sub
           Private Sub cmdLast_Click()
           rs.MoveLast cmdFirst.Enabled = True
            cmdPrev.Enabled = True
           cmdNext.Enabled = False
           cmdLast.Enabled = False
            cmdEdit.Enabled = True
            GetData
           End Sub
           cmdEdit.Enabled = True
           cmdFirst.Enabled = True
           cmdPrev.Enabled=True
           rs.MoveNext
           If rs.EOF = True
            Then
           cmdNext.Enabled=False
           ecmdLast.Enabled = False
           MsgBox "LAST RECORD", vbInformation, "Employees Project"rs.MoveLast
           Else GetData
           End If
           End Sub
           Private Sub cmdPrev_Click()cmdEdit.Enabled = True
           cmdFirst.Enabled = True
           cmdNext.Enabled = True
           cmdLast.Enabled = True
           rs.MovePrevious
           If rs.BOF = True
            Then
           cmdFirst.Enabled = False
           cmdPrev.Enabled = False
           MsgBox "FIRST RECORD", vbInformation, "Employees Project"

            rs.MoveFirs
```

```
tElse

GetData

End If


GetDat

aEnd

Sub


Private Sub

cmdReport_Click()

DataReport1.Show

End Sub

Private Sub

cmdSubmit_Click() On Error

GoTo errhandler2

rs("EMPID")=txtEmpID.Text

rs("FIRSTNAME") = txtFirstName.Text

rs("LASTNAME") = txtLastName.Text

 rs("ADDRESS") = txtAddress.Text

rs("DATEOFJOINING") = txtDOJ.Text

rs("SALARY") = txtSalary.Text
rs("HRA") = txtHRA.Text
rs("DA")  = txtDA.Text
rs("TA") =txtTA.Text
rs("PF") = txtPF.Text
 rs("GROSSPAY") =txtGrossPay.Text
rs("NETPAY") =txtNetpay.Text
```

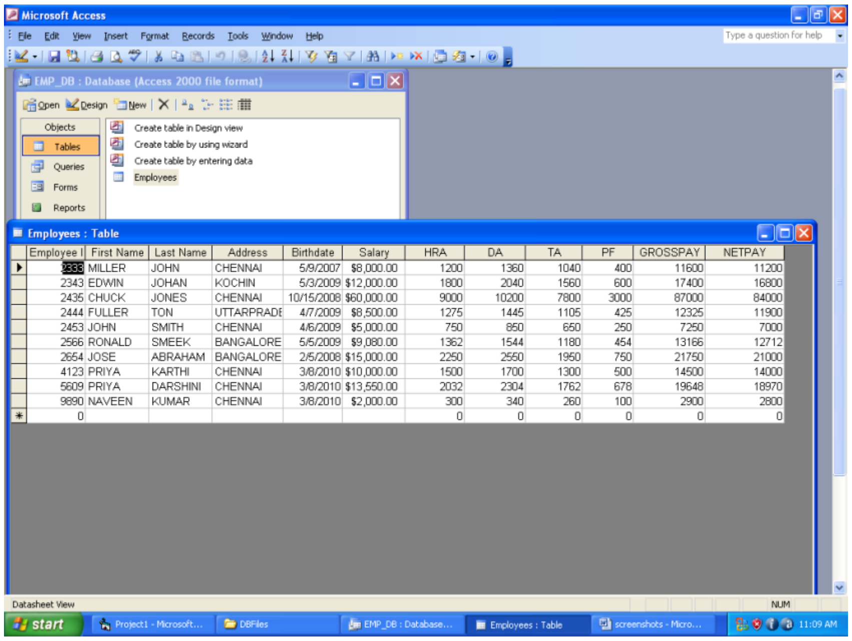
```
Private Sub

txtAddress_Change()

cmdSubmit.Enabled = True

 End Sub

 Private Sub

 txtDOJ_Change()

 cmdSubmit.Enabled =

 True End Sub

 [Private Sub

 txtEmpID_Change()

 cmdSubmit.Enabled = True

 End Sub

 Private Sub txtEmpID_KeyPress(KeyAscii As

 Integer)If(KeyAscii < 48 Or KeyAscii > 57) Then

 KeyAscii = 0

 MsgBox "PLEASE ENTER ONLY NUMBERS", vbCritical, "Employees

 Project"End IfEnd Sub

Private Sub

txtFirstName_Change()

cmdSubmit.Enabled = True

 End Sub

 Private Sub

 txtLastName_Change()
```

```
cmdSubmit.Enabled = True

End Sub
Private Sub

txtSalary_Change()If

txtSalary.Text <> "" Then



Cmd Calculate.Enabled = True
End If
End Sub


Public Sub ClearALL()Dim ctrl As Control
For Each ctrl In Me.Controls

If TypeOf ctrl Is TextBox
Then
ctrl.Text = ""Next
End Sub



Public Sub GetData()

If rs.RecordCount <> 0
Then
 txtEmpID.Text = rs("EMPID")
 txtFirstName.Text = rs("FIRSTNAME")
txtLastName.Text = rs("LASTNAME")
 txtAddress.Text = rs("ADDRESS")
txtDOJ.Text = rs("DATEOFJOINING")
txtSalary.Text = rs("SALARY")

txtHRA.Text = rs("HRA")
txtDA.Text = rs("DA")
txtTA.Text = rs("TA")
 txtPF.Text=rs("PF")
txtGrossPay.Text = rs("GROSSPAY")
txtNetPay.Text = rs("NETPAY")

txtEmpID.Enabled = False

txtFirstName.Enabled = False

txtLastName.Enabled = False
```

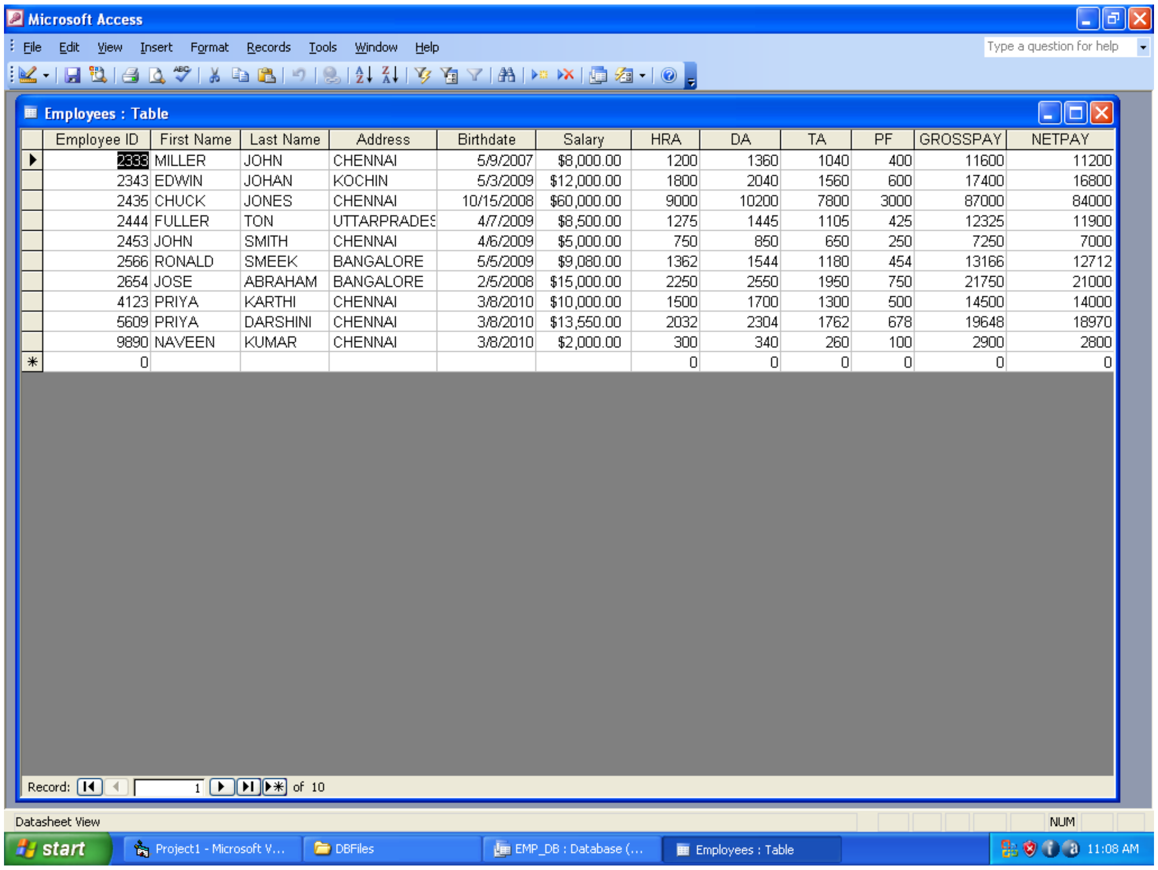
```
txtAddress.Enabled = False

txtDOJ.Enabled = False

txtSalary.Enabled = False

txtHRA.Enabled = False

txtDA.Enabled= False

 txtTA.Enabled = False

 txtPF.Enabled = False

 txtGrossPay.Enabled =False

 txtNetPay.Enabled = False

 cmdSubmit.Enabled = False

 cmdCalculate.Enabled=Fals

 DtPickerDOJ.Enabled=False

 Else

txtEmpID.Enabled = False
txtFirstName.Enabled = False
txtLastName.Enabled = False
txtAddress.Enabled = False
txtDOJ.Enabled = False
txtSalary.Enabled = False
 txtHRA.Enabled = False
```

```
txtDA.Enabled = False txtTA.Enabled = False txtPF.Enabled = False
txtGrossPay.Enabled = False
txtNetPay.Enabled = False c
mdSubmit.Enabled = False
cmdCalculate.Enabled = FalseEnd If
End Sub

        PrivateSubtxtSalary_KeyPress(KeyAsciiAs
        Integer)

        If(KeyAscii < 48 Or KeyAscii > 57) Then
        KeyAscii = 0

        MsgBox "PLEASE ENTER ONLY NUMBERS", vbCritical, "Employees
        Project"End IfEnd Sub
Public Sub DisableFields() txtEmpID.Enabled = False
txtFirstName.Enabled = False
txtLastName.Enabled = False
txtAddress.Enabled = False
txtDOJ.Enabled = False
txtSalary.Enabled = False

cmdADD.Enabled = True
cmdCalculate.Enabled = False
cmdSubmit.Enabled = False
```

DATABASE IN MS ACCESS:

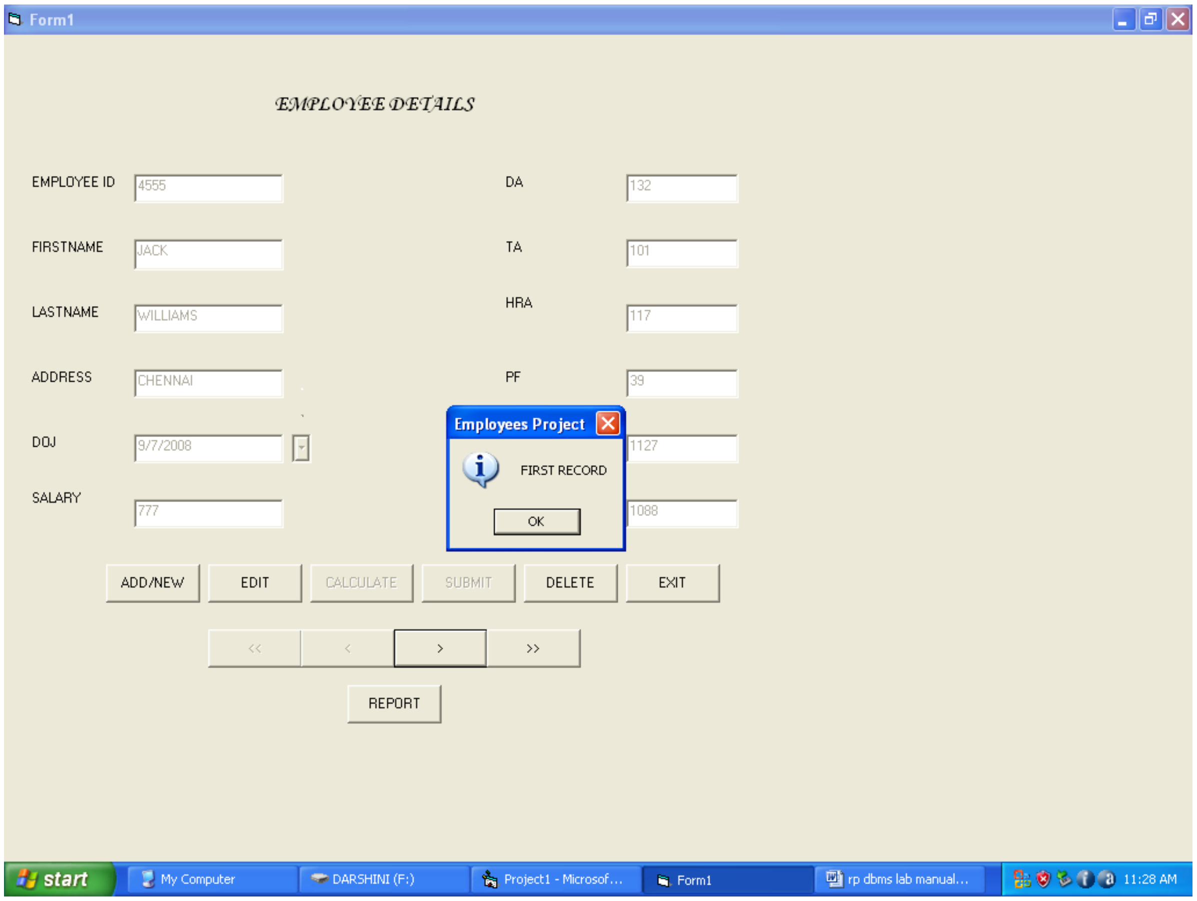


Employee I	First Name	Last Name	Address	Birthdate	Salary	HRA	DA	TA	PF	GROSSPAY	NETPAY
2333	MILLER	JOHN	CHENNAI	5/9/2007	$8,000.00	1200	1360	1040	400	11600	11200
2343	EDWIN	JOHAN	KOCHIN	5/3/2009	$12,000.00	1800	2040	1560	600	17400	16800
2435	CHUCK	JONES	CHENNAI	10/15/2008	$60,000.00	9000	10200	7800	3000	87000	84000
2444	FULLER	TON	UTTARPRADE	4/7/2009	$8,500.00	1275	1445	1105	425	12325	11900
2453	JOHN	SMITH	CHENNAI	4/6/2009	$5,000.00	750	850	650	250	7250	7000
2566	RONALD	SMEEK	BANGALORE	5/5/2009	$9,080.00	1362	1544	1180	454	13166	12712
2654	JOSE	ABRAHAM	BANGALORE	2/5/2008	$15,000.00	2250	2550	1950	750	21750	21000
4123	PRIYA	KARTHI	CHENNAI	3/8/2010	$10,000.00	1500	1700	1300	500	14500	14000
5609	PRIYA	DARSHINI	CHENNAI	3/8/2010	$13,550.00	2032	2304	1762	678	19648	18970
9890	NAVEEN	KUMAR	CHENNAI	3/8/2010	$2,000.00	300	340	260	100	2900	2800
0						0	0	0	0	0	0



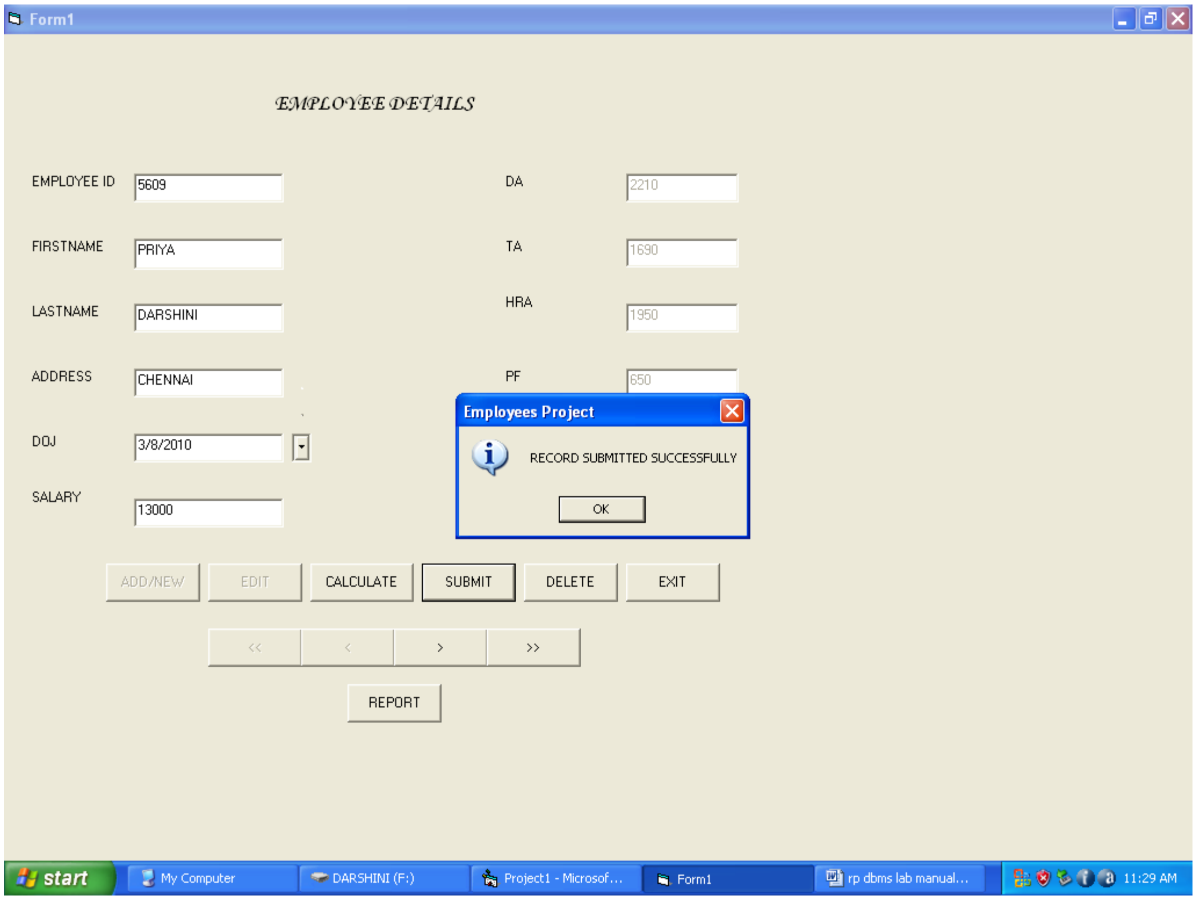
Employee ID	First Name	Last Name	Address	Birthdate	Salary	HRA	DA	TA	PF	GROSSPAY	NETPAY
2333	MILLER	JOHN	CHENNAI	5/9/2007	$8,000.00	1200	1360	1040	400	11600	11200
2343	EDWIN	JOHAN	KOCHIN	5/3/2009	$12,000.00	1800	2040	1560	600	17400	16800
2435	CHUCK	JONES	CHENNAI	10/15/2008	$60,000.00	9000	10200	7800	3000	87000	84000
2444	FULLER	TON	UTTARPRADES	4/7/2009	$8,500.00	1275	1445	1105	425	12325	11900
2453	JOHN	SMITH	CHENNAI	4/6/2009	$5,000.00	750	850	650	250	7250	7000
2566	RONALD	SMEEK	BANGALORE	5/5/2009	$9,080.00	1362	1544	1180	454	13166	12712
2654	JOSE	ABRAHAM	BANGALORE	2/5/2008	$15,000.00	2250	2550	1950	750	21750	21000
4123	PRIYA	KARTHI	CHENNAI	3/8/2010	$10,000.00	1500	1700	1300	500	14500	14000
5609	PRIYA	DARSHINI	CHENNAI	3/8/2010	$13,550.00	2032	2304	1762	678	19648	18970
9890	NAVEEN	KUMAR	CHENNAI	3/8/2010	$2,000.00	300	340	260	100	2900	2800
0						0	0	0	0	0	0

FORMS WHILE EXECUTION:



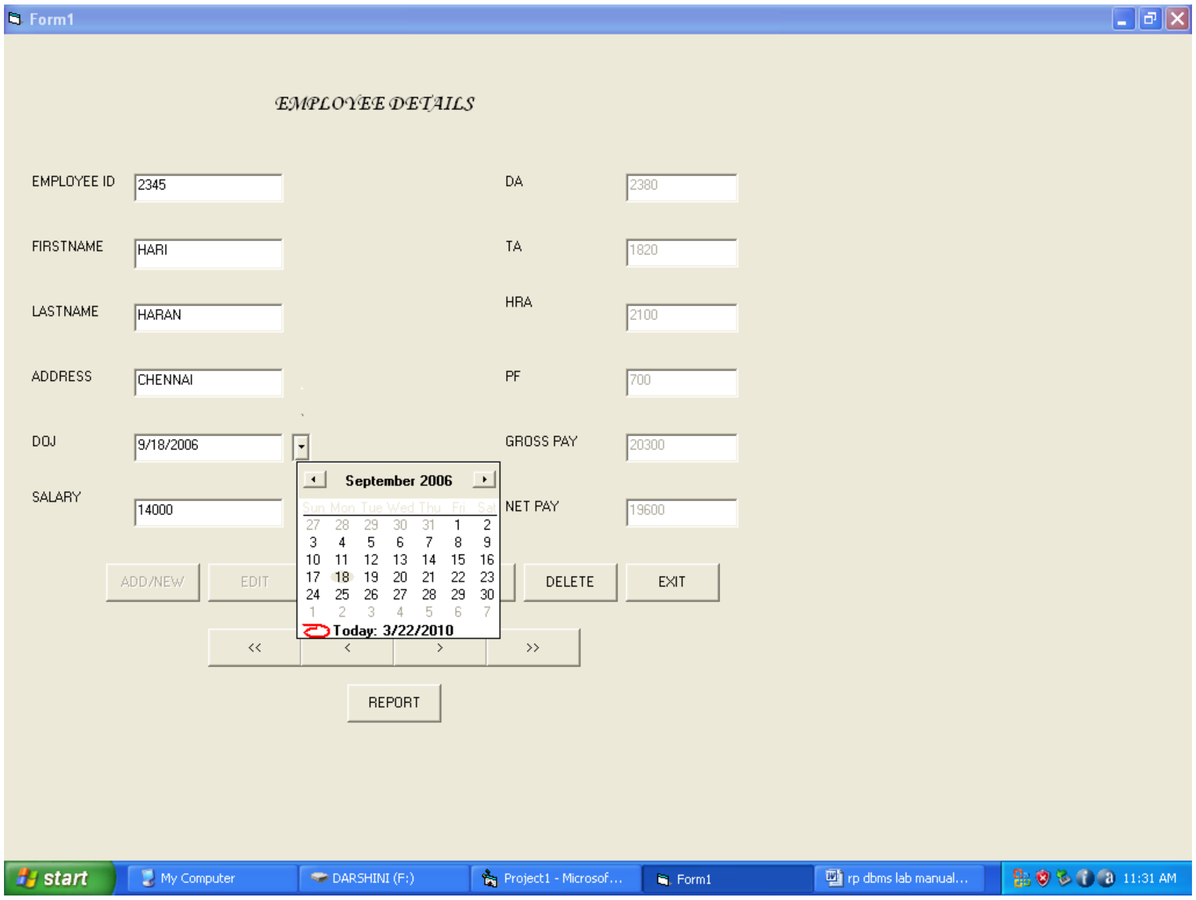
EMPLOYEE ID	4555	DA	132
FIRSTNAME	JACK	TA	101
LASTNAME	WILLIAMS	HRA	117
ADDRESS	CHENNAI	PF	39
DOJ	9/7/2008		1127
SALARY	777		1088

FIRST RECORD



EMPLOYEE ID	5609	DA	2210
FIRSTNAME	PRIYA	TA	1690
LASTNAME	DARSHINI	HRA	1950
ADDRESS	CHENNAI	PF	650
DOJ	3/8/2010		
SALARY	13000		

RECORD SUBMITTED SUCCESSFULLY



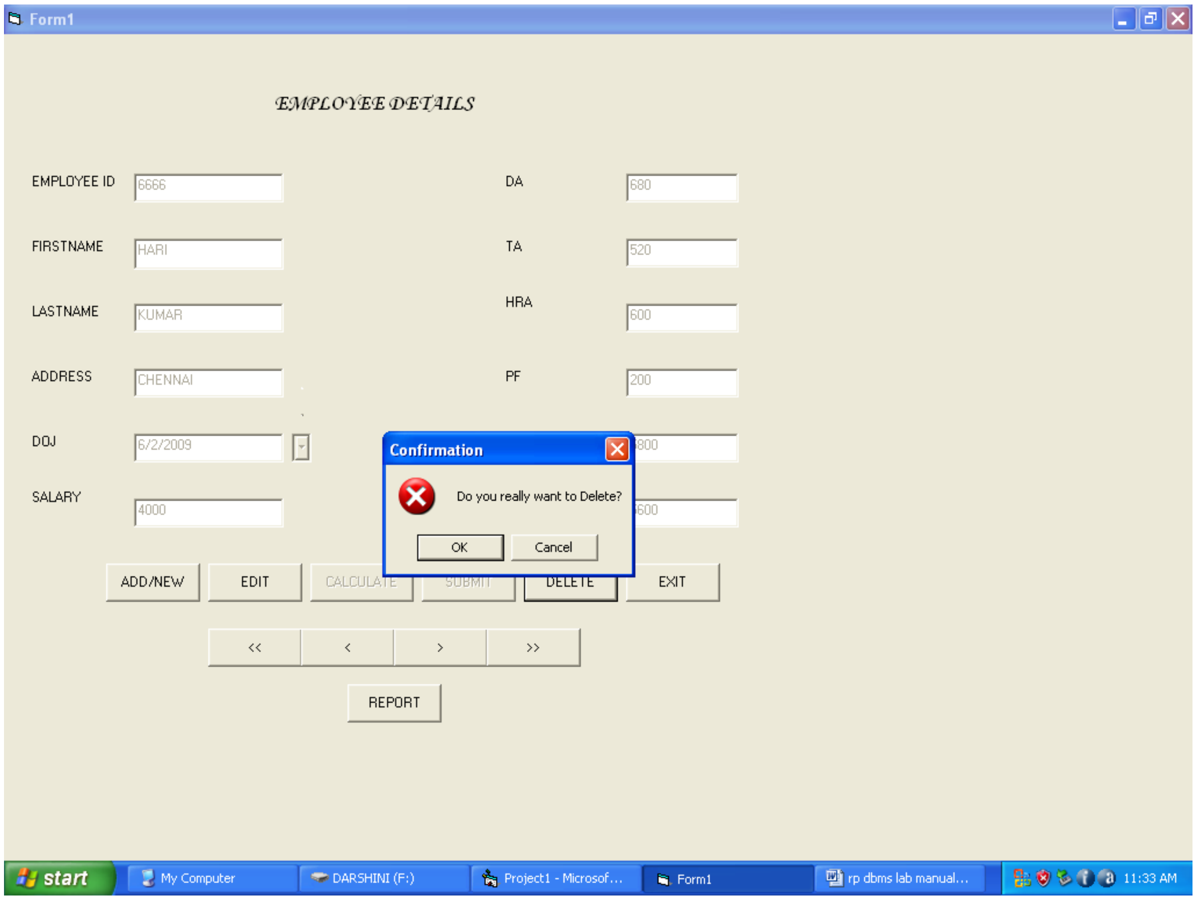
EMPLOYEE DETAILS

EMPLOYEE ID	2345	DA	2380
FIRSTNAME	HARI	TA	1820
LASTNAME	HARAN	HRA	2100
ADDRESS	CHENNAI	PF	700
DOJ	9/18/2006	GROSS PAY	20300
SALARY	14000	NET PAY	19600

September 2006

Sun	Mon	Tue	Wed	Thu	Fri	Sat
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

Today: 3/22/2010

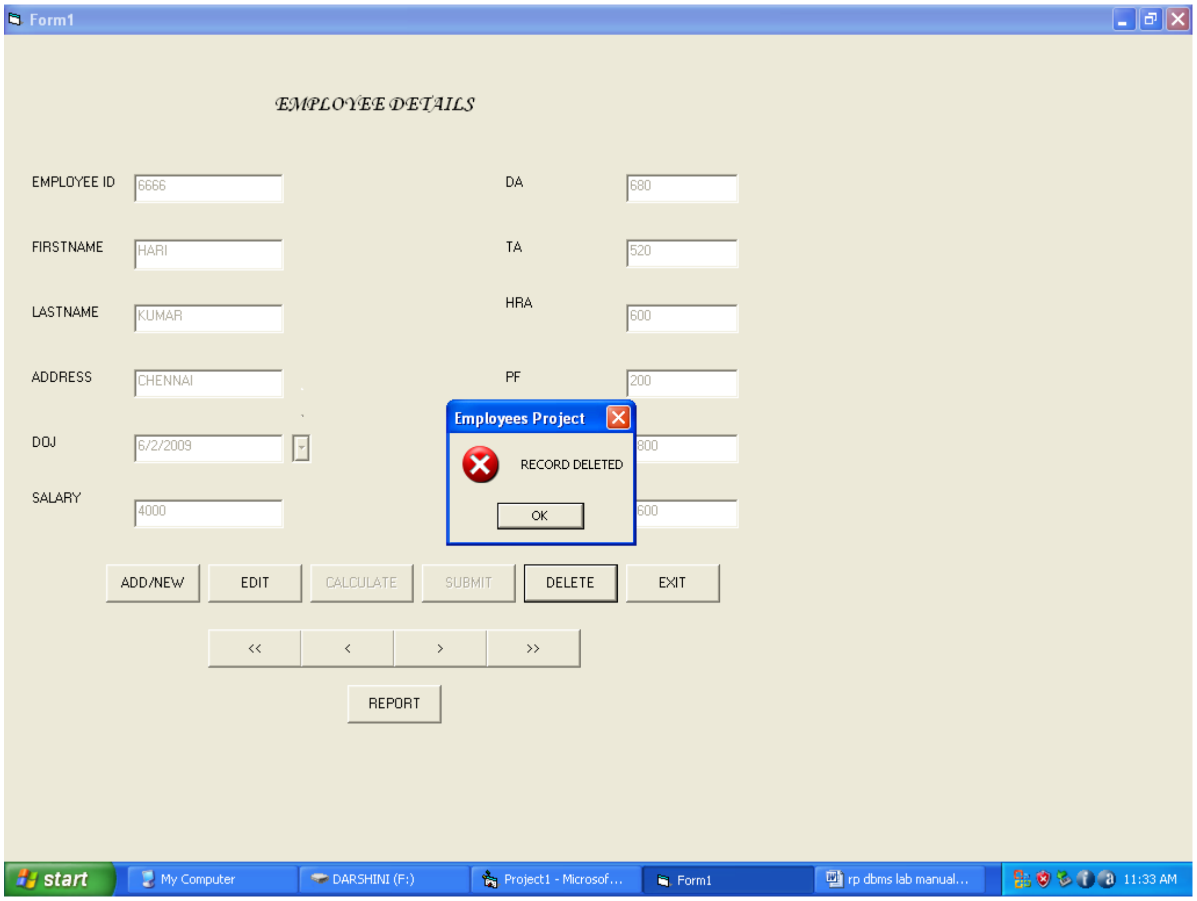

Form1

EMPLOYEE DETAILS

EMPLOYEE ID	6666	DA	680
FIRSTNAME	HARI	TA	520
LASTNAME	KUMAR	HRA	600
ADDRESS	CHENNAI	PF	200
DOJ	6/2/2009		800
SALARY	4000		600

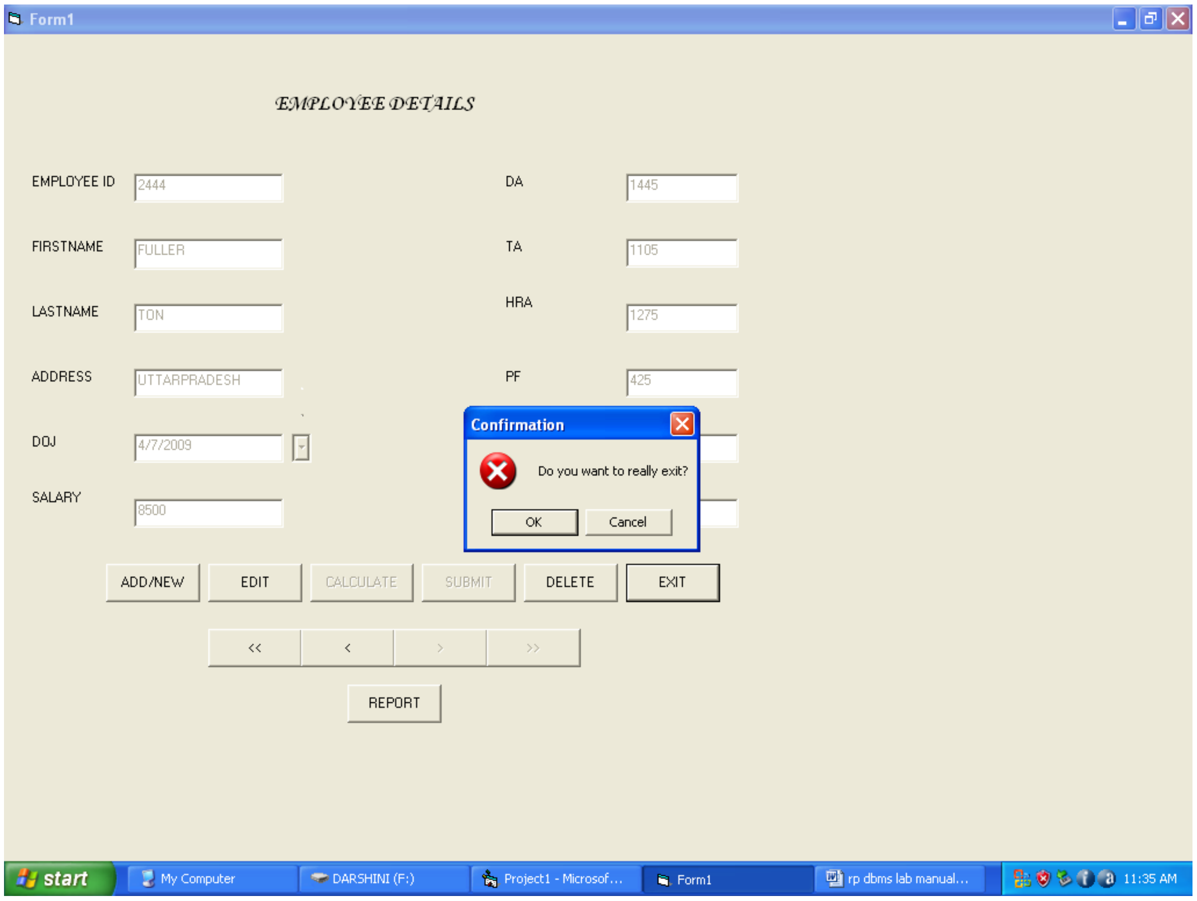
Confirmation

Do you really want to Delete?



EMPLOYEE ID 6666
FIRSTNAME HARI
LASTNAME KUMAR
ADDRESS CHENNAI
DOJ 6/2/2009
SALARY 4000
DA 680
TA 520
HRA 600
PF 200
800
600

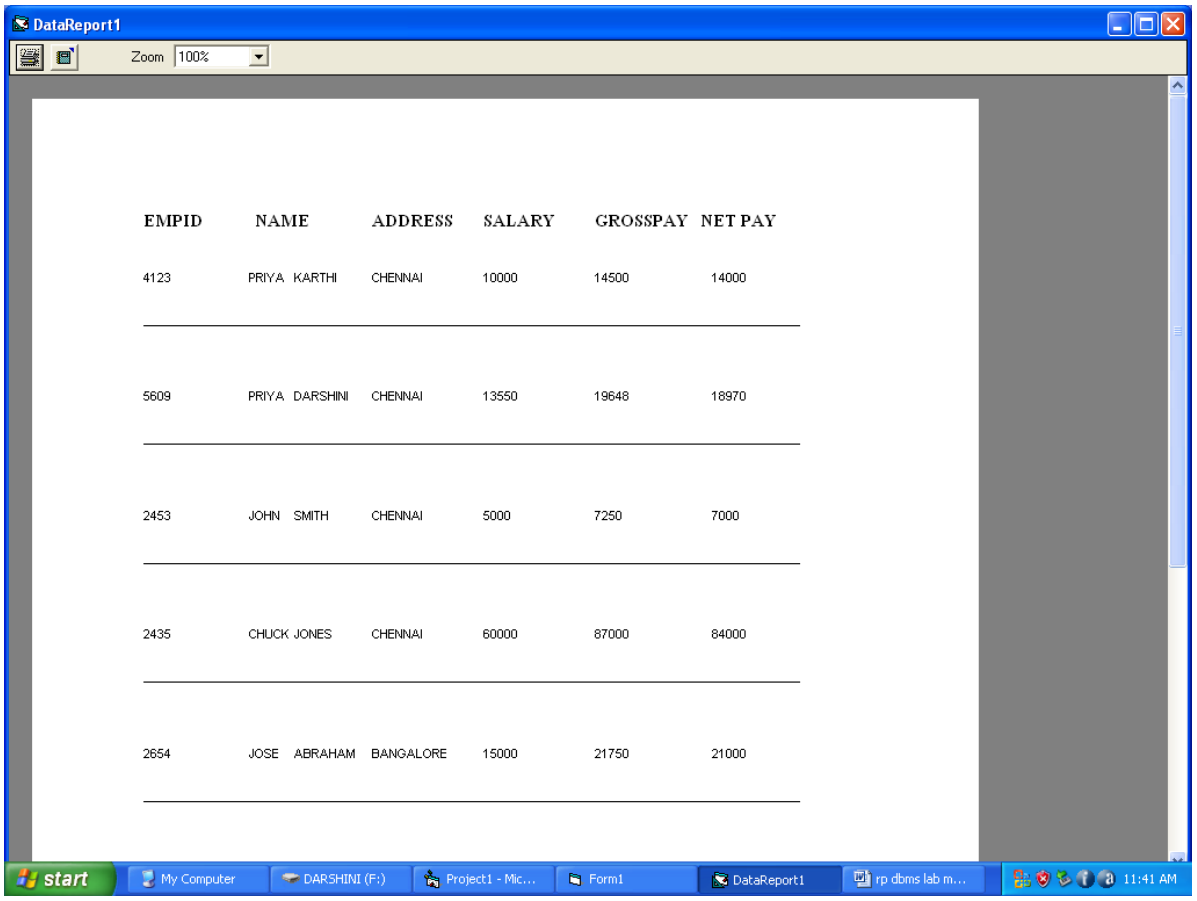
Employees Project
RECORD DELETED
OK


EMPLOYEE DETAILS

EMPLOYEE ID	2444	DA	1445
FIRSTNAME	FULLER	TA	1105
LASTNAME	TON	HRA	1275
ADDRESS	UTTARPRADESH	PF	425
DOJ	4/7/2009		
SALARY	8500		

Confirmation

Do you want to really exit?



DataReport1

Zoom 100%

EMPID	NAME	ADDRESS	SALARY	GROSSPAY	NET PAY
4123	PRIYA KARTHI	CHENNAI	10000	14500	14000
5609	PRIYA DARSHINI	CHENNAI	13550	19648	18970
2453	JOHN SMITH	CHENNAI	5000	7250	7000
2435	CHUCK JONES	CHENNAI	60000	87000	84000
2654	JOSE ABRAHAM	BANGALORE	15000	21750	21000

RESULT:

Thus the Student Personal Information System is successfully designed using PHPwith mysql as backend.

EX.NO:13

CASE STUDY E-Mart GroceryShop

AIM:

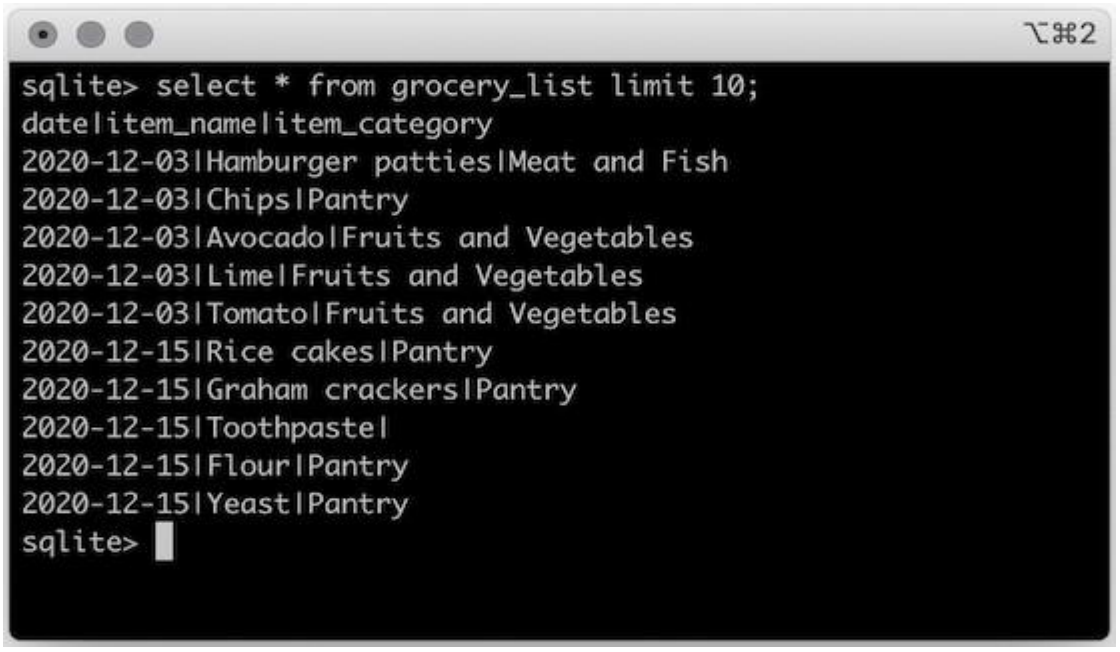
To create a database for E-mart Grocery shop and apply all sql properties.

SAMPLE CODE:

```
 SQL> create table grocery_visit(date TEXT, time_spent_min INTEGER, amount_spent REAL);
SQL>create table grocery_list(date TEXT, item_name TEXT, item_category TEXT);
insert into grocery_list values("2020-12-03", "Hamburger patties", "Meat and Fish");
insert into  grocery_list  values("2020-12-03", "Chips", "Pantry");
insert into grocery_list values("2020-12-03", "Avocado", "Fruits and
Vegetables");
insert into grocery_list values("2020-12-03", "Lime", "Fruits and
Vegetables");
insert into grocery_list values("2020-12-03", "Tomato", "Fruits and
Vegetables");
insert into grocery_list values("2020-12-15", "Rice cakes", "Pantry");
insert into grocery_list values("2020-12-15","Graham crackers", "Pantry");
insert into grocery_list values("2020-12-15",
"Toothpaste", NULL);
insert into grocery_list values("2020-12-15", "Flour",
"Pantry");
 insert into grocery_list values("2020-12-15", "Yeast",
"Pantry");
insert into grocery_list values("2020-12-15",
"Popcorn", "Pantry");
insert into grocery_list values("2020-12-15", "Eggs",
NULL);
insert into grocery_list values("2020-12-15", "Milk", "Dairy");
insert into grocery_list values("2020-12-15", "Bananas", "Fruits and Vegetables");
insert into grocery_listvalues("2020-12-15", "Frozen waffles", NULL);
insert    into    grocery_list    values("2020-12-23",
"Mayo", "Pantry");
insert    into    grocery_list    values("2020-12-23",
"Flour", "Pantry");
insert into grocery_list values("2020-12-23", "Milk",
"Dairy");
insert into grocery_list values("2020-12-23", "Roasted Chicken", "Meat and Fish");
insert into grocery_listvalues("2020-12-23", "Chocolate chip cookies", "Pantry");
insert into grocery_list values ("2020-12-23",
"Yogurt", "Dairy");
insert into grocery_list values("2020-12-
23","Soda",NULL);
```

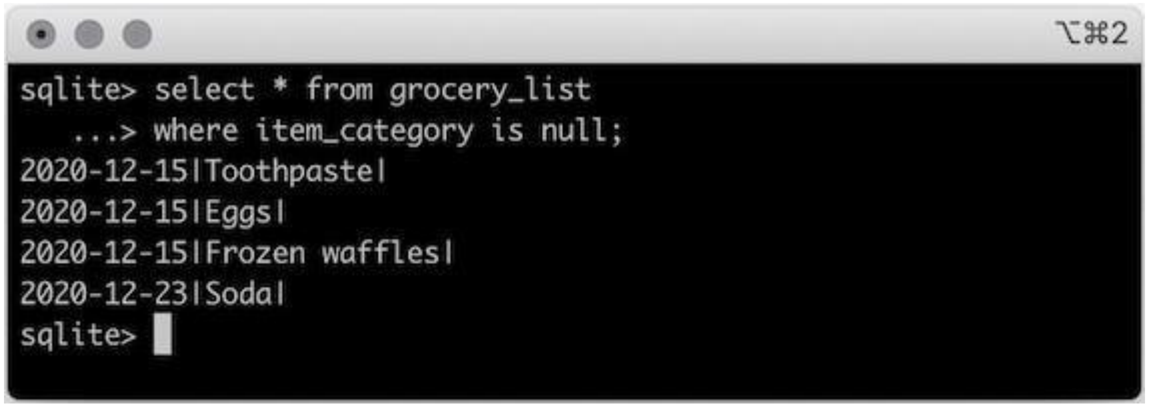
insert into grocery list values ("2020-12-23", "Grapes", "Fruits and Vegetables");

select * from grocery_list limit 10;



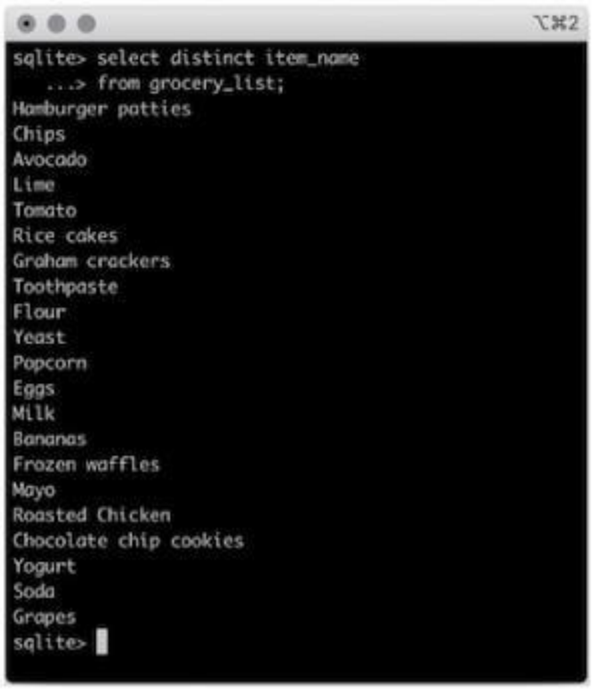
```
sqlite> select * from grocery_list limit 10;
date|item_name|item_category
2020-12-03|Hamburger patties|Meat and Fish
2020-12-03|Chips|Pantry
2020-12-03|Avocado|Fruits and Vegetables
2020-12-03|Lime|Fruits and Vegetables
2020-12-03|Tomato|Fruits and Vegetables
2020-12-15|Rice cakes|Pantry
2020-12-15|Graham crackers|Pantry
2020-12-15|Toothpaste|
2020-12-15|Flour|Pantry
2020-12-15|Yeast|Pantry
sqlite>
```

select * from grocery_list where item_category is null;

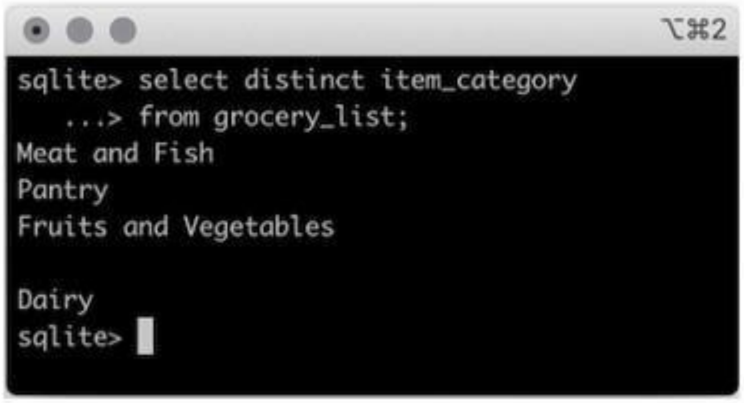


```
sqlite> select * from grocery_list
   ...> where item_category is null;
2020-12-15|Toothpaste|
2020-12-15|Eggs|
2020-12-15|Frozen waffles|
2020-12-23|Soda|
sqlite>
```

select distinct item_name from grocery_list;select distinct
item_categoryfrom grocery_list;

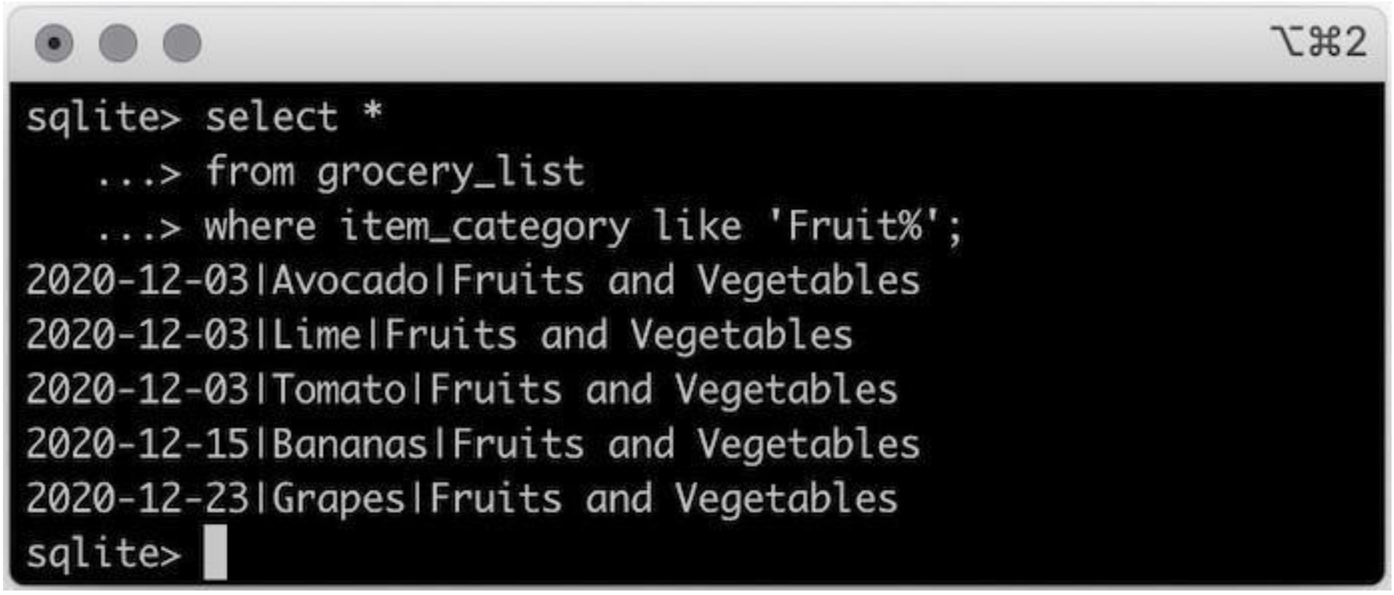


```
sqlite> select distinct item_name
   ...> from grocery_list;
Hamburger patties
Chips
Avocado
Lime
Tomato
Rice cakes
Graham crackers
Toothpaste
Flour
Yeast
Popcorn
Eggs
Milk
Bananas
Frozen waffles
Mayo
Roasted Chicken
Chocolate chip cookies
Yogurt
Soda
Grapes
sqlite>
```



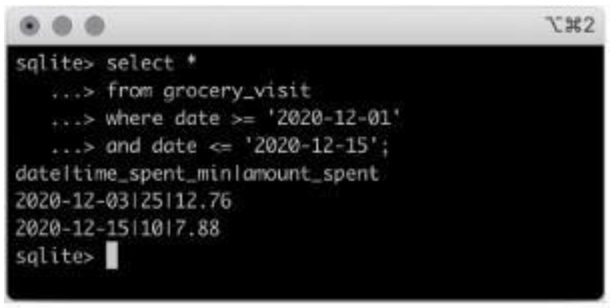
```
sqlite> select distinct item_category
   ...> from grocery_list;
Meat and Fish
Pantry
Fruits and Vegetables

Dairy
sqlite>
```

select *
from grocery_list
where item_category like 'Fruit%';

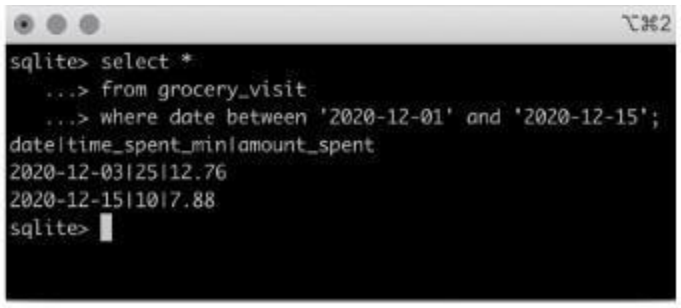


```
sqlite> select *
   ...> from grocery_list
   ...> where item_category like 'Fruit%';
2020-12-03|Avocado|Fruits and Vegetables
2020-12-03|Lime|Fruits and Vegetables
2020-12-03|Tomato|Fruits and Vegetables
2020-12-15|Bananas|Fruits and Vegetables
2020-12-23|Grapes|Fruits and Vegetables
sqlite>
```

select * from grocery_visit where date between '2020-12-01' and '2020-12-15';

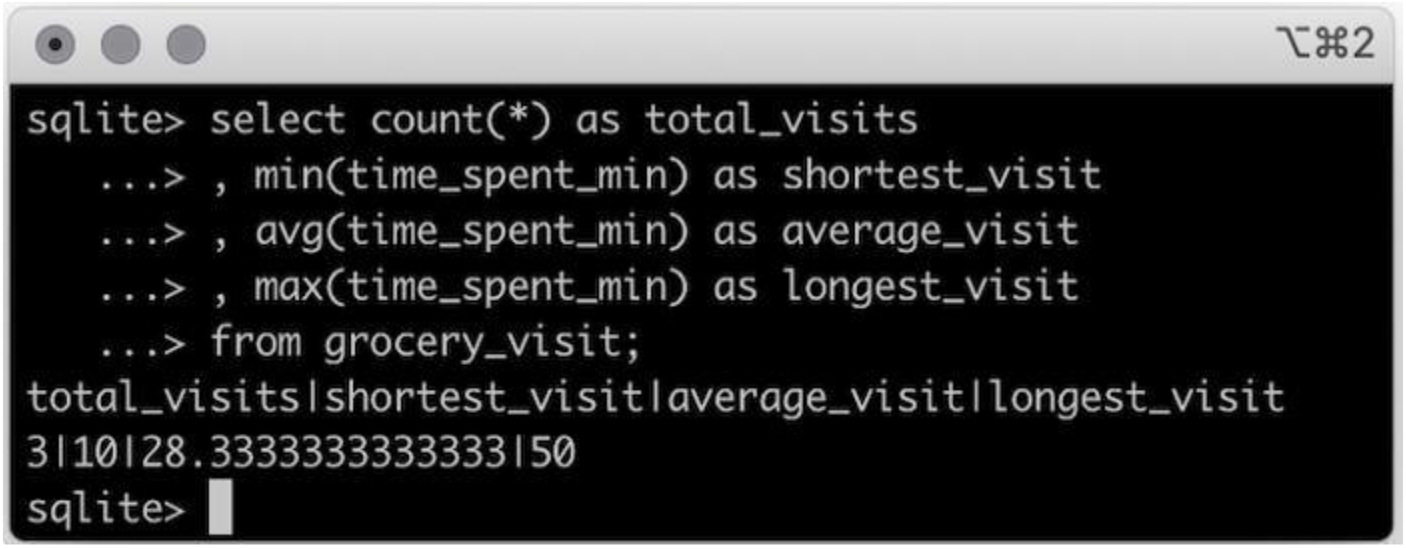


```
sqlite> select *
   ...> from grocery_visit
   ...> where date >= '2020-12-01'
   ...> and date <= '2020-12-15';
date|time_spent_min|amount_spent
2020-12-03|25|12.76
2020-12-15|10|7.88
sqlite>
```



```
sqlite> select *
   ...> from grocery_visit
   ...> where date between '2020-12-01' and '2020-12-15';
date|time_spent_min|amount_spent
2020-12-03|25|12.76
2020-12-15|10|7.88
sqlite>
```

select count(*) as total_visits
, min(time_spent_min) as shortest_visit
, avg(time_spent_min) as average_visit
, max(time_spent_min) as
longest_visitfrom grocery_visit;

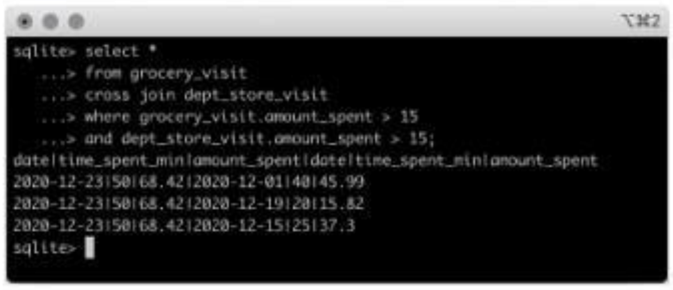


```
sqlite> select count(*) as total_visits
   ...> , min(time_spent_min) as shortest_visit
   ...> , avg(time_spent_min) as average_visit
   ...> , max(time_spent_min) as longest_visit
   ...> from grocery_visit;
total_visits|shortest_visit|average_visit|longest_visit
3|10|28.3333333333333|50
sqlite>
```

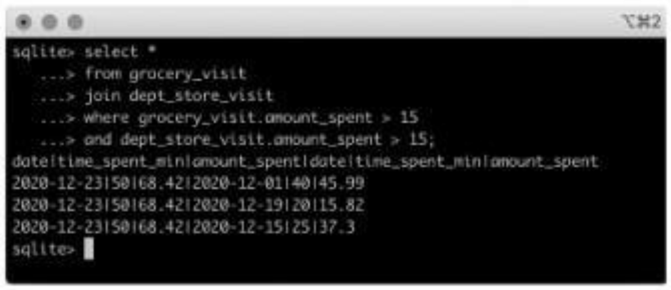
Cross join:

select *
from grocery_visit
cross join dept_store_visit

```
where grocery_visit.amount_spent >
and dept_store_visit.amount_spent > 15;
```



```
sqlite> select *
   ...> from grocery_visit
   ...> cross join dept_store_visit
   ...> where grocery_visit.amount_spent > 15
   ...> and dept_store_visit.amount_spent > 15;
date|time_spent_min|amount_spent|date|time_spent_min|amount_spent
2020-12-23|50|68.42|2020-12-01|40|45.99
2020-12-23|50|68.42|2020-12-19|20|15.82
2020-12-23|50|68.42|2020-12-15|25|37.3
sqlite>
```



```
sqlite> select *
   ...> from grocery_visit
   ...> join dept_store_visit
   ...> where grocery_visit.amount_spent > 15
   ...> and dept_store_visit.amount_spent > 15;
date|time_spent_min|amount_spent|date|time_spent_min|amount_spent
2020-12-23|50|68.42|2020-12-01|40|45.99
2020-12-23|50|68.42|2020-12-19|20|15.82
2020-12-23|50|68.42|2020-12-15|25|37.3
sqlite>
```

RESULT:

Thus the E-mart Grocery Shop is successfully designed.

1..What is DBMS and what is its utility? Explain RDBMS with examples.

2.What is a Database?

3.Mention the issues with traditional file-based systems that make DBMS a better choice?

4.Explain a few advantages of a DBMS.

5.Explain different languages present in DBMS.

6.What is meant by ACID properties in DBMS?

7.Are NULL values in a database the same as that of blank space or zero?

8.What is Data Warehousing?

9.Explain different levels of data abstraction in a DBMS.

10.What is meant by an entity-relationship (E-R) model? Explain the terms Entity, Entity Type, and Entity Set in DBMS.

11.Explain different types of relationships amongst tables in a DBMS.

12.Explain the difference between intension and extension in a database.

13.Explain the difference between the DELETE and TRUNCATE command in a DBMS.

14.What is a lock. Explain the major difference between a shared lock and an exclusive lock during a transaction in a database.

16.Explain different types of Normalization forms in a DBMS.

17.Explain different types of keys in a database.

18.Explain the difference between a 2-tier and 3-tier architecture in a DBMS.