CSLR 51 : DBMS-LAB

LAB RECORD

Roll no. : **106119100**

Name : **Rajneesh Pandey**

Section : **CSE-B**

Ex. No.: 01 Date: 13-08-21 Week: 1

Problem 1. Consider the Following Database:

A software company wants to track project details

Employee(Empid , Empname, Address, Doj, Salary) : Empid as Primary key

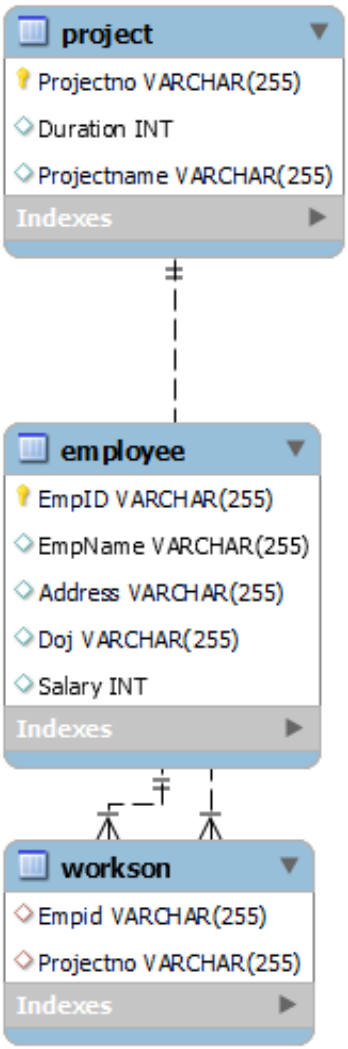
Project (Projectno, Duration, Projectname) : Project no as Primary Key

Workson(Empid,Projno) :

Empid as Foreign key references Employee

Projectno as Foreign key references Project

ER Diagram




```
CREATE DATABASE companyDB;
```

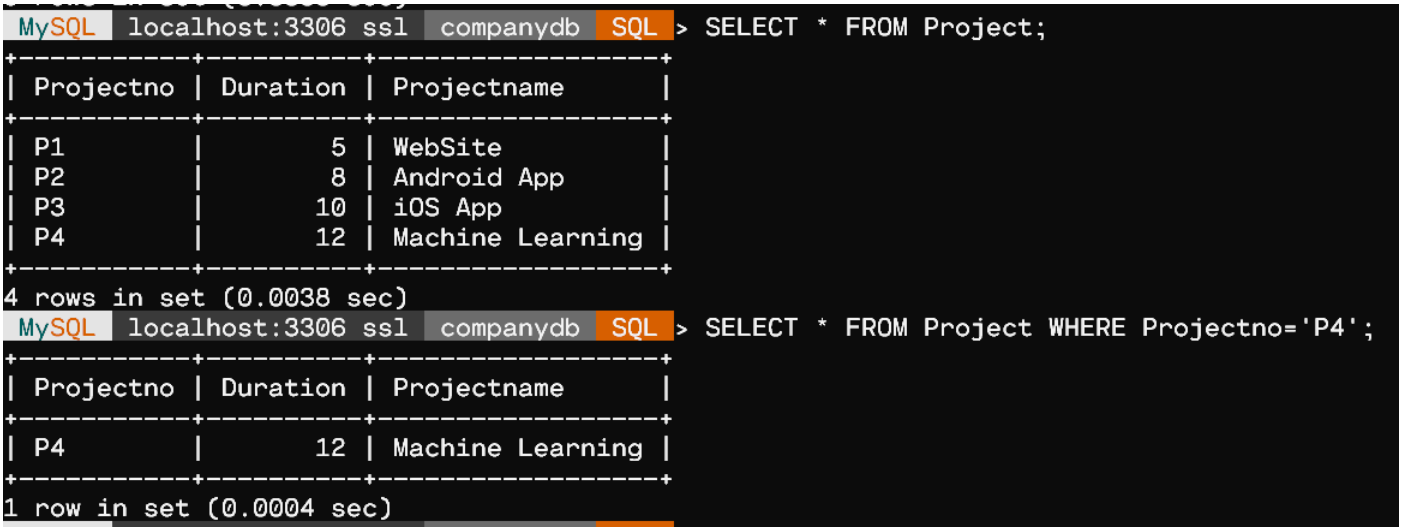
1. Display the Employee details in the descending order based on name.

```sql
CREATE TABLE Employee(
  EmpID varchar(255) NOT NULL,
  EmpName varchar(255),
  Address varchar(255),
  Doj varchar(255),
  Salary int,
  UNIQUE (EmpID),
  PRIMARY KEY (EmpID)
  );

INSERT INTO Employee(EmpID, EmpName, Address, Doj, Salary)
VALUES
 ('001','Harry','3427 Hall Valley Drive','12/28/1984',59000),
 ('002','Wilson','3950 Rinehart Road','5/18/1983',42000),
 ('003','Mildred','4768 Scenicview Drive','6/21/1969',78000),
 ('004','Anderson','78 Heritage Road','10/21/1995',49000),
 ('005','Bob','856 Tenmile Road','10/13/1960',50000);

SELECT *
FROM Employee e
ORDER BY e.EmpName DESC;
```

```
MySQL localhost:3306 ssl SQL > use companyDB;
Default schema set to `companyDB`.
Fetching table and column names from `companydb` for auto-completion... Press ^C to stop.
MySQL localhost:3306 ssl companydb SQL > SELECT *
                                      -> FROM Employee e
                                      -> ORDER BY e.EmpName DESC;
+-------+----------+-------------------------+------------+--------+
| EmpID | EmpName  | Address                 | Doj        | Salary |
+-------+----------+-------------------------+------------+--------+
| 002   | Wilson   | 3950 Rinehart Road      | 5/18/1983  |  42000 |
| 003   | Mildred  | 4768 Scenicview Drive   | 6/21/1969  |  78000 |
| 001   | Harry    | 3427 Hall Valley Drive  | 12/28/1984 |  59000 |
| 005   | Bob      | 856 Tenmile Road        | 10/13/1960 |  50000 |
| 004   | Anderson | 78 Heritage Road        | 10/21/1995 |  49000 |
+-------+----------+-------------------------+------------+--------+
5 rows in set (0.0008 sec)
MySQL localhost:3306 ssl companydb SQL >
```

2. Display the project details if project id is given.

```
CREATE TABLE Project(
  Projectno varchar(255) NOT NULL,
  Duration int,
  Projectname varchar(255),
  UNIQUE (Projectno),
  PRIMARY KEY (Projectno)
  );

INSERT INTO Project(Projectno,Duration, Projectname)
VALUES
 ('P1',5,'WebSite'),
 ('P2',8,'Android App'),
 ('P3',10,'iOS App'),
 ('P4',12,'Machine Learning');

SELECT * FROM Project;
SELECT * FROM Project WHERE Projectno='P4';
```



```
MySQL localhost:3306 ssl companydb SQL > SELECT * FROM Project;
+-----------+----------+------------------+
| Projectno | Duration | Projectname      |
+-----------+----------+------------------+
| P1        |        5 | WebSite          |
| P2        |        8 | Android App      |
| P3        |       10 | iOS App          |
| P4        |       12 | Machine Learning |
+-----------+----------+------------------+
4 rows in set (0.0038 sec)
MySQL localhost:3306 ssl companydb SQL > SELECT * FROM Project WHERE Projectno='P4';
+-----------+----------+------------------+
| Projectno | Duration | Projectname      |
+-----------+----------+------------------+
| P4        |       12 | Machine Learning |
+-----------+----------+------------------+
1 row in set (0.0004 sec)
```

3. Display the employee names starting with 'B'

```
SELECT EmpName
from Employee
where EmpName LIKE 'B%';
```

```
MySQL  localhost:3306 ssl  companydb  SQL > SELECT EmpName FROM Employee WHERE EmpName LIKE 'B%';
+---------+
| EmpName |
+---------+
| Bob     |
+---------+
1 row in set (0.0004 sec)
```

4. Display the employee ID's working in a particular project if project no is given.

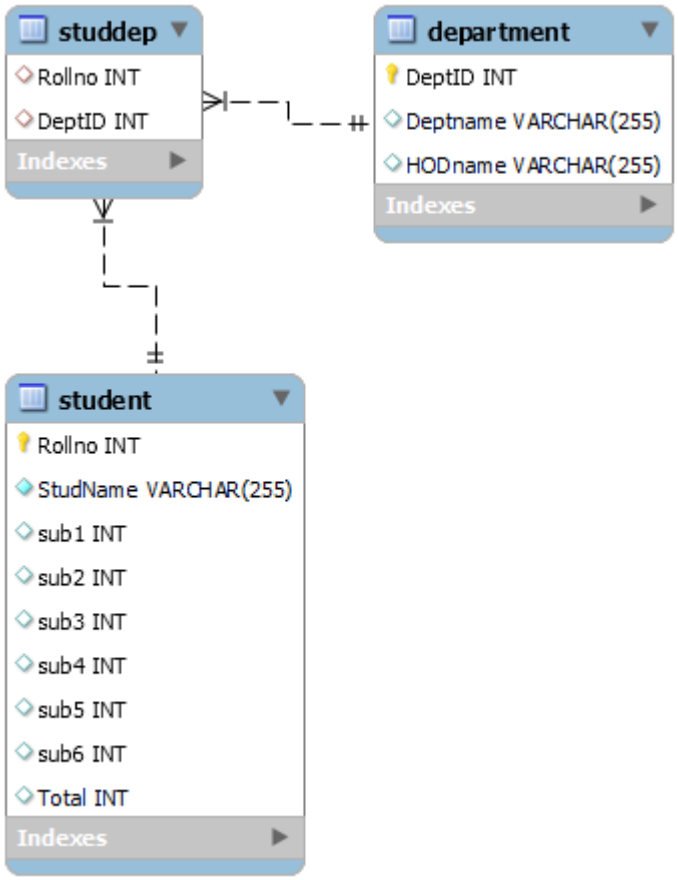
```
CREATE TABLE Workson(
  Empid varchar(255),
  Projectno  varchar(255),
  FOREIGN KEY (Empid) REFERENCES Employee(Empid),
  FOREIGN KEY (Projectno) REFERENCES Project(Projectno)
  );

INSERT INTO Workson(Projectno,Empid)
VALUES
  ('P3','002'),
  ('P2','004'),
  ('P1','003'),
  ('P4','001'),
  ('P2','005');

SELECT EmpID
FROM Workson
WHERE projectno = 'P2';
```

```
MySQL  localhost:3306 ssl  companydb  SQL > SELECT EmpID
                                         -> FROM Workson
                                         -> WHERE projectno = 'P2';
+--------+
| EmpID  |
+--------+
| 004    |
| 005    |
+--------+
2 rows in set (0.0018 sec)
```

Problem2. Consider the Following Database:

Student(Rollno, Name, Marks(of 6 subjects),total) : Rollno as Primary key
Department(Deptid, Deptname, HOD name) and Deptid as Primary key
StudDep(Rollno, Deptid).
Rollno as foreign key references Student
Deptid as foreign key references Department
The total field is updated automatically

ER Diagram:




```
CREATE DATABASE collegeDB;
```

```
CREATE TABLE Student(
                Rollno int NOT NULL,
                StudName varchar(255) NOT NULL,
                sub1 int,
                sub2 int,
                sub3 int,
                sub4 int,
                sub5 int,
                sub6 int,
                Total int,
                UNIQUE (Rollno),
                PRIMARY KEY(Rollno)
            );

            CREATE TABLE Department(
                DeptID int NOT NULL,
                Deptname varchar(255),
                HODname varchar(255),
                UNIQUE (DeptID),
                PRIMARY KEY (DeptID)
            );
            CREATE TABLE StudDep(
                Rollno int,
                DeptID int,
                FOREIGN KEY (Rollno) REFERENCES Student(Rollno),
                FOREIGN KEY (DeptID) REFERENCES Department(DeptID)
            );
```

1.Insert 10 student details and 3 department details. Insert details in the studdep table.

```
INSERT INTO Student(Rollno,StudName, sub1, sub2, sub3, sub4, sub5, sub6)
VALUES
 (1,'Amar',70,80,90,80,100,90),
 (2,'Shivam',80,90,80,100,90,70),
 (3,'Radha',80,100,90,70,70,40),
 (4,'Nitin',40,60,50,80,60,70),
 (5,'Ritik',50,50,60,100,90,70),
 (6,'Vaibhav',100,100,100,100,100,90),
 (7,'Kartik',10,4,50,80,100,90),
 (8,'Ram',80,90,80,100,90,70),
 (9,'Tom',90,80,100,90,50,90),
```

```
 (10,'Katy',90,80,100,90,100,100);

SET SQL_SAFE_UPDATES = 0;

UPDATE Student SET Total = sub1 + sub2 + sub3 + sub4 + sub5 + sub6;

INSERT INTO Department(DeptID,Deptname,HODname)
VALUES
 (1,'CSE','Dinesh'),
 (2,'Mech','Lakshmi'),
 (3,'ECE','Surya');

INSERT INTO StudDep(Rollno,DeptID)
VALUES
 (1,1),
 (2,1),
 (3,3),
 (4,2),
 (5,1),
 (6,2),
 (7,1),
 (8,3),
 (9,3),
 (10,2);
```

2.Display the Student details if deptid is given.

```
SELECT *
FROM Student
WHERE Rollno IN
(SELECT Rollno
FROM StudDep
WHERE DeptID=1
);
```

```
MySQL  localhost:3306 ssl  collegedb  SQL > SELECT *
                                        -> FROM Student
                                        -> WHERE Rollno IN
                                        -> (SELECT Rollno
                                        -> FROM StudDep
                                        -> WHERE DeptID=1
                                        -> );
+--------+----------+------+------+------+------+------+------+-------+
| Rollno | StudName | sub1 | sub2 | sub3 | sub4 | sub5 | sub6 | Total |
+--------+----------+------+------+------+------+------+------+-------+
|      1 | Amar     |   70 |   80 |   90 |   80 |  100 |   90 |   510 |
|      2 | Shivam   |   80 |   90 |   80 |  100 |   90 |   70 |   510 |
|      5 | Ritik    |   50 |   50 |   60 |  100 |   90 |   70 |   420 |
|      7 | Kartik   |   10 |    4 |   50 |   80 |  100 |   90 |   334 |
+--------+----------+------+------+------+------+------+------+-------+
4 rows in set (0.0015 sec)
```

3.Display the department details if rollno is given

```sql
SELECT *
FROM Department
WHERE DeptID IN
(SELECT DeptID
FROM StudDep
WHERE Rollno=5
);
```

```
MySQL  localhost:3306 ssl  collegedb  SQL > SELECT *
                                        -> FROM Department
                                        -> WHERE DeptID IN
                                        -> (SELECT DeptID
                                        -> FROM StudDep
                                        -> WHERE Rollno=5
                                        -> );
+--------+----------+---------+
| DeptID | Deptname | HODname |
+--------+----------+---------+
|      1 | CSE      | Dinesh  |
+--------+----------+---------+
1 row in set (0.0016 sec)
```

4.Display the student names who got total greater than 500

```
SELECT StudName
FROM Student
WHERE Total>500;
```

```
MySQL  localhost:3306 ssl  collegedb  SQL > SELECT StudName
                                         -> FROM Student
                                         -> WHERE Total>500;
+----------+
| StudName |
+----------+
| Amar     |
| Shivam   |
| Vaibhav  |
| Ram      |
| Katy     |
+----------+
5 rows in set (0.0006 sec)
```

5. Display the HOD name of the CSE department

```
SELECT HODName
FROM Department
WHERE Deptname='CSE';
```

```
MySQL  localhost:3306 ssl  collegedb  SQL > SELECT HODName
                                         -> FROM Department
                                         -> WHERE Deptname='CSE';
+---------+
| HODName |
+---------+
| Dinesh  |
+---------+
1 row in set (0.0005 sec)
```

6. Display the student rollnos of the CSE department

```
SELECT Rollno
```

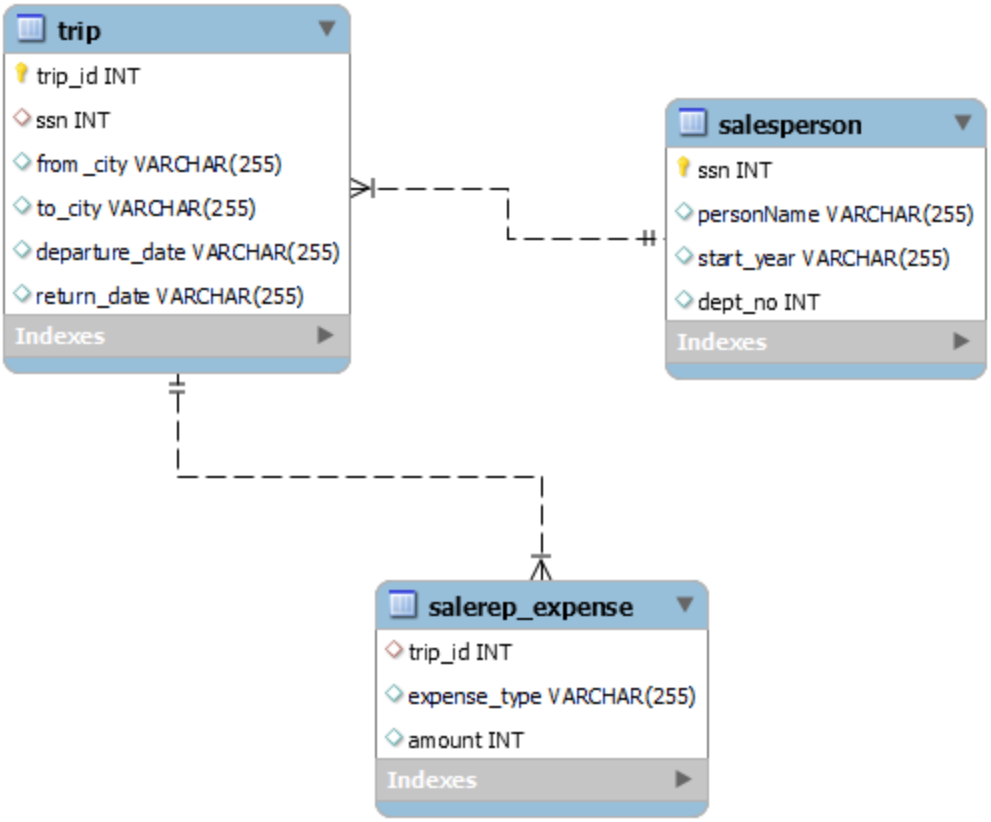
```
FROM StudDep
WHERE DeptID IN
(SELECT DeptID
FROM Department
WHERE Deptname='CSE'
);
```

```
MySQL  localhost:3306 ssl  collegedb  SQL > SELECT Rollno
                                         -> FROM StudDep
                                         -> WHERE DeptID IN
                                         -> (SELECT DeptID
                                         -> FROM Department
                                         -> WHERE Deptname='CSE'
                                         -> );
+--------+
| Rollno |
+--------+
|      1 |
|      2 |
|      5 |
|      7 |
+--------+
4 rows in set (0.0006 sec)
```

Problem 3.Consider the Following Database:

salesperson(ssn, name, start_year, dept_no)
ssn – Primary Key
trip(ssn, from_city, to_city, departure_date, return_date, trip_id))
ssn – Foreign key
trip_id – Primary key
salerep_expense(trip_id, expense_type,amount)
trip_id – Foreign key
The expense types are 'TRAVEL', 'STAY' and 'FOOD'

ER DIAGRAM




```
CREATE DATABASE vacationDB;
```

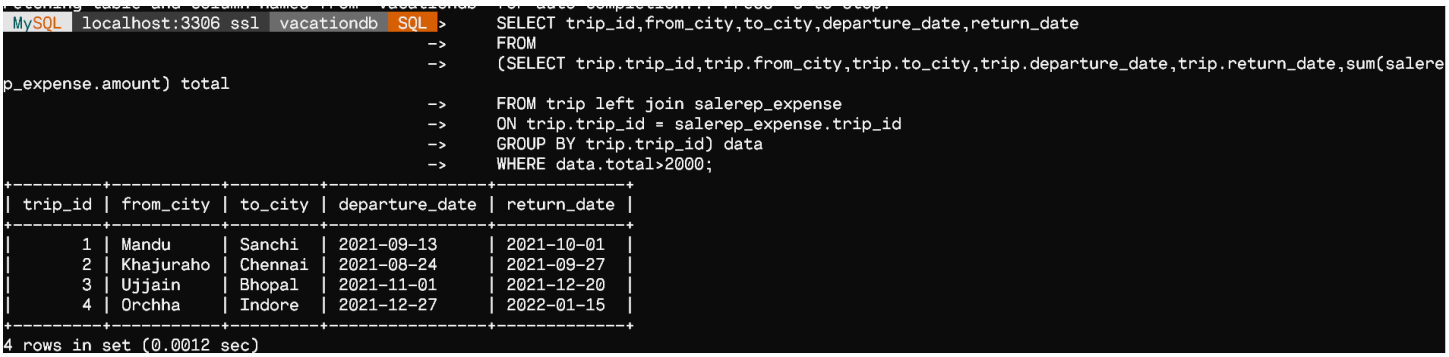
```
    CREATE TABLE salesperson(
      ssn int NOT NULL,
      personName varchar(255),
      start_year varchar(255),
      dept_no int,
      UNIQUE (ssn),
      PRIMARY KEY (ssn)
      );

    CREATE TABLE trip(
      trip_id int NOT NULL,
      ssn int,
      from_city varchar(255),
```

```
  to_city varchar(255),
  departure_date varchar(255),
  return_date varchar(255),
  UNIQUE (trip_id),
  PRIMARY KEY (trip_id),
  FOREIGN KEY (ssn) REFERENCES salesperson(ssn)
  );

CREATE TABLE salerep_expense(
  trip_id int,
  expense_type  varchar(255),
/* The expense types are 'TRAVEL', 'STAY' and 'FOOD' */
  amount int,
  FOREIGN KEY (trip_id) REFERENCES trip(trip_id)
  );


INSERT INTO salesperson(ssn,personName,start_year,dept_no)
VALUES
 (1000,'Harry','12/28/1984',100),
 (2000,'Wilson','5/18/1983',200),
 (3000,'Mildred','6/21/1969',300),
 (4000,'Anderson','10/21/1995',400),
 (5000,'Bob','10/13/1960',500);

INSERT INTO trip(trip_id,from_city,to_city,departure_date,return_date,ssn)
VALUES
 (1,'Mandu','Sanchi','2021-09-13','2021-10-01',2000),
 (2,'Khajuraho','Chennai','2021-08-24','2021-09-27',1000),
 (3,'Ujjain','Bhopal','2021-11-01','2021-12-20',5000),
 (4,'Orchha','Indore','2021-12-27','2022-01-15',3000),
 (5,'Pachmarhi','Chennai','2021-11-23','2021-12-20',4000),
 (6,'Jammu','Chennai','2021-11-20','2021-12-20',4000);

INSERT INTO salerep_expense(trip_id,expense_type,amount)
VALUES
 (1,'TRAVEL',2000),
 (2,'TRAVEL',1500),
 (3,'TRAVEL',2100),
 (4,'TRAVEL',1300),
 (5,'TRAVEL',1800),
```

```
(6,'TRAVEL',1000),
(1,'STAY',500),
(2,'STAY',600),
(4,'STAY',620),
(6,'STAY',600),
(1,'FOOD',800),
(2,'FOOD',100),
(3,'FOOD',700),
(4,'FOOD',600);
```

1.Give the details(all attributes of trip relation)for trips that exceed Rs2000

```
SELECT trip_id,from_city,to_city,departure_date,return_date
FROM
(SELECT trip.trip_id,trip.from_city,trip.to_city,trip.departure_date,trip.return_date,sum(salerep_expense.amount) total
FROM trip left join salerep_expense
ON trip.trip_id = salerep_expense.trip_id
GROUP BY trip.trip_id) data
WHERE data.total>2000;
```



```
MySQL  localhost:3306 ssl  vacationdb  SQL >    SELECT trip_id,from_city,to_city,departure_date,return_date
                                          ->    FROM
                                          ->    (SELECT trip.trip_id,trip.from_city,trip.to_city,trip.departure_date,trip.return_date,sum(salere
p_expense.amount) total
                                          ->    FROM trip left join salerep_expense
                                          ->    ON trip.trip_id = salerep_expense.trip_id
                                          ->    GROUP BY trip.trip_id) data
                                          ->    WHERE data.total>2000;
+---------+-----------+---------+----------------+-------------+
| trip_id | from_city | to_city | departure_date | return_date |
+---------+-----------+---------+----------------+-------------+
|       1 | Mandu     | Sanchi  | 2021-09-13     | 2021-10-01  |
|       2 | Khajuraho | Chennai | 2021-08-24     | 2021-09-27  |
|       3 | Ujjain    | Bhopal  | 2021-11-01     | 2021-12-20  |
|       4 | Orchha    | Indore  | 2021-12-27     | 2022-01-15  |
+---------+-----------+---------+----------------+-------------+
4 rows in set (0.0012 sec)
```

2.Print the ssn of salesperson who took trips to chennai more than once

```
SELECT ssn
FROM salesperson sp
```

```
    WHERE 1 <
        (SELECT COUNT(*)
         FROM trip
         WHERE ssn = sp.ssn AND to_city='Chennai');
```

```
MySQL  localhost:3306 ssl  vacationdb  SQL >     SELECT ssn
                                       ->     FROM salesperson sp
                                       ->     WHERE 1 <
                                       ->         (SELECT COUNT(*)
                                       ->          FROM trip
                                       ->          WHERE ssn = sp.ssn AND to_city='Chennai');
+------+
| ssn  |
+------+
| 4000 |
+------+
1 row in set (0.0011 sec)
```

3.Print the total trip expenses incurred by the salesperson with ssn = 1000

```
    SELECT sum(salerep_expense.amount) total
    FROM trip left join salerep_expense
    ON trip.trip_id = salerep_expense.trip_id
    GROUP BY trip.ssn = 1000;
```

```
MySQL  localhost:3306 ssl  vacationdb  SQL >     SELECT sum(salerep_expense.amount) total
                                       ->     FROM trip left join salerep_expense
                                       ->     ON trip.trip_id = salerep_expense.trip_id
                                       ->     GROUP BY trip.ssn = 1000;
+--------+
| total  |
+--------+
|  2200  |
```

4.Display the salesperson details in the sorted order based on name

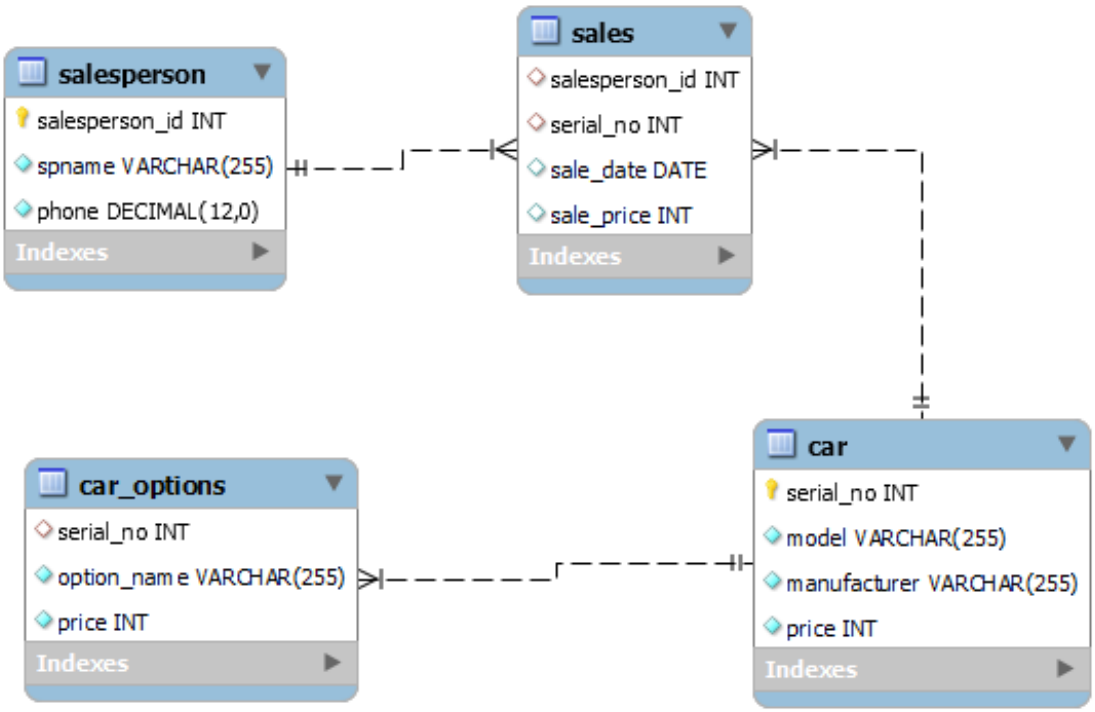
```
    SELECT *
    FROM salesperson sp
    ORDER BY sp.personName ASC;
```

```
MySQL  localhost:3306 ssl  vacationdb  SQL >    SELECT *
                                        ->    FROM salesperson sp
                                        ->    ORDER BY sp.personName ASC;
+------+------------+------------+---------+
| ssn  | personName | start_year | dept_no |
+------+------------+------------+---------+
| 4000 | Anderson   | 10/21/1995 |     400 |
| 5000 | Bob        | 10/13/1960 |     500 |
| 1000 | Harry      | 12/28/1984 |     100 |
| 3000 | Mildred    | 6/21/1969  |     300 |
| 2000 | Wilson     | 5/18/1983  |     200 |
+------+------------+------------+---------+
5 rows in set (0.0005 sec)
```

Problem 4. Consider the Following Database:

car(serial_no, model, manufacturer, price)
serial_no – Primary key
options(serial_no, option_name, price)
serial_no – Foreign key
sales(salesperson_id, serial_no, date, sale_price)
serial_no – Foreign key
salesperson_id – Foreign key
salesperson(salesperson_id, name, phone)
salesperson_id – Primary key

ER DIAGRAM



```
CREATE DATABASE carDB;
```

```
CREATE TABLE Car(
    serial_no int NOT NULL,
    model varchar(255) NOT NULL,
    manufacturer varchar(255) NOT NULL,
    price int NOT NULL,
    UNIQUE (serial_no),
    PRIMARY KEY(serial_no)
);
```

```
CREATE TABLE salesperson(
    salesperson_id int NOT NULL,
    spname varchar(255) NOT NULL,
    phone DECIMAL(12) NOT NULL,
    UNIQUE (salesperson_id),
    PRIMARY KEY(salesperson_id)
);

CREATE TABLE car_options(
    serial_no int,
    option_name varchar(255) NOT NULL,
    price int NOT NULL,
    FOREIGN KEY (serial_no) REFERENCES Car(serial_no)
);

CREATE TABLE sales(
    salesperson_id int,
    serial_no int,
    sale_date DATE,
    sale_price int,
    FOREIGN KEY (serial_no) REFERENCES Car(serial_no),
    FOREIGN KEY (salesperson_id) REFERENCES salesperson(salesperson_id)
);
```

```
INSERT INTO Car(serial_no,model,manufacturer,price)
VALUES
 (1,'Swift','Suzuki',700000),
 (2,'City','Honda',900000),
 (3,'Nano','Tata',500000),
 (4,'Fortuner','Toyota',1000000);

INSERT INTO salesperson(salesperson_id,spname,phone)
VALUES
 (1,'John',8168915356),
 (2,'Tom',9368570708),
 (3,'Martin',7895247308);

INSERT INTO car_options(serial_no,option_name,price)
VALUES
 (1,'Black',850000),
 (1,'Blue',760000),
 (1,'White',900000),
 (2,'Black',850000),
 (2,'Blue',760000),
 (4,'Blue',760000),
 (4,'White',900000);


INSERT INTO sales(salesperson_id,serial_no,sale_date,sale_price)
VALUES
 (2,1,'2021-01-15',850000),
 (1,2,'2021-04-07',750000),
 (3,3,'2021-03-23',600000),
 (1,4,'2021-03-12',900000);
```

1. For the sales person named 'John' list the following information for all the cars sold : serial no, manufacturer, sale_price

```
SELECT *
FROM Car
WHERE serial_no IN(
    SELECT serial_no
    FROM sales
    WHERE salesperson_id=
        (SELECT salesperson_id FROM salesperson WHERE spname='John')
```

```
);
```

```
MySQL localhost:3306 ssl vacationdb SQL > use carDB;
Default schema set to `carDB`.
Fetching table and column names from `cardb` for auto-completion... Press ^C to stop.
MySQL localhost:3306 ssl cardb SQL > SELECT *
                                   -> FROM Car
                                   -> WHERE serial_no IN(
                                   -> SELECT serial_no
                                   -> FROM sales
                                   -> WHERE salesperson_id=
                                   -> (SELECT salesperson_id FROM salesperson WHERE spname='John')
                                   -> );
+-----------+----------+--------------+---------+
| serial_no | model    | manufacturer | price   |
+-----------+----------+--------------+---------+
|         2 | City     | Honda        |  900000 |
|         4 | Fortuner | Toyota       | 1000000 |
+-----------+----------+--------------+---------+
2 rows in set (0.0012 sec)
```

2.List the serial_no and model of cars that have no options

```
SELECT serial_no,model
FROM
(SELECT Car.serial_no,Car.model,car_options.option_name
FROM Car left join car_options
ON Car.serial_no = car_options.serial_no
GROUP BY Car.serial_no) data
WHERE data.option_name IS NULL;
```

```
MySQL localhost:3306 ssl cardb SQL >     SELECT serial_no,model
                                   ->     FROM
                                   ->     (SELECT Car.serial_no,Car.model,car_options.option_name
                                   ->     FROM Car left join car_options
                                   ->     ON Car.serial_no = car_options.serial_no
                                   ->     GROUP BY Car.serial_no) data
                                   ->     WHERE data.option_name IS NULL;
+-----------+-------+
| serial_no | model |
+-----------+-------+
|         3 | Nano  |
+-----------+-------+
1 row in set (0.0009 sec)
```

3.List the serial_no, model, sale_price for the cars that have optional parts.

```
SELECT serial_no,model,SP
FROM
```

```
    (SELECT Car.serial_no,Car.model,car_options.option_name, (SELECT sale_price F
ROM sales WHERE serial_no = Car.serial_no) SP
    FROM Car left join car_options
    ON Car.serial_no = car_options.serial_no
    GROUP BY Car.serial_no) data
    WHERE data.option_name IS NOT NULL;
```

```
MySQL  localhost:3306 ssl  cardb  SQL >     SELECT serial_no,model,SP
                                    ->     FROM
                                    ->     (SELECT Car.serial_no,Car.model,car_options.option_name, (SELECT sale_price FROM sales WHERE serial_n
o = Car.serial_no) SP
                                    ->     FROM Car left join car_options
                                    ->     ON Car.serial_no = car_options.serial_no
                                    ->     GROUP BY Car.serial_no) data
                                    ->     WHERE data.option_name IS NOT NULL;
+-----------+----------+--------+
| serial_no | model    | SP     |
+-----------+----------+--------+
|         1 | Swift    | 850000 |
|         2 | City     | 750000 |
|         4 | Fortuner | 900000 |
+-----------+----------+--------+
3 rows in set (0.0012 sec)
```

4.Modify the phone no of a particular sales person

```
    UPDATE salesperson
    SET phone = 8941999954
    WHERE spname='Tom';
```

```
MySQL  localhost:3306 ssl  cardb  SQL > UPDATE salesperson
                                    -> SET phone = 8941999954
                                    -> WHERE spname='Tom';
Query OK, 0 rows affected (0.0005 sec)

Rows matched: 1  Changed: 0  Warnings: 0
```

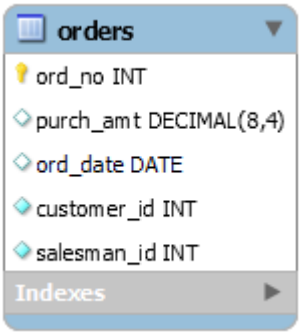
```
    SELECT * FROM salesperson;
```

```
MySQL  localhost:3306 ssl  cardb  SQL >     SELECT * FROM salesperson;
+----------------+---------+------------+
| salesperson_id | spname  | phone      |
+----------------+---------+------------+
|              1 | John    | 8168915356 |
|              2 | Tom     | 8941999954 |
|              3 | Martin  | 7895247308 |
+----------------+---------+------------+
3 rows in set (0.0005 sec)
```

PROBLEM 1

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006
70003	2480.4	2012-10-10	3009	5003
70012	250.45	2012-06-27	3008	5002
70011	75.29	2012-08-17	3003	5007
70013	3045.6	2012-04-25	3002	5001

ER Diagram :




```
CREATE DATABASE purchaseDB;


CREATE TABLE orders(
  ord_no int NOT NULL,
  purch_amt DECIMAL(8,4),
  ord_date DATE,
  customer_id int NOT NULL,
  salesman_id int NOT NULL,
  PRIMARY KEY(ord_no)
  );
```

```sql
INSERT INTO orders(ord_no,purch_amt,ord_date, customer_id,salesman_id)
VALUES
(70001, 150.5, '2012-10-05', 3005, 5002),
(70009, 270.65, '2012-09-10', 3001, 5005),
(70002, 65.26, '2012-10-05', 3002, 5001),
(70004, 110.5, '2012-08-17', 3009, 5003),
(70007, 948.5, '2012-09-10', 3005, 5002),
(70005, 2400.6, '2012-07-27', 3007, 5001),
(70008, 5760, '2012-09-10', 3002, 5001),
(70010, 1983.43, '2012-10-10', 3004, 5006),
(70003, 2480.4, '2012-10-10', 3009, 5003),
(70012, 250.45, '2012-06-27', 3008, 5002),
(70011, 75.29, '2012-08-17', 3003, 5007),
(70013, 3045.6, '2012-04-25', 3002, 5001);
```

```sql
/* Queries */

SELECT * FROM orders;
```

```
MySQL localhost:3306 ssl SQL > use purchaseDB
Default schema set to `purchaseDB`.
Fetching table and column names from `purchasedb` for auto-completion... Press ^C to stop.
MySQL localhost:3306 ssl purchasedb SQL > SELECT * FROM orders;
+--------+-----------+------------+-------------+-------------+
| ord_no | purch_amt | ord_date   | customer_id | salesman_id |
+--------+-----------+------------+-------------+-------------+
|  70001 |  150.5000 | 2012-10-05 |        3005 |        5002 |
|  70002 |   65.2600 | 2012-10-05 |        3002 |        5001 |
|  70003 | 2480.4000 | 2012-10-10 |        3009 |        5003 |
|  70004 |  110.5000 | 2012-08-17 |        3009 |        5003 |
|  70005 | 2400.6000 | 2012-07-27 |        3007 |        5001 |
|  70007 |  948.5000 | 2012-09-10 |        3005 |        5002 |
|  70008 | 5760.0000 | 2012-09-10 |        3002 |        5001 |
|  70009 |  270.6500 | 2012-09-10 |        3001 |        5005 |
|  70010 | 1983.4300 | 2012-10-10 |        3004 |        5006 |
|  70011 |   75.2900 | 2012-08-17 |        3003 |        5007 |
|  70012 |  250.4500 | 2012-06-27 |        3008 |        5002 |
|  70013 | 3045.6000 | 2012-04-25 |        3002 |        5001 |
+--------+-----------+------------+-------------+-------------+
12 rows in set (0.0030 sec)
```

```sql
/* 1. From the given table, Write a SQL statement to get the minimum
```

```
purchase amount of all the orders. */


SELECT MIN(purch_amt)
FROM orders;
```

```
MySQL  localhost:3306 ssl  purchasedb  SQL > SELECT MIN(purch_amt)
                                           -> FROM orders;
+----------------+
| MIN(purch_amt) |
+----------------+
|        65.2600 |
+----------------+
1 row in set (0.0017 sec)
```

```
/* 2. From the given table, Write a SQL statement to get the maximum
purchase amount of all the orders. */

SELECT MAX(purch_amt)
FROM orders;
```

```
MySQL  localhost:3306 ssl  purchasedb  SQL > SELECT MAX(purch_amt)
                                           -> FROM orders;
+----------------+
| MAX(purch_amt) |
+----------------+
|      5760.0000 |
+----------------+
1 row in set (0.0008 sec)
```

```
/* 3. From the given table, Write a SQL statement to get the total purchase
amount of all the orders.*/

SELECT SUM(purch_amt)
FROM orders;
```

```
MySQL  localhost:3306 ssl  purchasedb  SQL > SELECT SUM(purch_amt)
                                           -> FROM orders;
+----------------+
| SUM(purch_amt) |
+----------------+
|     17541.1800 |
+----------------+
1 row in set (0.0006 sec)
```

```sql
/* 4. From the given table,Write a SQL statement to get the average
purchase amount of all the orders. */

SELECT AVG(purch_amt)
FROM orders;
```

```
MySQL  localhost:3306 ssl  purchasedb  SQL > SELECT AVG(purch_amt)
                                           -> FROM orders;
+----------------+
| AVG(purch_amt) |
+----------------+
|  1461.76500000 |
+----------------+
1 row in set (0.0012 sec)
```

```sql
/* 5. From the given table, write a SQL query to count the number of
customers. Return number of customers. */

SELECT COUNT(*)
FROM orders;
```

```
MySQL  localhost:3306 ssl  purchasedb  SQL > SELECT COUNT(*)
                                           -> FROM orders;
+----------+
| COUNT(*) |
+----------+
|       12 |
+----------+
1 row in set (0.0015 sec)
```

```sql
/* 6. In the above table, add a new column “warranty” which is six months
from the date of order.*/

ALTER TABLE orders
ADD warranty DATE default (adddate(ord_date, interval 6 month));
```

```
MySQL  localhost:3306 ssl  purchasedb  SQL > ALTER TABLE orders
                                           -> ADD warranty DATE default (adddate(ord_date, interval 6 month));
Query OK, 12 rows affected (0.1864 sec)

Records: 12  Duplicates: 0  Warnings: 0
```

```sql
SELECT * FROM orders;
```

```
MySQL  localhost:3306 ssl  purchasedb  SQL > SELECT * FROM orders;
+--------+----------+------------+-------------+-------------+------------+
| ord_no | purch_amt | ord_date   | customer_id | salesman_id | warranty   |
+--------+----------+------------+-------------+-------------+------------+
|  70001 |  150.5000 | 2012-10-05 |        3005 |        5002 | 2013-04-05 |
|  70002 |   65.2600 | 2012-10-05 |        3002 |        5001 | 2013-04-05 |
|  70003 | 2480.4000 | 2012-10-10 |        3009 |        5003 | 2013-04-10 |
|  70004 |  110.5000 | 2012-08-17 |        3009 |        5003 | 2013-02-17 |
|  70005 | 2400.6000 | 2012-07-27 |        3007 |        5001 | 2013-01-27 |
|  70007 |  948.5000 | 2012-09-10 |        3005 |        5002 | 2013-03-10 |
|  70008 | 5760.0000 | 2012-09-10 |        3002 |        5001 | 2013-03-10 |
|  70009 |  270.6500 | 2012-09-10 |        3001 |        5005 | 2013-03-10 |
|  70010 | 1983.4300 | 2012-10-10 |        3004 |        5006 | 2013-04-10 |
|  70011 |   75.2900 | 2012-08-17 |        3003 |        5007 | 2013-02-17 |
|  70012 |  250.4500 | 2012-06-27 |        3008 |        5002 | 2012-12-27 |
|  70013 | 3045.6000 | 2012-04-25 |        3002 |        5001 | 2012-10-25 |
+--------+----------+------------+-------------+-------------+------------+
12 rows in set (0.0037 sec)
```

```sql
/* 7. From the given table, Find out the most recently purchased order and
least recently purchased order using the ord_date.*/

SELECT MIN(ord_date) AS LEAST_RECENTLY_USED, MAX(ord_date)
AS MOST_RECENTLY_USED from orders;
```

```
MySQL  localhost:3306 ssl  purchasedb  SQL > SELECT MIN(ord_date) AS LEAST_RECENTLY_USED, MAX(ord_date)
                                         -> AS MOST_RECENTLY_USED from orders;
+--------------------+--------------------+
| LEAST_RECENTLY_USED | MOST_RECENTLY_USED |
+--------------------+--------------------+
| 2012-04-25         | 2012-10-10         |
+--------------------+--------------------+
1 row in set (0.0005 sec)
```

```sql
/* 8. From the given table, find and display the next day of order.*/

SELECT ord_no,ADDDATE(ord_date,INTERVAL 1 DAY)
AS NEXT_DAY FROM orders;
```

```
MySQL  localhost:3306 ssl  purchasedb  SQL > SELECT ord_no,ADDDATE(ord_date,INTERVAL 1 DAY)
                                          -> AS NEXT_DAY FROM orders;
+--------+------------+
| ord_no | NEXT_DAY   |
+--------+------------+
|  70001 | 2012-10-06 |
|  70002 | 2012-10-06 |
|  70003 | 2012-10-11 |
|  70004 | 2012-08-18 |
|  70005 | 2012-07-28 |
|  70007 | 2012-09-11 |
|  70008 | 2012-09-11 |
|  70009 | 2012-09-11 |
|  70010 | 2012-10-11 |
|  70011 | 2012-08-18 |
|  70012 | 2012-06-28 |
|  70013 | 2012-04-26 |
+--------+------------+
12 rows in set (0.0005 sec)
```

```
/* 9. From the given table, print the smallest and largest integer just smaller
and larger than the purchase amount.*/

SELECT ord_no, CEIL(purch_amt) AS JUST_BIGGER FROM orders;
```

```
MySQL  localhost:3306 ssl  purchasedb  SQL > SELECT ord_no, CEIL(purch_amt) AS JUST_BIGGER FROM orders;
+--------+-------------+
| ord_no | JUST_BIGGER |
+--------+-------------+
|  70001 |         151 |
|  70002 |          66 |
|  70003 |        2481 |
|  70004 |         111 |
|  70005 |        2401 |
|  70007 |         949 |
|  70008 |        5760 |
|  70009 |         271 |
|  70010 |        1984 |
|  70011 |          76 |
|  70012 |         251 |
|  70013 |        3046 |
+--------+-------------+
12 rows in set (0.0006 sec)
```

```
SELECT ord_no, FLOOR(purch_amt) AS JUST_SMALLER FROM orders;
```

```
MySQL  localhost:3306 ssl  purchasedb  SQL > SELECT ord_no, FLOOR(purch_amt) AS JUST_SMALLER FROM orders;
+--------+--------------+
| ord_no | JUST_SMALLER |
+--------+--------------+
|  70001 |          150 |
|  70002 |           65 |
|  70003 |         2480 |
|  70004 |          110 |
|  70005 |         2400 |
|  70007 |          948 |
|  70008 |         5760 |
|  70009 |          270 |
|  70010 |         1983 |
|  70011 |           75 |
|  70012 |          250 |
|  70013 |         3045 |
+--------+--------------+
12 rows in set (0.0006 sec)
```

```
/*10. Explore the round function with various parameter settings on the
column purchase amount.*/
- The ROUND() function rounds a number to a specified number of decimal places.
SELECT ord_no,ROUND(purch_amt, 3) AS ROUNDED_UPTO_2_DECIMAL FROM orders;
```

```
MySQL  localhost:3306 ssl  purchasedb  SQL > SELECT ord_no,ROUND(purch_amt, 3) AS ROUNDED_UPTO_2_DECIMAL FROM orders;
+--------+-----------------------+
| ord_no | ROUNDED_UPTO_2_DECIMAL |
+--------+-----------------------+
|  70001 |                150.500 |
|  70002 |                 65.260 |
|  70003 |               2480.400 |
|  70004 |                110.500 |
|  70005 |               2400.600 |
|  70007 |                948.500 |
|  70008 |               5760.000 |
|  70009 |                270.650 |
|  70010 |               1983.430 |
|  70011 |                 75.290 |
|  70012 |                250.450 |
|  70013 |               3045.600 |
+--------+-----------------------+
12 rows in set (0.0005 sec)
```

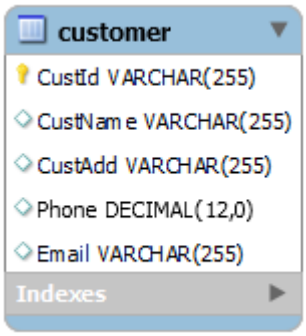
```
/* But If the 2nd Parameter is given Negative, then it rounds off upto Units Place,Tens.
SELECT ord_no,ROUND(purch_amt, -1) AS ROUNDED_UPTO_TENS FROM orders;
```

```
MySQL  localhost:3306 ssl  purchasedb  SQL > SELECT ord_no,ROUND(purch_amt, -1) AS ROUNDED_UPTO_TENS FROM orders;
+--------+-------------------+
| ord_no | ROUNDED_UPTO_TENS |
+--------+-------------------+
|  70001 |               150 |
|  70002 |                70 |
|  70003 |              2480 |
|  70004 |               110 |
|  70005 |              2400 |
|  70007 |               950 |
|  70008 |              5760 |
|  70009 |               270 |
|  70010 |              1980 |
|  70011 |                80 |
|  70012 |               250 |
|  70013 |              3050 |
+--------+-------------------+
12 rows in set (0.0005 sec)
```

PROBLEM 2

CustId	CustName	CustAdd	Phone	Email
C0001	AmitSaha	L-10, Pitampura	4564587852	amitsaha2@gmail.com
C0002	Rehnuma	J-12, SAKET	5527688761	rehnuma@hotmail.com
C0003	CharviNayyar	10/9, FF, Rohini	6811635425	charvi123@yahoo.com
C0004	Gurpreet	A-10/2, SF, MayurVihar	3511056125	gur_singh@yahoo.com

ER Diagram :




```
CREATE DATABASE customerDB;

CREATE TABLE customer(
CustId varchar(255) NOT NULL,
CustName varchar(255),
CustAdd varchar(255),
Phone DECIMAL(12),
Email varchar(255),
PRIMARY KEY(CustId)
);

INSERT INTO customer (CustId,CustName ,CustAdd ,Phone ,Email )
 VALUES
( 'C0001' ,'AmitSaha' , 'L-10, Pitampura' , 4564587852 ,'amitsaha2@gmail.com'),
( 'C0002' ,'Rehnuma', 'J-12, SAKET' ,5527688761 ,'rehnuma@hotmail.com'),
( 'C0003' ,'CharviNayyar', '10/9, FF, Rohini' , 6811635425 ,'charvi123@yahoo.com'
),
( 'C0004' ,'Gurpreet' , 'A-
10/2, SF, MayurVihar', 3511056125 ,'gur_singh@yahoo.com');
```

```
/* Queries */

SELECT * FROM customer;
```

```
MySQL  localhost:3306 ssl  purchasedb  SQL > use customerDB
Default schema set to `customerDB`.
Fetching table and column names from `customerdb` for auto-completion... Press ^C to stop.
MySQL  localhost:3306 ssl  customerdb  SQL > SELECT * FROM customer;
+--------+-------------+-----------------------+------------+--------------------+
| CustId | CustName    | CustAdd               | Phone      | Email              |
+--------+-------------+-----------------------+------------+--------------------+
| C0001  | AmitSaha    | L-10, Pitampura       | 4564587852 | amitsaha2@gmail.com |
| C0002  | Rehnuma     | J-12, SAKET           | 5527688761 | rehnuma@hotmail.com |
| C0003  | CharviNayyar | 10/9, FF, Rohini     | 6811635425 | charvi123@yahoo.com |
| C0004  | Gurpreet    | A-10/2, SF, MayurVihar | 3511056125 | gur_singh@yahoo.com |
+--------+-------------+-----------------------+------------+--------------------+
4 rows in set (0.0052 sec)
```

- 1. Display customer name in lower case and customer email in upper case from table CUSTOMER.

```
SELECT LOWER(CustName), UPPER(Email)
FROM customer;
```

```
MySQL  localhost:3306 ssl  customerdb  SQL > SELECT LOWER(CustName), UPPER(Email)
                                          -> FROM customer;
+-----------------+---------------------+
| LOWER(CustName) | UPPER(Email)        |
+-----------------+---------------------+
| amitsaha        | AMITSAHA2@GMAIL.COM |
| rehnuma         | REHNUMA@HOTMAIL.COM |
| charvinayyar    | CHARVI123@YAHOO.COM |
| gurpreet        | GUR_SINGH@YAHOO.COM |
+-----------------+---------------------+
4 rows in set (0.0006 sec)
```

- 2. Display emails after removing the domain name extension ".com" from emails of the customers.

```
SELECT TRIM('.com' from Email) 'Email without .com'
FROM customer;
```

```
MySQL  localhost:3306 ssl  customerdb  SQL > SELECT TRIM('.com' from Email) 'Email without .com'
                                          -> FROM customer;
+--------------------+
| Email without .com |
+--------------------+
| amitsaha2@gmail    |
| rehnuma@hotmail    |
| charvi123@yahoo    |
| gur_singh@yahoo    |
+--------------------+
4 rows in set (0.0007 sec)
```

```
-- 3. Display the length of the customer name.

SELECT CustName, LENGTH(CustName)  'Length of Names'
FROM customer;
```

```
MySQL  localhost:3306 ssl  customerdb  SQL > SELECT CustName, LENGTH(CustName)  'Length of Names'
                                           -> FROM customer;
+--------------+------------------+
| CustName     | Length of Names |
+--------------+------------------+
| AmitSaha     |                8 |
| Rehnuma      |                7 |
| CharviNayyar |               12 |
| Gurpreet     |                8 |
+--------------+------------------+
4 rows in set (0.0016 sec)
```

```
-
- 4. Display the custid and CustName joined together and the numeric position of
the letter A in the Customer name.

SELECT CONCAT(CustID,CustName) AS ID_AND_NAMES, LOCATE ('a',CustName) 'Position'
FROM customer;
```

```
MySQL  localhost:3306 ssl  customerdb  SQL > SELECT CONCAT(CustID,CustName) AS ID_AND_NAMES, LOCATE ('a',CustName) 'Position'
                                           -> FROM customer;
+------------------+----------+
| ID_AND_NAMES     | Position |
+------------------+----------+
| C0001AmitSaha    |        1 |
| C0002Rehnuma     |        7 |
| C0003CharviNayyar |        3 |
| C0004Gurpreet    |        0 |
+------------------+----------+
4 rows in set (0.0007 sec)
```

```
-
- 5. Write a SQL statement to display the data for those customers whose names en
d with ‘a’.

SELECT * FROM customer WHERE CustName LIKE '%a';
```

```
MySQL  localhost:3306 ssl  customerdb  SQL > SELECT * FROM customer WHERE CustName LIKE '%a';
+--------+----------+-----------------+------------+---------------------+
| CustId | CustName | CustAdd         | Phone      | Email               |
+--------+----------+-----------------+------------+---------------------+
| C0001  | AmitSaha | L-10, Pitampura | 4564587852 | amitsaha2@gmail.com |
| C0002  | Rehnuma  | J-12, SAKET     | 5527688761 | rehnuma@hotmail.com |
+--------+----------+-----------------+------------+---------------------+
2 rows in set (0.0006 sec)
```

```
- 6. Extract a substring from email starting at position 2 and 3 characters long.

SELECT SUBSTRING(Email,2,3) 'Substring 3 len'
FROM customer;
```

```
MySQL  localhost:3306 ssl  customerdb  SQL > SELECT SUBSTRING(Email,2,3) 'Substring 3 len'
                                          -> FROM customer;
+-----------------+
| Substring 3 len |
+-----------------+
| mit             |
| ehn             |
| har             |
| ur_             |
+-----------------+
4 rows in set (0.0005 sec)
```

```
-- 7. Extract 20 character string from Customer Address starting with position 1.

SELECT SUBSTRING(CustAdd,1,20) 'Address 20 chars'
FROM customer;
```

```
MySQL  localhost:3306 ssl  customerdb  SQL > SELECT SUBSTRING(CustAdd,1,20) 'Address 20 chars'
                                          -> FROM customer;
+----------------------+
| Address 20 chars     |
+----------------------+
| L-10, Pitampura      |
| J-12, SAKET          |
| 10/9, FF, Rohini     |
| A-10/2, SF, MayurVih |
+----------------------+
4 rows in set (0.0005 sec)
```

Ex. No.: 03 Date: 25-08-21 Week:3

PROBLEM 1

1) Given three tables, perform the following queries using joins:

Customer Table :

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007
3001	Brad Guzan	London	300	5005

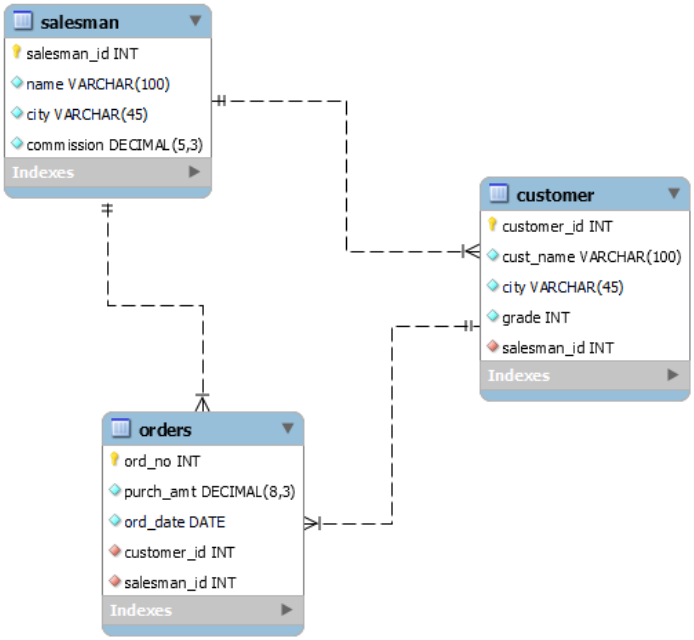
Salesman Table :

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
5003	Lauson Hen	San Jose	0.12

Orders Table :

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006
70003	2480.4	2012-10-10	3009	5003
70012	250.45	2012-06-27	3008	5002
70011	75.29	2012-08-17	3003	5007
70013	3045.6	2012-04-25	3002	5001

ER Diagram :



```sql
CREATE DATABASE salesDB;

CREATE TABLE salesman (
  `salesman_id` INT NOT NULL,
  `name` VARCHAR(100) NOT NULL,
  `city` VARCHAR(45) NOT NULL,
  `commission` DECIMAL(5,3) NOT NULL,
  PRIMARY KEY (`salesman_id`)
  );

CREATE TABLE customer (
  `customer_id` INT NOT NULL,
  `cust_name` VARCHAR(100) NOT NULL,
  `city` VARCHAR(45) NOT NULL,
  `grade` INT NOT NULL,
  `salesman_id` INT NOT NULL,
  PRIMARY KEY (`customer_id`),
  INDEX `salesman_id_idx` (`salesman_id` ASC) VISIBLE,
  CONSTRAINT `salesman_id`
    FOREIGN KEY (`salesman_id`)
    REFERENCES `salesDB`.`salesman` (`salesman_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    );

CREATE TABLE orders(
  `ord_no` INT NOT NULL,
  `purch_amt` DECIMAL(8,3) NOT NULL,
  `ord_date` DATE NOT NULL,
  `customer_id` INT NOT NULL,
  `salesman_id` INT NOT NULL,
    PRIMARY KEY (`ord_no`),
    FOREIGN KEY (`customer_id`)
    REFERENCES `salesDB`.`customer` (`customer_id`),
    FOREIGN KEY (`salesman_id`)
    REFERENCES `salesDB`.`salesman` (`salesman_id`)
    );
```

```
INSERT INTO salesman (`salesman_id`, `name`, `city`, `commission`)
VALUES
        ('5001', 'James Hoog', 'New York', '0.15'),
        ('5002', 'Nail Knite', 'Paris', '0.13'),
        ('5005', 'Pit Alex', 'London', '0.11'),
        ('5006', 'Mc Lyon', 'Paris', '0.14'),
        ('5007', 'Paul Adam', 'Rome', '0.13'),
        ('5003', 'Lauson Hen', 'San Jose', '0.12');

INSERT INTO customer (`customer_id`, `cust_name`, `city`, `grade`,
`salesman_id`)
VALUES
        ('3002', 'Nick Rimando', 'New York', '100', '5001'),
        ('3007', 'Brad Davis', 'New York', '200', '5001'),
        ('3005', 'Graham Zusi', 'California', '200', '5002'),
        ('3008', 'Julian Green', 'London', '300', '5002'),
        ('3004', 'Fabian Johnson', 'Paris', '300', '5006'),
        ('3009', 'Geoff Cameroon', 'Berlin', '100', '5003'),
        ('3003', 'Jozy Altidor', 'Moscow', '200', '5007'),
        ('3001', 'Brad Guzan', 'London', '200', '5005');


INSERT INTO orders (`ord_no`, `purch_amt`, `ord_date`, `customer_id
`, `salesman_id`)
VALUES
        ('70001', '150.5', '2012-10-05', '3005', '5002'),
        ('70009', '270.65', '2012-09-10', '3001', '5005'),
        ('70002', '65.26', '2012-10-05', '3002', '5001'),
        ('70004', '110.5', '2012-08-17', '3009', '5003'),
        ('70007', '948.5', '2012-09-10', '3005', '5002'),
        ('70005', '2400.6', '2012-07-27', '3007', '5001'),
        ('70008', '5760', '2012-09-10', '3002', '5001'),
        ('70010', '1983.43', '2012-10-10', '3004', '5006'),
        ('70003', '2480.4', '2012-10-10', '3009', '5003'),
        ('70012', '250.45', '2012-06-27', '3008', '5002'),
        ('70011', '75.29', '2012-08-17', '3003', '5007'),
        ('70013', '3045.6', '2012-04-25', '3002', '5001');
```

Query :-

Show Tables :

```
SELECT * FROM salesman;
```

```
MySQL  localhost:3306 ssl  SQL > use salesDB
Default schema set to `salesDB`.
Fetching table and column names from `salesdb` for auto-completion... Press ^C to stop.
MySQL  localhost:3306 ssl  salesdb  SQL >
MySQL  localhost:3306 ssl  salesdb  SQL > SELECT * FROM salesman;
+-------------+------------+----------+------------+
| salesman_id | name       | city     | commission |
+-------------+------------+----------+------------+
|        5001 | James Hoog | New York |      0.150 |
|        5002 | Nail Knite | Paris    |      0.130 |
|        5003 | Lauson Hen | San Jose |      0.120 |
|        5005 | Pit Alex   | London   |      0.110 |
|        5006 | Mc Lyon    | Paris    |      0.140 |
|        5007 | Paul Adam  | Rome     |      0.130 |
+-------------+------------+----------+------------+
6 rows in set (0.0020 sec)
```

```
SELECT * FROM customer;
```

```
MySQL  localhost:3306 ssl  salesdb  SQL > SELECT * FROM customer;
+-------------+---------------+------------+-------+-------------+
| customer_id | cust_name     | city       | grade | salesman_id |
+-------------+---------------+------------+-------+-------------+
|        3001 | Brad Guzan    | London     |   200 |        5005 |
|        3002 | Nick Rimando  | New York   |   100 |        5001 |
|        3003 | Jozy Altidor  | Moscow     |   200 |        5007 |
|        3004 | Fabian Johnson| Paris      |   300 |        5006 |
|        3005 | Graham Zusi   | California |   200 |        5002 |
|        3007 | Brad Davis    | New York   |   200 |        5001 |
|        3008 | Julian Green  | London     |   300 |        5002 |
|        3009 | Geoff Cameroon| Berlin     |   100 |        5003 |
+-------------+---------------+------------+-------+-------------+
8 rows in set (0.0123 sec)
```

```sql
SELECT * FROM orders;
```

```
MySQL  localhost:3306 ssl  salesdb  SQL > SELECT * FROM orders;
+--------+-----------+------------+-------------+-------------+
| ord_no | purch_amt | ord_date   | customer_id | salesman_id |
+--------+-----------+------------+-------------+-------------+
|  70001 |   150.500 | 2012-10-05 |        3005 |        5002 |
|  70002 |    65.260 | 2012-10-05 |        3002 |        5001 |
|  70003 |  2480.400 | 2012-10-10 |        3009 |        5003 |
|  70004 |   110.500 | 2012-08-17 |        3009 |        5003 |
|  70005 |  2400.600 | 2012-07-27 |        3007 |        5001 |
|  70007 |   948.500 | 2012-09-10 |        3005 |        5002 |
|  70008 |  5760.000 | 2012-09-10 |        3002 |        5001 |
|  70009 |   270.650 | 2012-09-10 |        3001 |        5005 |
|  70010 |  1983.430 | 2012-10-10 |        3004 |        5006 |
|  70011 |    75.290 | 2012-08-17 |        3003 |        5007 |
|  70012 |   250.450 | 2012-06-27 |        3008 |        5002 |
|  70013 |  3045.600 | 2012-04-25 |        3002 |        5001 |
+--------+-----------+------------+-------------+-------------+
12 rows in set (0.0089 sec)
```

a) Write a SQL query to find those salespersons who received a commission from the company more than 12%.Return Customer Name,customer city, Salesman, commission (Use Inner join)

```sql
SELECT c.cust_name AS "Customer Name", c.city, s.name AS "Salesman"
, s.commission
  FROM customer c
  INNER JOIN salesman s
    ON c.salesman_id = s.salesman_id
  WHERE s.commission > 0.12;
```

```
MySQL  localhost:3306 ssl  salesdb  SQL > SELECT c.cust_name AS "Customer Name", c.city, s.name AS "Salesman", s.commission
                                      ->   FROM customer c
                                      ->   INNER JOIN salesman s
                                      ->     ON c.salesman_id = s.salesman_id
                                      ->   WHERE s.commission > 0.12;
+---------------+-----------+-----------+------------+
| Customer Name | city      | Salesman  | commission |
+---------------+-----------+-----------+------------+
| Nick Rimando  | New York  | James Hoog |      0.150 |
| Brad Davis    | New York  | James Hoog |      0.150 |
| Graham Zusi   | California | Nail Knite |      0.130 |
| Julian Green  | London    | Nail Knite |      0.130 |
| Fabian Johnson | Paris     | Mc Lyon   |      0.140 |
| Jozy Altidor  | Moscow    | Paul Adam |      0.130 |
+---------------+-----------+-----------+------------+
6 rows in set (0.0017 sec)
```

b) Write a SQL statement to make a report with customer name, city, order number, order date,and order amount in ascending order according to the order date to find that either any of the existing customers have placed no order or placed one or more orders. (Use left outer join)

```
SELECT c.cust_name AS "Customer Name", c.city, o.ord_no, o.ord_date
, o.purch_amt
  FROM customer c
  LEFT OUTER JOIN orders o
    ON c.customer_id=o.customer_id
  ORDER BY o.ord_date;
```

```
MySQL  localhost:3306 ssl  salesdb  SQL > SELECT c.cust_name AS "Customer Name", c.city, o.ord_no, o.ord_date, o.purch_amt
                                      ->   FROM customer c
                                      ->   LEFT OUTER JOIN orders o
                                      ->     ON c.customer_id=o.customer_id
                                      ->   ORDER BY o.ord_date;
+---------------+------------+--------+------------+-----------+
| Customer Name | city       | ord_no | ord_date   | purch_amt |
+---------------+------------+--------+------------+-----------+
| Nick Rimando  | New York   |  70013 | 2012-04-25 |  3045.600 |
| Julian Green  | London     |  70012 | 2012-06-27 |   250.450 |
| Brad Davis    | New York   |  70005 | 2012-07-27 |  2400.600 |
| Jozy Altidor  | Moscow     |  70011 | 2012-08-17 |    75.290 |
| Geoff Cameroon | Berlin    |  70004 | 2012-08-17 |   110.500 |
| Brad Guzan    | London     |  70009 | 2012-09-10 |   270.650 |
| Nick Rimando  | New York   |  70008 | 2012-09-10 |  5760.000 |
| Graham Zusi   | California |  70007 | 2012-09-10 |   948.500 |
| Nick Rimando  | New York   |  70002 | 2012-10-05 |    65.260 |
| Graham Zusi   | California |  70001 | 2012-10-05 |   150.500 |
| Fabian Johnson | Paris     |  70010 | 2012-10-10 |  1983.430 |
| Geoff Cameroon | Berlin    |  70003 | 2012-10-10 |  2480.400 |
+---------------+------------+--------+------------+-----------+
12 rows in set (0.0099 sec)
```

c) Write a SQL statement to make a report with customer name, city, order number, order date, order amount salesman name and commission tofind that either any of the existing customers have placed no ord e or placed one or more orders by their salesman or by own.

```
SELECT c.cust_name AS "Customer Name",c.city,o.ord_no,o.ord_date,o.
purch_amt,s.name AS "Salesman",
       s.commission
  FROM customer c
  LEFT OUTER JOIN orders o
    ON c.customer_id=o.customer_id
  LEFT OUTER JOIN salesman s
    ON c.salesman_id=s.salesman_id;
```

```
MySQL  localhost:3306 ssl  salesdb  SQL > SELECT c.cust_name AS "Customer Name",c.city,o.ord_no,o.ord_date,o.purch_amt,s.name AS "Salesman",
                                      ->        s.commission
                                      ->   FROM customer c
                                      ->   LEFT OUTER JOIN orders o
                                      ->     ON c.customer_id=o.customer_id
                                      ->   LEFT OUTER JOIN salesman s
                                      ->     ON c.salesman_id=s.salesman_id;
+---------------+------------+--------+------------+-----------+------------+------------+
| Customer Name | city       | ord_no | ord_date   | purch_amt | Salesman   | commission |
+---------------+------------+--------+------------+-----------+------------+------------+
| Brad Guzan    | London     |  70009 | 2012-09-10 |   270.650 | Pit Alex   |      0.110 |
| Nick Rimando  | New York   |  70002 | 2012-10-05 |    65.260 | James Hoog |      0.150 |
| Nick Rimando  | New York   |  70008 | 2012-09-10 |  5760.000 | James Hoog |      0.150 |
| Nick Rimando  | New York   |  70013 | 2012-04-25 |  3045.600 | James Hoog |      0.150 |
| Jozy Altidor  | Moscow     |  70011 | 2012-08-17 |    75.290 | Paul Adam  |      0.130 |
| Fabian Johnson| Paris      |  70010 | 2012-10-10 |  1983.430 | Mc Lyon    |      0.140 |
| Graham Zusi   | California |  70001 | 2012-10-05 |   150.500 | Nail Knite |      0.130 |
| Graham Zusi   | California |  70007 | 2012-09-10 |   948.500 | Nail Knite |      0.130 |
| Brad Davis    | New York   |  70005 | 2012-07-27 |  2400.600 | James Hoog |      0.150 |
| Julian Green  | London     |  70012 | 2012-06-27 |   250.450 | Nail Knite |      0.130 |
| Geoff Cameroon| Berlin     |  70003 | 2012-10-10 |  2480.400 | Lauson Hen |      0.120 |
| Geoff Cameroon| Berlin     |  70004 | 2012-08-17 |   110.500 | Lauson Hen |      0.120 |
+---------------+------------+--------+------------+-----------+------------+------------+
12 rows in set (0.0006 sec)
```

d) Write a SQL statement to make a list in ascending order for the salesmenwho works either for one or more customer or not yet join u nder and of the customers. (Use Right outer join)

```sql
SELECT s.name AS "Salesman"
  FROM salesman s
  RIGHT OUTER JOIN customer c
    ON s.salesman_id=c.salesman_id
  ORDER BY c.salesman_id ASC;
```

```
MySQL  localhost:3306 ssl  salesdb  SQL > SELECT s.name AS "Salesman"
                                      ->   FROM salesman s
                                      ->   LEFT OUTER JOIN customer c
                                      ->     ON s.salesman_id=c.salesman_id
                                      ->   ORDER BY c.salesman_id ASC;
+-------------+
| Salesman    |
+-------------+
| James Hoog  |
| James Hoog  |
| Nail Knite  |
| Nail Knite  |
| Lauson Hen  |
| Pit Alex    |
| Mc Lyon     |
| Paul Adam   |
+-------------+
8 rows in set (0.0006 sec)
```

e) Write a SQL query to combine each row of salesman table with eac
h row of customer table. (Use cross join)

```
SELECT *
     FROM salesman a
     CROSS JOIN customer b;
```

```
MySQL  localhost:3306 ssl  salesdb  SQL > SELECT *
                                      ->     FROM salesman a
                                      ->     CROSS JOIN customer b;
+-------------+------------+----------+------------+-------------+---------------+------------+-------+-------------+
| salesman_id | name       | city     | commission | customer_id | cust_name     | city       | grade | salesman_id |
+-------------+------------+----------+------------+-------------+---------------+------------+-------+-------------+
|        5007 | Paul Adam  | Rome     |      0.130 |        3001 | Brad Guzan    | London     |   200 |        5005 |
|        5006 | Mc Lyon    | Paris    |      0.140 |        3001 | Brad Guzan    | London     |   200 |        5005 |
|        5005 | Pit Alex   | London   |      0.110 |        3001 | Brad Guzan    | London     |   200 |        5005 |
|        5003 | Lauson Hen | San Jose |      0.120 |        3001 | Brad Guzan    | London     |   200 |        5005 |
|        5002 | Nail Knite | Paris    |      0.130 |        3001 | Brad Guzan    | London     |   200 |        5005 |
|        5001 | James Hoog | New York |      0.150 |        3001 | Brad Guzan    | London     |   200 |        5005 |
|        5007 | Paul Adam  | Rome     |      0.130 |        3002 | Nick Rimando  | New York   |   100 |        5001 |
|        5006 | Mc Lyon    | Paris    |      0.140 |        3002 | Nick Rimando  | New York   |   100 |        5001 |
|        5005 | Pit Alex   | London   |      0.110 |        3002 | Nick Rimando  | New York   |   100 |        5001 |
|        5003 | Lauson Hen | San Jose |      0.120 |        3002 | Nick Rimando  | New York   |   100 |        5001 |
|        5002 | Nail Knite | Paris    |      0.130 |        3002 | Nick Rimando  | New York   |   100 |        5001 |
|        5001 | James Hoog | New York |      0.150 |        3002 | Nick Rimando  | New York   |   100 |        5001 |
|        5007 | Paul Adam  | Rome     |      0.130 |        3003 | Jozy Altidor  | Moscow     |   200 |        5007 |
|        5006 | Mc Lyon    | Paris    |      0.140 |        3003 | Jozy Altidor  | Moscow     |   200 |        5007 |
|        5005 | Pit Alex   | London   |      0.110 |        3003 | Jozy Altidor  | Moscow     |   200 |        5007 |
|        5003 | Lauson Hen | San Jose |      0.120 |        3003 | Jozy Altidor  | Moscow     |   200 |        5007 |
|        5002 | Nail Knite | Paris    |      0.130 |        3003 | Jozy Altidor  | Moscow     |   200 |        5007 |
|        5001 | James Hoog | New York |      0.150 |        3003 | Jozy Altidor  | Moscow     |   200 |        5007 |
|        5007 | Paul Adam  | Rome     |      0.130 |        3004 | Fabian Johnson | Paris     |   300 |        5006 |
|        5006 | Mc Lyon    | Paris    |      0.140 |        3004 | Fabian Johnson | Paris     |   300 |        5006 |
|        5005 | Pit Alex   | London   |      0.110 |        3004 | Fabian Johnson | Paris     |   300 |        5006 |
|        5003 | Lauson Hen | San Jose |      0.120 |        3004 | Fabian Johnson | Paris     |   300 |        5006 |
|        5002 | Nail Knite | Paris    |      0.130 |        3004 | Fabian Johnson | Paris     |   300 |        5006 |
|        5001 | James Hoog | New York |      0.150 |        3004 | Fabian Johnson | Paris     |   300 |        5006 |
|        5007 | Paul Adam  | Rome     |      0.130 |        3005 | Graham Zusi   | California |   200 |        5002 |
|        5006 | Mc Lyon    | Paris    |      0.140 |        3005 | Graham Zusi   | California |   200 |        5002 |
|        5005 | Pit Alex   | London   |      0.110 |        3005 | Graham Zusi   | California |   200 |        5002 |
|        5003 | Lauson Hen | San Jose |      0.120 |        3005 | Graham Zusi   | California |   200 |        5002 |
|        5002 | Nail Knite | Paris    |      0.130 |        3005 | Graham Zusi   | California |   200 |        5002 |
|        5001 | James Hoog | New York |      0.150 |        3005 | Graham Zusi   | California |   200 |        5002 |
|        5007 | Paul Adam  | Rome     |      0.130 |        3007 | Brad Davis    | New York   |   200 |        5001 |
|        5006 | Mc Lyon    | Paris    |      0.140 |        3007 | Brad Davis    | New York   |   200 |        5001 |
|        5005 | Pit Alex   | London   |      0.110 |        3007 | Brad Davis    | New York   |   200 |        5001 |
|        5003 | Lauson Hen | San Jose |      0.120 |        3007 | Brad Davis    | New York   |   200 |        5001 |
|        5002 | Nail Knite | Paris    |      0.130 |        3007 | Brad Davis    | New York   |   200 |        5001 |
|        5001 | James Hoog | New York |      0.150 |        3007 | Brad Davis    | New York   |   200 |        5001 |
|        5007 | Paul Adam  | Rome     |      0.130 |        3008 | Julian Green  | London     |   300 |        5002 |
|        5006 | Mc Lyon    | Paris    |      0.140 |        3008 | Julian Green  | London     |   300 |        5002 |
|        5005 | Pit Alex   | London   |      0.110 |        3008 | Julian Green  | London     |   300 |        5002 |
|        5003 | Lauson Hen | San Jose |      0.120 |        3008 | Julian Green  | London     |   300 |        5002 |
|        5002 | Nail Knite | Paris    |      0.130 |        3008 | Julian Green  | London     |   300 |        5002 |
|        5001 | James Hoog | New York |      0.150 |        3008 | Julian Green  | London     |   300 |        5002 |
|        5007 | Paul Adam  | Rome     |      0.130 |        3009 | Geoff Cameroon | Berlin    |   100 |        5003 |
|        5006 | Mc Lyon    | Paris    |      0.140 |        3009 | Geoff Cameroon | Berlin    |   100 |        5003 |
|        5005 | Pit Alex   | London   |      0.110 |        3009 | Geoff Cameroon | Berlin    |   100 |        5003 |
|        5003 | Lauson Hen | San Jose |      0.120 |        3009 | Geoff Cameroon | Berlin    |   100 |        5003 |
|        5002 | Nail Knite | Paris    |      0.130 |        3009 | Geoff Cameroon | Berlin    |   100 |        5003 |
|        5001 | James Hoog | New York |      0.150 |        3009 | Geoff Cameroon | Berlin    |   100 |        5003 |
+-------------+------------+----------+------------+-------------+---------------+------------+-------+-------------+
48 rows in set (0.0014 sec)
```

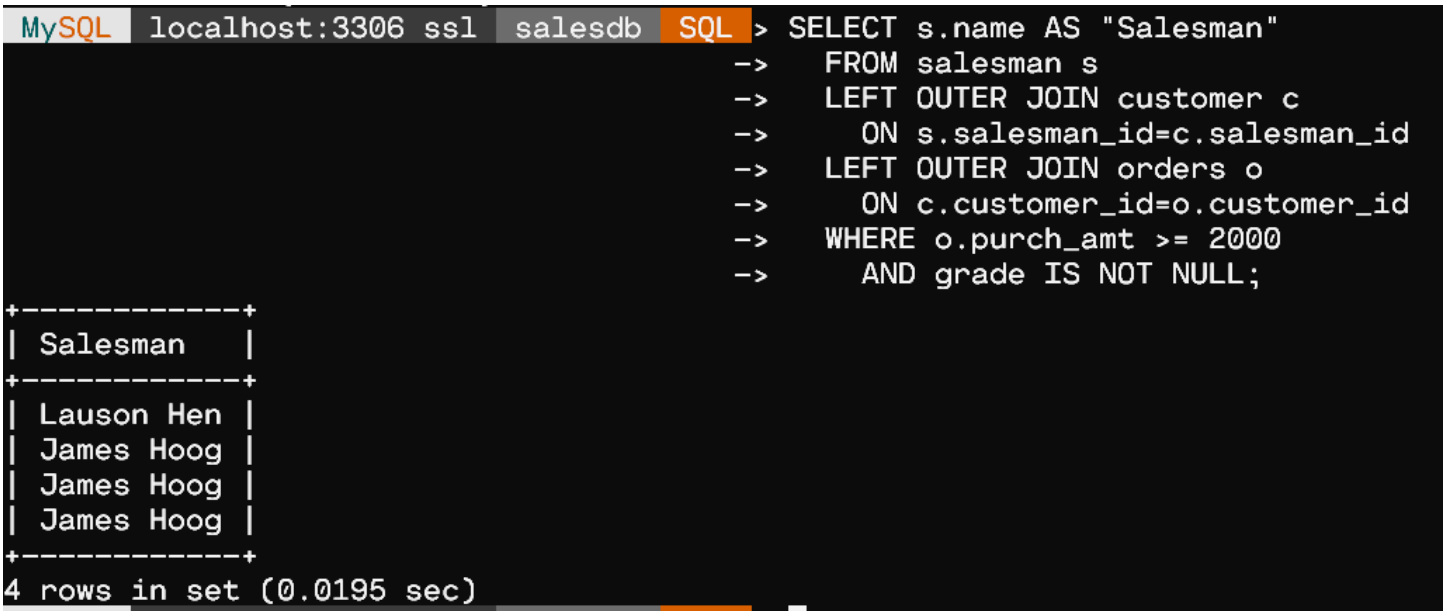
f) Write a SQL statement to make a cartesian product between salesm
an and customer i.e.each salesman will appear for all customer and
vice versa for that salesman who belongs to a city.(Use cross join)

```
SELECT s.name AS "Salesman", c.cust_name AS "Customer Name"
    FROM salesman s
    CROSS JOIN customer c;
```

```
MySQL  localhost:3306 ssl  salesdb  SQL > SELECT s.name AS "Salesman", c.cust_name AS "Customer Name"
                                      ->     FROM salesman s
                                      ->     CROSS JOIN customer c;
+------------+----------------+
| Salesman   | Customer Name  |
+------------+----------------+
| Paul Adam  | Brad Guzan     |
| Mc Lyon    | Brad Guzan     |
| Pit Alex   | Brad Guzan     |
| Lauson Hen | Brad Guzan     |
| Nail Knite | Brad Guzan     |
| James Hoog | Brad Guzan     |
| Paul Adam  | Nick Rimando   |
| Mc Lyon    | Nick Rimando   |
| Pit Alex   | Nick Rimando   |
| Lauson Hen | Nick Rimando   |
| Nail Knite | Nick Rimando   |
| James Hoog | Nick Rimando   |
| Paul Adam  | Jozy Altidor   |
| Mc Lyon    | Jozy Altidor   |
| Pit Alex   | Jozy Altidor   |
| Lauson Hen | Jozy Altidor   |
| Nail Knite | Jozy Altidor   |
| James Hoog | Jozy Altidor   |
| Paul Adam  | Fabian Johnson |
| Mc Lyon    | Fabian Johnson |
| Pit Alex   | Fabian Johnson |
| Lauson Hen | Fabian Johnson |
| Nail Knite | Fabian Johnson |
| James Hoog | Fabian Johnson |
| Paul Adam  | Graham Zusi    |
| Mc Lyon    | Graham Zusi    |
| Pit Alex   | Graham Zusi    |
| Lauson Hen | Graham Zusi    |
| Nail Knite | Graham Zusi    |
| James Hoog | Graham Zusi    |
| Paul Adam  | Brad Davis     |
| Mc Lyon    | Brad Davis     |
| Pit Alex   | Brad Davis     |
| Lauson Hen | Brad Davis     |
| Nail Knite | Brad Davis     |
| James Hoog | Brad Davis     |
| Paul Adam  | Julian Green   |
| Mc Lyon    | Julian Green   |
| Pit Alex   | Julian Green   |
| Lauson Hen | Julian Green   |
| Nail Knite | Julian Green   |
| James Hoog | Julian Green   |
| Paul Adam  | Geoff Cameroon |
| Mc Lyon    | Geoff Cameroon |
| Pit Alex   | Geoff Cameroon |
| Lauson Hen | Geoff Cameroon |
| Nail Knite | Geoff Cameroon |
| James Hoog | Geoff Cameroon |
+------------+----------------+
48 rows in set (0.0008 sec)
```

g) Write a SQL statement to make a list for the salesmen who either work for one or more customers or yet to join any of the customer. The customer may have placed, either one or more orders on or above order amount 2000 and must have a grade, or he may not have placed any order to the associated supplier.
(Use left outer join and right outer join)

```
SELECT s.name AS "Salesman"
  FROM salesman s
  LEFT OUTER JOIN customer c
    ON s.salesman_id=c.salesman_id
  RIGHT OUTER JOIN orders o
    ON c.customer_id=o.customer_id
  WHERE o.purch_amt >= 2000
    AND grade IS NOT NULL;
```

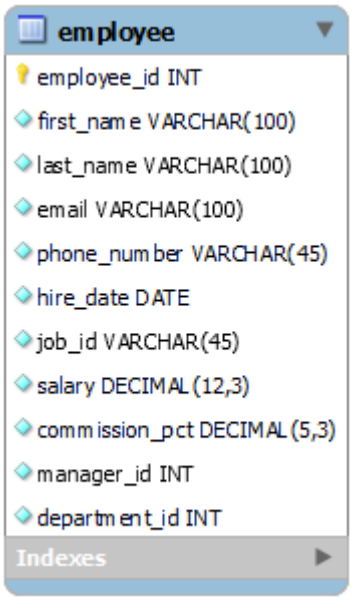


PROBLEM 2

2) Given the Employee table, perform the following statements using nested queries :

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	2003-06-17	AD_PRES	24000.00	0.00	0	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	2005-09-21	AD_VP	17000.00	0.00	100	40
102	Lex	De Haan	LDEHAAN	515.123.4569	2001-01-13	AD_VP	17000.00	0.00	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	2006-01-03	IT_PROG	9000.00	0.00	102	60
104	Bruce	Ernst	BERNST	590.423.4568	2007-05-21	IT_PROG	6000.00	0.00	103	60
105	David	Austin	DAUSTIN	590.423.4569	2005-06-25	IT_PROG	4800.00	0.00	103	40
106	Valli	Pataballa	VPATABAL	590.423.4560	2006-02-05	IT_PROG	4800.00	0.00	103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	2007-02-07	IT_PROG	4200.00	0.00	103	60
108	Nancy	Greenberg	NGREENBE	515.124.4569	2002-08-17	FI_MGR	12008.00	0.00	101	100
109	Daniel	Faviet	DFAVIET	515.124.4169	2002-08-16	FI_ACCOUNT	9000.00	0.00	108	100
110	John	Chen	JCHEN	515.124.4269	2005-09-28	FI_ACCOUNT	8200.00	0.00	108	40

ER Diagram :




```
CREATE DATABASE employeeDB;

CREATE TABLE `employee`(
  `employee_id` INT NOT NULL,
  `first_name` VARCHAR(100) NOT NULL,
  `last_name` VARCHAR(100) NOT NULL,
  `email` VARCHAR(100) NOT NULL,
  `phone_number` VARCHAR(45) NOT NULL,
  `hire_date` DATE NOT NULL,
  `job_id` VARCHAR(45) NOT NULL,
```

```
  `salary` DECIMAL(12,3) NOT NULL,
  `commission_pct` DECIMAL(5,3) NOT NULL,
  `manager_id` INT NOT NULL,
  `department_id` INT NOT NULL,
  PRIMARY KEY (`employee_id`)
  );

INSERT INTO employee
(`employee_id`, `first_name`, `last_name`, `email`, `phone_number`,
 `hire_date`, `job_id`, `salary`, `commission_pct`, `manager_id`, `
department_id`)
VALUES
       ('100', 'Steven', 'King', 'SKING', '515.123.4567', '2003-
06-17', 'AD_PRES', 24000.00, '0.00', '0', '90'),
       ('101', 'Neena', 'Kochhar', 'NKOCHHAR ', '515.123.4568 ', '
 2005-09-21 ', 'AD_VP', 17000.00, '0.00', '100', '40'),
       ('102', 'Lex', 'De Haan', 'LDEHAAN', '515.123.4569', '2001-
01-13 ', 'AD_VP', 17000.00, '0.00', '100', '90'),
       ('103', 'Alexander', 'Hunold', 'AHUNOLD', '590.423.4567', '
2006-01-03', 'IT_PROG', 9000.00, '0.00', '102', '60'),
       ('104', 'Bruce', 'Ernst', 'BERNST', '590.423.4568 ', '2007-
05-21', 'IT_PROG', 6000.00, '0.00', '103', '60'),
       ('105', 'David', 'Austin', 'DAUSTIN', '590.423.4569', '2005
-06-25', 'IT_PROG', 4800.00, '0.00', '103', '40'),
       ('106', 'Valli', 'Pataballa', 'VPATABAL', '590.423.4560', '
2006-02-05', 'IT_PROG', 4800.00, '0.00', '103', '60'),
       ('107', 'Diana', 'Lorentz', 'DLORENTZ', '590.423.5567', '20
07-02-07', 'IT_PROG', 4200.00, '0.00', '103', '60'),
       ('108', 'Nancy', 'Greenberg', 'NGREENBE', '515.124.4569', '
2002-08-17', 'FI_MGR', 12008.00, '0.00', '101', '100'),
       ('109', 'Daniel', 'Faviet', 'DFAVIET', '515.124.4169', '200
2-08-16', 'FI_ACCOUNT', 9000.00, '0.00', '108', '100'),
       ('110', 'John', 'Chen', 'JCHEN', '515.124.4269', '2005-09-
28', 'FI_ACCOUNT', 8200.00, '0.00', '108', '40');
```

```
-- Query :-
-- Show Tables :
    SELECT * FROM employee;
```

```
MySQL  localhost:3306 ssl  employeedb  SQL > SELECT * FROM employee;
+-------------+------------+-----------+----------+--------------+------------+------------+----------+----------------+------------+-----------+
| employee_id | first_name | last_name | email    | phone_number | hire_date  | job_id     | salary   | commission_pct | manager_id | departmen
t_id |
+-------------+------------+-----------+----------+--------------+------------+------------+----------+----------------+------------+-----------+
|         100 | Steven     | King      | SKING    | 515.123.4567 | 2003-06-17 | AD_PRES    | 24000.000 |          0.000 |          0 |
  90 |
|         101 | Neena      | Kochhar   | NKOCHHAR | 515.123.4568 | 2005-09-21 | AD_VP      | 17000.000 |          0.000 |        100 |
  40 |
|         102 | Lex        | De Haan   | LDEHAAN  | 515.123.4569 | 2001-01-13 | AD_VP      | 17000.000 |          0.000 |        100 |
  90 |
|         103 | Alexander  | Hunold    | AHUNOLD  | 590.423.4567 | 2006-01-03 | IT_PROG    |  9000.000 |          0.000 |        102 |
  60 |
|         104 | Bruce      | Ernst     | BERNST   | 590.423.4568 | 2007-05-21 | IT_PROG    |  6000.000 |          0.000 |        103 |
  60 |
|         105 | David      | Austin    | DAUSTIN  | 590.423.4569 | 2005-06-25 | IT_PROG    |  4800.000 |          0.000 |        103 |
  40 |
|         106 | Valli      | Pataballa | VPATABAL | 590.423.4560 | 2006-02-05 | IT_PROG    |  4800.000 |          0.000 |        103 |
  60 |
|         107 | Diana      | Lorentz   | DLORENTZ | 590.423.5567 | 2007-02-07 | IT_PROG    |  4200.000 |          0.000 |        103 |
  60 |
|         108 | Nancy      | Greenberg | NGREENBE | 515.124.4569 | 2002-08-17 | FI_MGR     | 12008.000 |          0.000 |        101 |
 100 |
|         109 | Daniel     | Faviet    | DFAVIET  | 515.124.4169 | 2002-08-16 | FI_ACCOUNT |  9000.000 |          0.000 |        108 |
 100 |
|         110 | John       | Chen      | JCHEN    | 515.124.4269 | 2005-09-28 | FI_ACCOUNT |  8200.000 |          0.000 |        108 |
  40 |
+-------------+------------+-----------+----------+--------------+------------+------------+----------+----------------+------------+-----------+
11 rows in set (0.0008 sec)
```

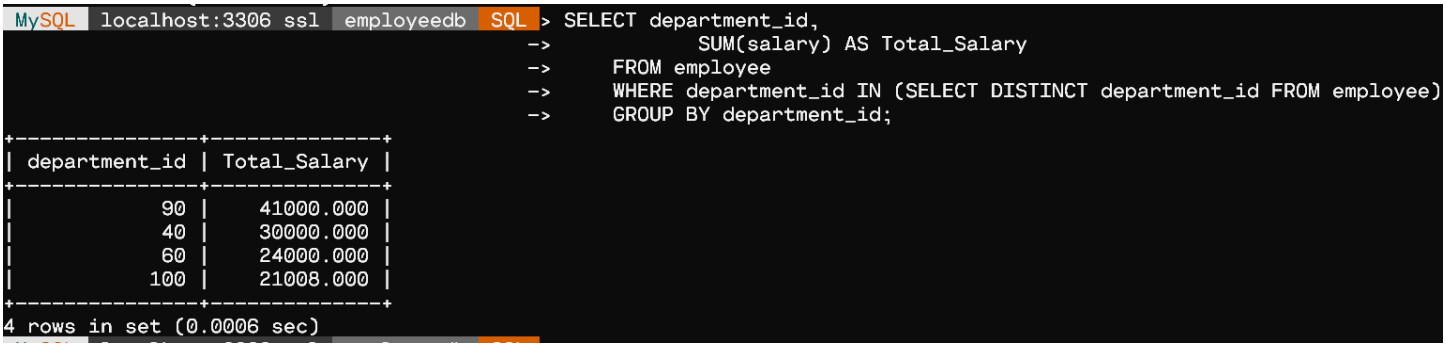
a) Write a query to display the employee name (first name and last name) and department for all employees for any existence of those employees whose salary is more than 3700.

```
    SELECT first_name, last_name, department_id
        FROM employee
        WHERE EXISTS (SELECT *
              FROM employee
              WHERE salary > 3700.00);
```

```
MySQL  localhost:3306 ssl  employeedb  SQL > SELECT first_name, last_name, department_id
                                          ->         FROM employee
                                          ->         WHERE EXISTS (SELECT *
                                          ->               FROM employee
                                          ->               WHERE salary > 3700.00);
+------------+-----------+---------------+
| first_name | last_name | department_id |
+------------+-----------+---------------+
| Steven     | King      |            90 |
| Neena      | Kochhar   |            40 |
| Lex        | De Haan   |            90 |
| Alexander  | Hunold    |            60 |
| Bruce      | Ernst     |            60 |
| David      | Austin    |            40 |
| Valli      | Pataballa |            60 |
| Diana      | Lorentz   |            60 |
| Nancy      | Greenberg |           100 |
| Daniel     | Faviet    |           100 |
| John       | Chen      |            40 |
+------------+-----------+---------------+
11 rows in set (0.0019 sec)
```

b) Write a query to display the department id and the total salary for those departments which contains at least one employee.

```
    SELECT department_id,
           SUM(salary) AS Total_Salary
    FROM employee
    WHERE department_id IN (SELECT DISTINCT department_id FROM empl
oyee)
    GROUP BY department_id;
```



c) Write a subquery that returns a set of rows to find all departments that do actually have one or more employees assigned to them.

```
    SELECT DISTINCT department_id FROM employee;
```

d) Write a query in SQL to display the first and last name, salary, and department ID for all those employees who earn more than the average salary and arrange the list in descending order on salary.

```sql
SELECT first_name, last_name, salary, department_id
  FROM employee
  WHERE salary > (SELECT AVG(salary)
            FROM employee)
  ORDER BY salary DESC;
```

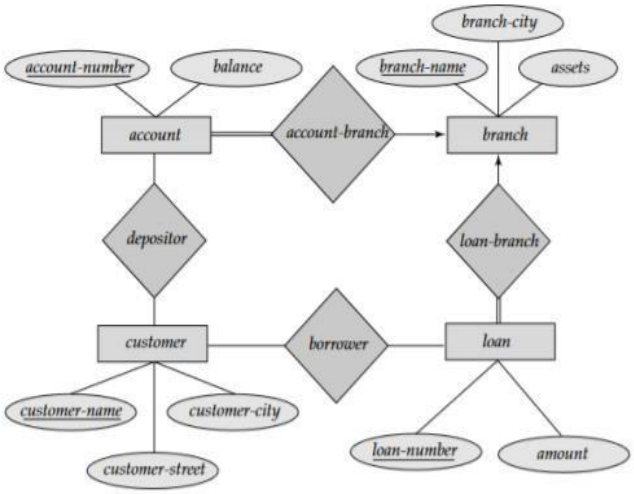
```
MySQL  localhost:3306 ssl  employeedb  SQL >    SELECT first_name, last_name, salary, department_id
                                           ->      FROM employee
                                           ->      WHERE salary > (SELECT AVG(salary)
                                           ->                FROM employee)
                                           ->      ORDER BY salary DESC;
+------------+-----------+-----------+---------------+
| first_name | last_name | salary    | department_id |
+------------+-----------+-----------+---------------+
| Steven     | King      | 24000.000 |            90 |
| Neena      | Kochhar   | 17000.000 |            40 |
| Lex        | De Haan   | 17000.000 |            90 |
| Nancy      | Greenberg | 12008.000 |           100 |
+------------+-----------+-----------+---------------+
4 rows in set (0.0006 sec)
```

e) Write a query in SQL to display the first and last name, salary, and department ID for those employees who earn more than the maximum salary of a department which ID is 40.

```sql
SELECT first_name, last_name, salary, department_id
    FROM employee
    WHERE salary > (SELECT MAX(salary)
              FROM employee
                  WHERE department_id = 40);
```

```
MySQL  localhost:3306 ssl  employeedb  SQL >      SELECT first_name, last_name, salary, department_id
                                           ->          FROM employee
                                           ->          WHERE salary > (SELECT MAX(salary)
                                           ->                    FROM employee
                                           ->                        WHERE department_id = 40);
+------------+-----------+-----------+---------------+
| first_name | last_name | salary    | department_id |
+------------+-----------+-----------+---------------+
| Steven     | King      | 24000.000 |            90 |
+------------+-----------+-----------+---------------+
1 row in set (0.0008 sec)
```

PROBLEM 1



branch-name	branch-city	assets
Brighton	Brooklyn	7100000
Downtown	Brooklyn	9000000
Mianus	Horseneck	400000
North Town	Rye	3700000
Perryridge	Horseneck	1700000
Pownal	Bennington	300000
Redwood	Palo Alto	2100000
Round Hill	Horseneck	8000000

customer-name	customer-street	customer-city
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford

customer-name	account-number
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

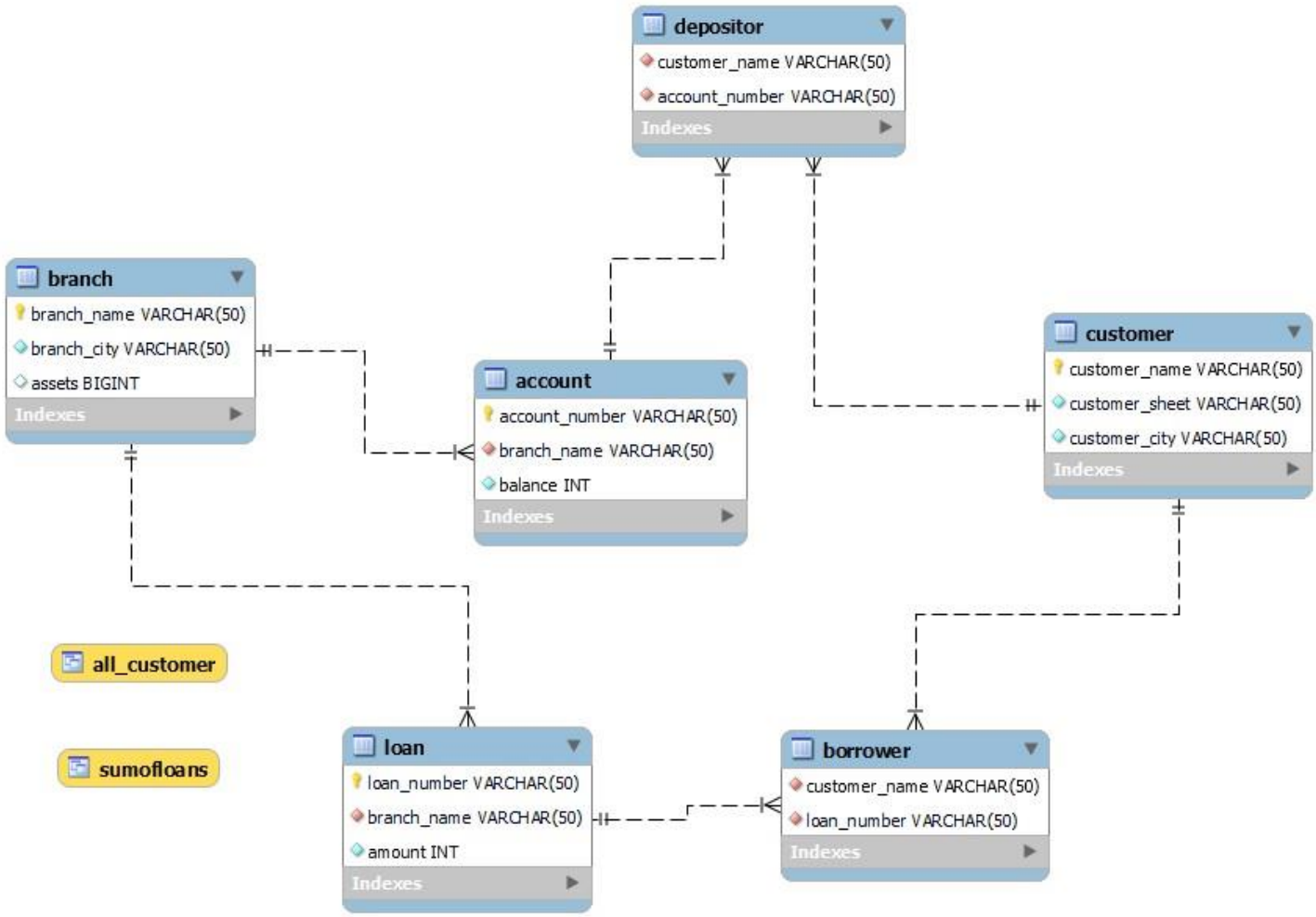
account-number	branch-name	balance
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

A) **Given Above tables, perform the following queries**

1. Create a view consisting of branch names and the names of customers who have either an account or a loan at that branch. Assume that view to be called *all-customer*.
2. Create a view gives for each branch the sum of the amounts of all the loans at the branch.
3. Using the view all-customer, we can find *all customers* of the *Perryridge* branch.
4. Write a Query for below Relational algebraic notation :

$$\Pi_{customer\text{-}name}(borrower) \cup \Pi_{customer\text{-}name}(depositor)$$

ER Diagram :




```
CREATE DATABASE bankDB;
USE bankDB;

CREATE TABLE branch
(     branch_name varchar(50) PRIMARY
KEY,      branch_city varchar(50) NOT
NULL,      assets bigint
);


CREATE TABLE account
```

```
(     account_number varchar(50) PRIMARY KEY,
branch_name varchar(50) NOT NULL,     balance INT NOT
NULL,     foreign key(branch_name) references
branch(branch_name)
);

CREATE TABLE loan
(     loan_number varchar(50) PRIMARY KEY,     branch_name
varchar(50) NOT NULL,     amount INT NOT NULL,     foreign
key(branch_name) references branch(branch_name)
);

CREATE TABLE customer
(     customer_name varchar(50) PRIMARY
KEY,     customer_sheet varchar(50) NOT
NULL,     customer_city varchar(50) NOT
NULL
);

CREATE TABLE depositor
(     customer_name varchar(50) NOT NULL,     account_number
varchar(50) NOT NULL,     foreign key(customer_name) references
customer(customer_name),     foreign key(account_number)
references account(account_number)
);

CREATE TABLE borrower
(     customer_name varchar(50) NOT NULL,
loan_number varchar(50) NOT NULL,     foreign
key(customer_name) references
customer(customer_name),     foreign key(loan_number)
references loan(loan_number) );  insert into branch
values
('Brighton','Brooklyn',7100000),
('Downtown','Brooklyn',9000000),
('Mianus','Horseneck',400000),
('North Town','Rye',3700000),
('Perryridge','Horseneck',1700000),
('Pownal','Bennington',300000),
('Redwood','Palo Alto',2100000),
```

```
('Round Hill','Horseneck',8000000);
 insert into
account values
('A-101','Downtown',500),
('A-102','Perryridge',400),
('A-201','Brighton',900), ('A-
215','Mianus',700),
('A-217','Brighton',750),
('A-222','Redwood',700),
('A-305','Round Hill',350);
 insert into
loan  values
('L-11','Round Hill',900),
('L-14','Downtown',1500),
('L-15','Perryridge',1500),
('L-16','Perryridge',1300),
('L-17','Downtown',1000),
('L-23','Redwood',2000),
('L-93','Mianus',500);
 insert into
customer  values
('Adams','Spring','Pittsfield'),
('Brooks','Senator','Brooklyn'),
('Curry','North','Rye'),
('Glenn','Sand Hill','Woodside'),
('Green','Walnut','Stamford'),
('Hayes','Main','Harrison'),
('Johnson','Alma','Palo Alto'),
('Jones','Main','Harrison'),
('Lindsay','Park','Pittsfield'),
('Smith','North','Rye'),
('Turner','Putnam','Stamford'),
('Williams','Nassau','Princeton');
 insert into
depositor values
('Hayes','A-102'),
('Johnson','A-101'),
('Johnson','A-201'), ('Jones','A-
217'),
```

```
('Lindsay','A-222'),
('Smith','A-215'),
('Turner','A-305');
 insert into
borrower values
('Adams','L-16'),
('Curry','L-93'),
('Hayes','L-15'),
('Johnson','L-14'),
('Jones','L-17'),
('Smith','L-11'),
('Smith','L-23'),
('Williams','L-17');

/*----------------------------- Querys------------------------------ */
```

customer;

```
MySQL localhost:3306 ssl bankdb SQL > SELECT * FROM customer;
+---------------+---------------+---------------+
| customer_name | customer_sheet | customer_city |
+---------------+---------------+---------------+
| Adams         | Spring         | Pittsfield    |
| Brooks        | Senator        | Brooklyn      |
| Curry         | North          | Rye           |
| Glenn         | Sand Hill      | Woodside      |
| Green         | Walnut         | Stamford      |
| Hayes         | Main           | Harrison      |
| Johnson       | Alma           | Palo Alto     |
| Jones         | Main           | Harrison      |
| Lindsay       | Park           | Pittsfield    |
| Smith         | North          | Rye           |
| Turner        | Putnam         | Stamford      |
| Williams      | Nassau         | Princeton     |
+---------------+---------------+---------------+
12 rows in set (0.0018 sec)
```

```
SELECT * FROM depositor;
```

```
MySQL localhost:3306 ssl bankdb SQL > SELECT * FROM depositor;
+---------------+----------------+
| customer_name | account_number |
+---------------+----------------+
| Hayes         | A-102          |
| Johnson       | A-101          |
| Johnson       | A-201          |
| Jones         | A-217          |
| Lindsay       | A-222          |
| Smith         | A-215          |
| Turner        | A-305          |
+---------------+----------------+
7 rows in set (0.0094 sec)
```

branch;

```
MySQL  localhost:3306 ssl  bankdb  SQL > SELECT * FROM branch;
+-------------+-------------+---------+
| branch_name | branch_city | assets  |
+-------------+-------------+---------+
| Brighton    | Brooklyn    | 7100000 |
| Downtown    | Brooklyn    | 9000000 |
| Mianus      | Horseneck   |  400000 |
| North Town  | Rye         | 3700000 |
| Perryridge  | Horseneck   | 1700000 |
| Pownal      | Bennington  |  300000 |
| Redwood     | Palo Alto   | 2100000 |
| Round Hill  | Horseneck   | 8000000 |
+-------------+-------------+---------+
8 rows in set (0.0141 sec)
```

```
SELECT * FROM account;
```

```
MySQL  localhost:3306 ssl  bankdb  SQL > SELECT * FROM account;
+----------------+-------------+---------+
| account_number | branch_name | balance |
+----------------+-------------+---------+
| A-101          | Downtown    |     500 |
| A-102          | Perryridge  |     400 |
| A-201          | Brighton    |     900 |
| A-215          | Mianus      |     700 |
| A-217          | Brighton    |     750 |
| A-222          | Redwood     |     700 |
| A-305          | Round Hill  |     350 |
+----------------+-------------+---------+
7 rows in set (0.0016 sec)
```

```
SELECT * FROM loan;
```

```
MySQL  localhost:3306 ssl  bankdb  SQL > SELECT * FROM loan;
+-------------+-------------+--------+
| loan_number | branch_name | amount |
+-------------+-------------+--------+
| L-11        | Round Hill  |    900 |
| L-14        | Downtown    |   1500 |
| L-15        | Perryridge  |   1500 |
| L-16        | Perryridge  |   1300 |
| L-17        | Downtown    |   1000 |
| L-23        | Redwood     |   2000 |
| L-93        | Mianus      |    500 |
+-------------+-------------+--------+
7 rows in set (0.0021 sec)
```

```
borrower;
```

```
MySQL  localhost:3306 ssl  bankdb  SQL >    SELECT * FROM borrower;
+---------------+-------------+
| customer_name | loan_number |
+---------------+-------------+
| Adams         | L-16        |
| Curry         | L-93        |
| Hayes         | L-15        |
| Johnson       | L-14        |
| Jones         | L-17        |
| Smith         | L-11        |
| Smith         | L-23        |
| Williams      | L-17        |
+---------------+-------------+
8 rows in set (0.0101 sec)
```

```
/*1.Create a view consisting of branch names and the names of custome
rs who have either an account or a loan at that branch.
 Assume that view to be called all-customer.*/
      create view all_customer as      select branch_name ,
customer_name     from depositor , account     where
depositor.account_number = account.account_number     union
    (select branch_name , customer_name     from
borrower , loan      where borrower.loan_number =
loan.loan_number);



 -- For Output     select *
from all_customer;
```

SELECT * FROM

```
MySQL  localhost:3306 ssl  bankdb  SQL > select * from all_customer;
+-------------+---------------+
| branch_name | customer_name |
+-------------+---------------+
| Perryridge  | Hayes         |
| Downtown    | Johnson       |
| Brighton    | Johnson       |
| Brighton    | Jones         |
| Redwood     | Lindsay       |
| Mianus      | Smith         |
| Round Hill  | Turner        |
| Downtown    | Jones         |
| Downtown    | Williams      |
| Mianus      | Curry         |
| Perryridge  | Adams         |
| Redwood     | Smith         |
| Round Hill  | Smith         |
+-------------+---------------+
13 rows in set (0.0108 sec)
```

```
/*2.Create a view gives for each branch the sum of the amounts of
all  the loans at the branch.*/
     create view sumOfLoans as
select branch_name , sum(amount)
from loan     group by
branch_name;      -- For Output
select * from sumOfLoans;
```

```
MySQL  localhost:3306 ssl  bankdb  SQL > select * from sumOfLoans;
+-------------+-------------+
| branch_name | sum(amount) |
+-------------+-------------+
| Downtown    |        2500 |
| Mianus      |         500 |
| Perryridge  |        2800 |
| Redwood     |        2000 |
| Round Hill  |         900 |
+-------------+-------------+
5 rows in set (0.0088 sec)
```

```
/*3.Using the view allcustomer, we can find all customers of
the Perryridge branch.*/
```

```
     select customer_name     from
all_customer     where branch_name
= "Perryridge";
```

```
MySQL  localhost:3306 ssl  bankdb  SQL > select customer_name
                                      ->     from all_customer
                                      ->     where branch_name = "Perryridge";
+---------------+
| customer_name |
+---------------+
| Hayes         |
| Adams         |
+---------------+
2 rows in set (0.0006 sec)
```

```
/*4.Write a Query for below Relational algebraic notation :*/
     select
depositor.customer_name
from depositor     union
    (select  borrower.customer_name
from borrower);
```

```
MySQL  localhost:3306 ssl  bankdb  SQL >     select depositor.customer_name
                                      ->     from depositor
                                      ->     union
                                      ->     (select  borrower.customer_name
                                      ->     from borrower);
+---------------+
| customer_name |
+---------------+
| Hayes         |
| Johnson       |
| Jones         |
| Lindsay       |
| Smith         |
| Turner        |
| Adams         |
| Curry         |
| Williams      |
+---------------+
9 rows in set (0.0108 sec)
```

PROBLEM 2

Customer Table :

customer_id	cust_name	city	grade	salesman_id
3002	Nick Rimando	New York	100	5001
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3008	Julian Green	London	300	5002
3004	Fabian Johnson	Paris	300	5006
3009	Geoff Cameron	Berlin	100	5003
3003	Jozy Altidor	Moscow	200	5007
3001	Brad Guzan	London	300	5005

Salesman Table :

salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
5003	Lauson Hen	San Jose	0.12

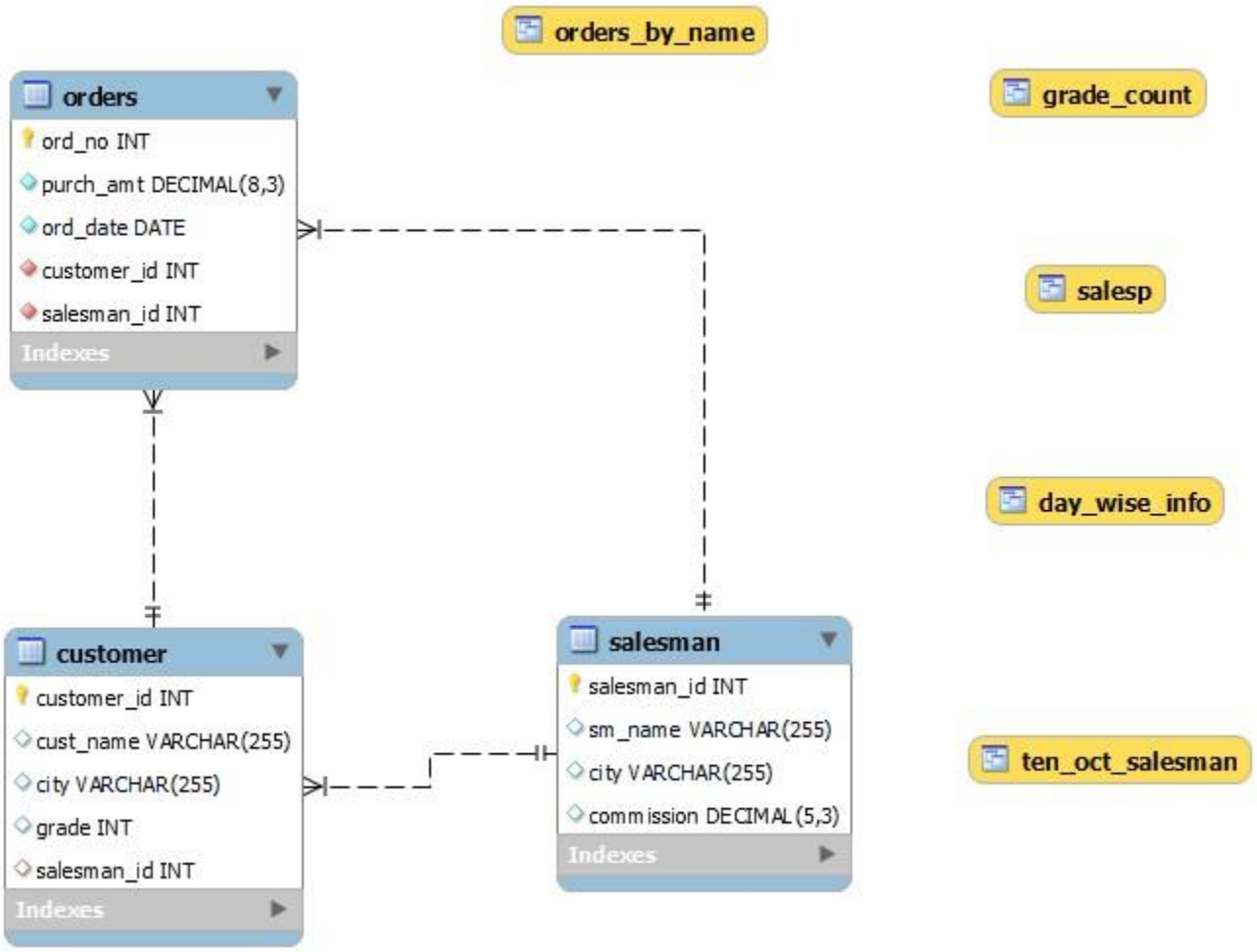
Orders Table :

ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	3004	5006
70003	2480.4	2012-10-10	3009	5003
70012	250.45	2012-06-27	3008	5002
70011	75.29	2012-08-17	3003	5007
70013	3045.6	2012-04-25	3002	5001

B) **Given three tables, perform the following queries:**

1. From the table, create a view to count the number of customers in each grade.
2. From the following table, create a view to count the number of unique customer, compute average and total purchase amount of customer orders by each date.
3. create a view to get the salesperson and customer by name. Return order name, purchase amount, salesperson ID, name, customer name.
4. create a view to find the salespersons who issued orders on October 10th, 2012. Return all the fields of salesperson.

ER Diagram :




```
CREATE DATABASE orderDB;
USE orderDB;

CREATE TABLE salesman(
salesman_id int NOT NULL,
sm_name varchar(255),
city varchar(255),
commission DECIMAL(5,3),
UNIQUE (salesman_id),
    PRIMARY KEY (salesman_id)
);
```

```
INSERT INTO salesman (salesman_id, sm_name, city, commission)
VALUES
        ('5001', 'James Hoog', 'New York', '0.15'),
        ('5002', 'Nail Knite', 'Paris', '0.13'),
        ('5005', 'Pit Alex', 'London', '0.11'),
        ('5006', 'Mc Lyon', 'Paris', '0.14'),
        ('5007', 'Paul Adam', 'Rome', '0.13'),
        ('5003', 'Lauson Hen', 'San Jose', '0.12');


CREATE TABLE customer(
customer_id int NOT NULL,
cust_name varchar(255),
city varchar(255),
grade int,     salesman_id
int,
    UNIQUE (customer_id),
    PRIMARY KEY (customer_id),
    FOREIGN KEY (salesman_id) REFERENCES salesman(salesman_id)
);

INSERT INTO customer(customer_id, cust_name, city,
                    grade, salesman_id)
 VALUES
      (3002 , 'Nick Rimando' , 'New York' , 100 , 5001),
      (3007 , 'Brad Davis' , 'New York' , 200 , 5001),
      (3005 , 'Graham Zusi' , 'California' , 200 , 5002),
      (3008 , 'Julian Green' , 'London' , 300 , 5002),
      (3004 , 'Fabian Johnson' , 'Paris' , 300 , 5006),
      (3009 , 'Geoff Cameron' , 'Berlin' , 100 , 5003),
      (3003 , 'Jozy Altidor' , 'Moscow' , 200 , 5007),
      (3001 , 'Brad Guzan' , 'London' , 300 , 5005);

CREATE TABLE orders(      ord_no
INT NOT NULL,     purch_amt
DECIMAL(8,3) NOT NULL,
ord_date DATE NOT NULL,
customer_id INT NOT NULL,
salesman_id INT NOT NULL,
```

```
    PRIMARY KEY (ord_no),
    FOREIGN KEY (customer_id) REFERENCES customer(customer_id),
FOREIGN KEY (salesman_id) REFERENCES salesman(salesman_id)
    );

INSERT INTO orders (ord_no, purch_amt, ord_date, customer_id,
salesma n_id)  VALUES
        ('70001', '150.5', '2012-10-05', '3005', '5002'),
        ('70009', '270.65', '2012-09-10', '3001',
'5005'),
        ('70002', '65.26', '2012-10-05', '3002', '5001'),
        ('70004', '110.5', '2012-08-17', '3009', '5003'),
        ('70007', '948.5', '2012-09-10', '3005', '5002'),
        ('70005', '2400.6', '2012-07-27', '3007',
'5001'),
        ('70008', '5760', '2012-09-10', '3002', '5001'),
        ('70010', '1983.43', '2012-10-10', '3004',
'5006'),
        ('70003', '2480.4', '2012-10-10', '3009',
'5003'),
        ('70012', '250.45', '2012-06-27', '3008',
'5002'),
        ('70011', '75.29', '2012-08-17', '3003', '5007'),
('70013', '3045.6', '2012-04-25', '3002', '5001');

      /*------------------------Queries---------------------------*/
SELECT * FROM customer;
```

```
MySQL  localhost:3306 ssl  orderdb  SQL > SELECT * FROM customer;
+-------------+---------------+-----------+-------+-------------+
| customer_id | cust_name     | city      | grade | salesman_id |
+-------------+---------------+-----------+-------+-------------+
|        3001 | Brad Guzan    | London    |   300 |        5005 |
|        3002 | Nick Rimando  | New York  |   100 |        5001 |
|        3003 | Jozy Altidor  | Moscow    |   200 |        5007 |
|        3004 | Fabian Johnson| Paris     |   300 |        5006 |
|        3005 | Graham Zusi   | California|   200 |        5002 |
|        3007 | Brad Davis    | New York  |   200 |        5001 |
|        3008 | Julian Green  | London    |   300 |        5002 |
|        3009 | Geoff Cameron | Berlin    |   100 |        5003 |
+-------------+---------------+-----------+-------+-------------+
8 rows in set (0.0095 sec)
```

```
SELECT * FROM salesman;
```

```
MySQL  localhost:3306 ssl  orderdb  SQL > SELECT * FROM salesman;
+-------------+------------+----------+------------+
| salesman_id | sm_name    | city     | commission |
+-------------+------------+----------+------------+
|        5001 | James Hoog | New York |      0.150 |
|        5002 | Nail Knite | Paris    |      0.130 |
|        5003 | Lauson Hen | San Jose |      0.120 |
|        5005 | Pit Alex   | London   |      0.110 |
|        5006 | Mc Lyon    | Paris    |      0.140 |
|        5007 | Paul Adam  | Rome     |      0.130 |
+-------------+------------+----------+------------+
6 rows in set (0.0005 sec)
```

```
SELECT * FROM orders;
```

```
MySQL  localhost:3306 ssl  orderdb  SQL > SELECT * FROM orders;
+--------+-----------+------------+-------------+-------------+
| ord_no | purch_amt | ord_date   | customer_id | salesman_id |
+--------+-----------+------------+-------------+-------------+
|  70001 |   150.500 | 2012-10-05 |        3005 |        5002 |
|  70002 |    65.260 | 2012-10-05 |        3002 |        5001 |
|  70003 |  2480.400 | 2012-10-10 |        3009 |        5003 |
|  70004 |   110.500 | 2012-08-17 |        3009 |        5003 |
|  70005 |  2400.600 | 2012-07-27 |        3007 |        5001 |
|  70007 |   948.500 | 2012-09-10 |        3005 |        5002 |
|  70008 |  5760.000 | 2012-09-10 |        3002 |        5001 |
|  70009 |   270.650 | 2012-09-10 |        3001 |        5005 |
|  70010 |  1983.430 | 2012-10-10 |        3004 |        5006 |
|  70011 |    75.290 | 2012-08-17 |        3003 |        5007 |
|  70012 |   250.450 | 2012-06-27 |        3008 |        5002 |
|  70013 |  3045.600 | 2012-04-25 |        3002 |        5001 |
+--------+-----------+------------+-------------+-------------+
12 rows in set (0.0006 sec)
```

```
-
- 1 From the table, create a view to count the number of customers
in  each grade.

    CREATE VIEW Grade_Count (Grade, Number)
    AS SELECT grade, COUNT(*)
    FROM customer
GROUP BY grade;

    SELECT * FROM Grade_Count;
```

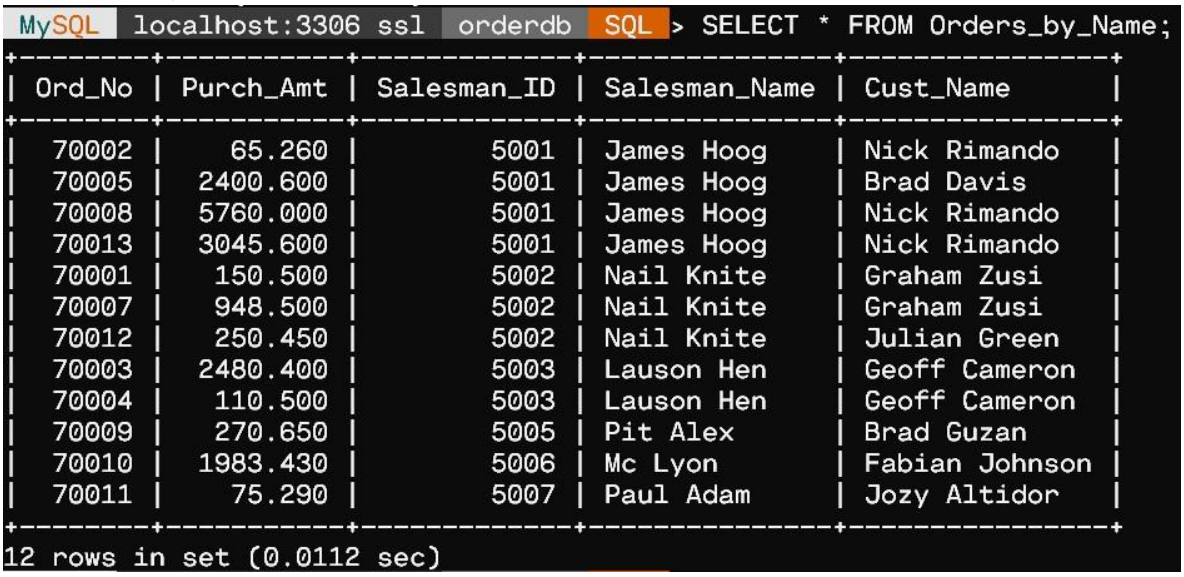
```
MySQL  localhost:3306 ssl  orderdb  SQL > SELECT * FROM Grade_Count;
+-------+--------+
| Grade | Number |
+-------+--------+
|   300 |      3 |
|   100 |      2 |
|   200 |      3 |
+-------+--------+
3 rows in set (0.0007 sec)
```

```
/* 2 From the following table, create a view to count the number of
u nique customer, compute average and total purchase amount of
customer order s by
each date. */

    CREATE VIEW Day_Wise_Info (Ord_Date, Unique_Cust, Avg_Purch,
Tota l_Purch)
    AS SELECT ord_date, COUNT(DISTINCT customer_id),
    AVG(purch_amt), SUM(purch_amt)
    FROM orders
    GROUP BY ord_date;
```

```
    SELECT * FROM Day_Wise_Info;
```

```
MySQL  localhost:3306 ssl  orderdb  SQL >    SELECT * FROM Day_Wise_Info;
+------------+-------------+--------------+-------------+
| Ord_Date   | Unique_Cust | Avg_Purch    | Total_Purch |
+------------+-------------+--------------+-------------+
| 2012-04-25 |           1 | 3045.6000000 |    3045.600 |
| 2012-06-27 |           1 |  250.4500000 |     250.450 |
| 2012-07-27 |           1 | 2400.6000000 |    2400.600 |
| 2012-08-17 |           2 |   92.8950000 |     185.790 |
| 2012-09-10 |           3 | 2326.3833333 |    6979.150 |
| 2012-10-05 |           2 |  107.8800000 |     215.760 |
| 2012-10-10 |           2 | 2231.9150000 |    4463.830 |
+------------+-------------+--------------+-------------+
7 rows in set (0.0006 sec)
```

```
/* 3 create a view to get the salesperson and customer by name. Retur
n order name, purchase amount, salesperson ID, name, customer name.
*/       CREATE VIEW Orders_by_Name (Ord_No,Purch_Amt,Salesman_ID,
Salesma n_Name, Cust_Name)
    AS SELECT ord_no, purch_amt, a.salesman_id, sm_name, cust_name
    FROM orders a, customer b, salesman c
    WHERE a.customer_id = b.customer_id
    AND a.salesman_id = c.salesman_id;

    SELECT * FROM
Orders_by_Name;
```



```
MySQL  localhost:3306 ssl  orderdb  SQL > SELECT * FROM Orders_by_Name;
+--------+-----------+-------------+---------------+----------------+
| Ord_No | Purch_Amt | Salesman_ID | Salesman_Name | Cust_Name      |
+--------+-----------+-------------+---------------+----------------+
|  70002 |    65.260 |        5001 | James Hoog    | Nick Rimando   |
|  70005 |  2400.600 |        5001 | James Hoog    | Brad Davis     |
|  70008 |  5760.000 |        5001 | James Hoog    | Nick Rimando   |
|  70013 |  3045.600 |        5001 | James Hoog    | Nick Rimando   |
|  70001 |   150.500 |        5002 | Nail Knite    | Graham Zusi    |
|  70007 |   948.500 |        5002 | Nail Knite    | Graham Zusi    |
|  70012 |   250.450 |        5002 | Nail Knite    | Julian Green   |
|  70003 |  2480.400 |        5003 | Lauson Hen    | Geoff Cameron  |
|  70004 |   110.500 |        5003 | Lauson Hen    | Geoff Cameron  |
|  70009 |   270.650 |        5005 | Pit Alex      | Brad Guzan     |
|  70010 |  1983.430 |        5006 | Mc Lyon       | Fabian Johnson |
|  70011 |    75.290 |        5007 | Paul Adam     | Jozy Altidor   |
+--------+-----------+-------------+---------------+----------------+
12 rows in set (0.0112 sec)
```

```
/* 4 create a view to find the salespersons who issued orders on Octo
ber 10th,
2012. Return all the fields of salesperson. */

CREATE VIEW Ten_Oct_Salesman (Salesman_ID, Salesman_Name, City, C
ommission)     AS SELECT *
    FROM salesman
    WHERE salesman_id IN
        (SELECT salesman_id
            FROM orders
WHERE ord_date = '2012-10-10');
```

```
SELECT * FROM Ten_Oct_Salesman;
```

```
MySQL  localhost:3306 ssl  orderdb  SQL > SELECT * FROM Ten_Oct_Salesman;
+-------------+---------------+---------+------------+
| Salesman_ID | Salesman_Name | City    | Commission |
+-------------+---------------+---------+------------+
|        5003 | Lauson Hen    | San Jose |      0.120 |
|        5006 | Mc Lyon       | Paris    |      0.140 |
+-------------+---------------+---------+------------+
2 rows in set (0.0007 sec)
```

Ex. No.: 05 Date: 08-09-21 Week:5

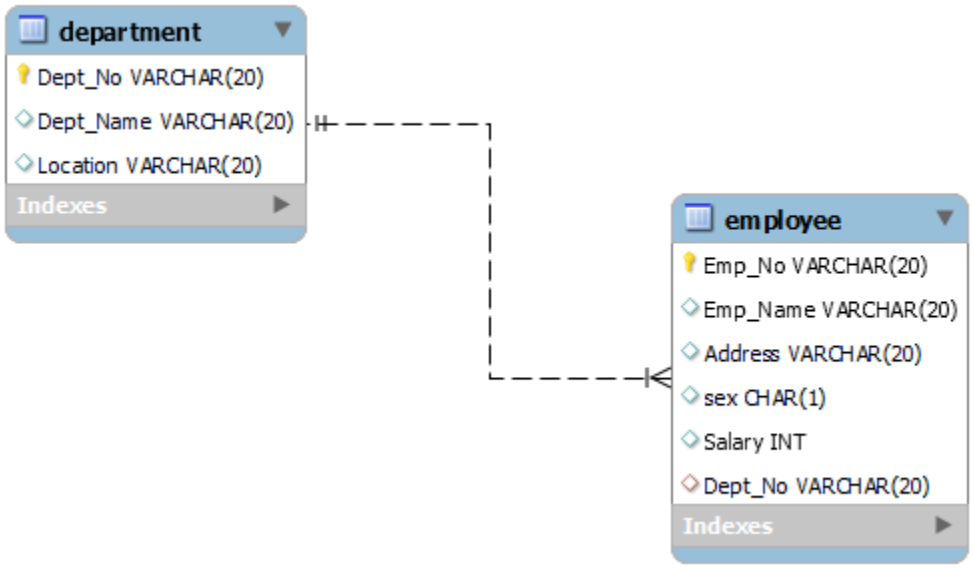
PROBLEM 1

Question 1

Employee (Emp_No, Emp_Name, sex, Salary, Address, Dept_No)
Department (Dept_No, Dept_Name, Location)

Insert at least 10 employees and 4 departments so that all queries result at least 1 tuple
1. Write a procedure to display employee details given the Emp_No.
2. Write a procedure to delete an employee record given the Emp_Name.
3. Write a procedure to list all the employee names belonging to a particular department given the Dept_No.
4. Write a procedure to display the number of employees whose salary is greater than 30K.

ER Diagram :




```
CREATE DATABASE CompanyDB;
USE CompanyDB;

CREATE TABLE Department  (
    Dept_No VARCHAR (20) PRIMARY KEY,
```

```
    Dept_Name VARCHAR (20),
    Location VARCHAR (20)
);

CREATE TABLE Employee (
    Emp_No VARCHAR (20) PRIMARY KEY,
    Emp_Name VARCHAR (20),
    Address VARCHAR (20),
    sex CHAR (1),
    Salary INTEGER,
    Dept_No VARCHAR (20),
    foreign key (Dept_No) REFERENCES Department (Dept_No)
);

INSERT INTO Department
    VALUES
        (1,'ACCOUNTS','Trichy'),
        (2,'IT','Chennai'),
        (3,'ECE','Salem'),
        (4,'ISE','Coimbatore'),
        (5,'CSE','Chennai');

INSERT INTO Employee
 VALUES
       ('ECE01','PREETI','BANGALORE','F', 40000,3),
       ('CSE01','JAMES','BANGALORE','M', 50000,5),
       ('CSE02','HEARN','BANGALORE','M', 70000,5),
       ('CSE03','EDWARD','MYSORE','M', 50000,5),
       ('CSE04','PAVAN','MANGALORE','M', 15000,5),
       ('CSE05','GIRISH','MYSORE','M', 20000,5),
       ('CSE06','NEHA','BANGALORE','F', 80000,5),
       ('ACC01','AHANA','MANGALORE','F', 35000,1),
       ('ACC02','SANTHOSH','MANGALORE','M', 30000,1),
       ('ISE01','VEENA','MYSORE','M', 10000,4),
       ('IT01','NAGESH','BANGALORE','M', 50000,2);
```

```
/*------------Queries-------------*/
```

```
SELECT * FROM Department;
```

```
MySQL localhost:3306 ssl SQL > use companyDB;
Default schema set to `companyDB`.
Fetching table and column names from `companydb` for auto-completion... Press ^C to stop.
MySQL localhost:3306 ssl companydb SQL > SELECT * FROM Department;
+---------+-----------+------------+
| Dept_No | Dept_Name | Location   |
+---------+-----------+------------+
| 1       | ACCOUNTS  | Trichy     |
| 2       | IT        | Chennai    |
| 3       | ECE       | Salem      |
| 4       | ISE       | Coimbatore |
| 5       | CSE       | Chennai    |
+---------+-----------+------------+
5 rows in set (0.0005 sec)
```

```
SELECT * FROM Employee;
```

```
MySQL localhost:3306 ssl companydb SQL > SELECT * FROM Employee;
+--------+----------+-----------+-----+--------+---------+
| Emp_No | Emp_Name | Address   | sex | Salary | Dept_No |
+--------+----------+-----------+-----+--------+---------+
| ACC01  | AHANA    | MANGALORE | F   |  35000 | 1       |
| ACC02  | SANTHOSH | MANGALORE | M   |  30000 | 1       |
| CSE01  | JAMES    | BANGALORE | M   |  50000 | 5       |
| CSE02  | HEARN    | BANGALORE | M   |  70000 | 5       |
| CSE03  | EDWARD   | MYSORE    | M   |  50000 | 5       |
| CSE04  | PAVAN    | MANGALORE | M   |  15000 | 5       |
| CSE05  | GIRISH   | MYSORE    | M   |  20000 | 5       |
| CSE06  | NEHA     | BANGALORE | F   |  80000 | 5       |
| ECE01  | PREETI   | BANGALORE | F   |  40000 | 3       |
| ISE01  | VEENA    | MYSORE    | M   |  10000 | 4       |
| IT01   | NAGESH   | BANGALORE | M   |  50000 | 2       |
+--------+----------+-----------+-----+--------+---------+
11 rows in set (0.0006 sec)
```

```
/*1.Write a procedure to display employee details given the Emp_No.*/

DELIMITER $$
CREATE PROCEDURE display_Emp(IN X VARCHAR(20))
    BEGIN
        SELECT * FROM Employee
        WHERE Emp_No=X;
    END$$
```

```
MySQL  localhost:3306 ssl  companydb  SQL > DELIMITER $$
MySQL  localhost:3306 ssl  companydb  SQL > CREATE PROCEDURE display_Emp(IN X VARCHAR(20))
                                      ->     BEGIN
                                      ->         SELECT * FROM Employee
                                      ->         WHERE Emp_No=X;
                                      ->     END$$
Query OK, 0 rows affected (0.0136 sec)
```

```
CALL display_Emp('CSE05')$$
```

```
MySQL  localhost:3306 ssl  companydb  SQL >
MySQL  localhost:3306 ssl  companydb  SQL > CALL display_Emp('CSE05')$$
+--------+----------+---------+-----+--------+---------+
| Emp_No | Emp_Name | Address | sex | Salary | Dept_No |
+--------+----------+---------+-----+--------+---------+
| CSE05  | GIRISH   | MYSORE  | M   |  20000 | 5       |
+--------+----------+---------+-----+--------+---------+
1 row in set (0.0007 sec)

Query OK, 0 rows affected (0.0007 sec)
```

```
/*2.Write a procedure to delete an employee record given the Emp_Name
.*/

DELIMITER //
CREATE PROCEDURE delete_Emp(IN NAME VARCHAR(20))
    BEGIN
        SET SQL_SAFE_UPDATES = 0;
        DELETE FROM Employee WHERE Emp_Name=NAME;
```

```
        SET SQL_SAFE_UPDATES = 1;
    END//

CALL delete_Emp('VEENA')//
```

```
MySQL  localhost:3306 ssl  companydb  SQL > DELIMITER //
MySQL  localhost:3306 ssl  companydb  SQL > CREATE PROCEDURE delete_Emp(IN NAME VARCHAR(20))
                                         ->     BEGIN
                                         ->         SET SQL_SAFE_UPDATES = 0;
                                         ->         DELETE FROM Employee WHERE Emp_Name=NAME;
                                         ->         SET SQL_SAFE_UPDATES = 1;
                                         ->     END//
Query OK, 0 rows affected (0.0082 sec)
MySQL  localhost:3306 ssl  companydb  SQL >
MySQL  localhost:3306 ssl  companydb  SQL > CALL delete_Emp('VEENA')//
Query OK, 0 rows affected (0.0065 sec)
```

```
SELECT * FROM Employee//
```

```
MySQL  localhost:3306 ssl  companydb  SQL > SELECT * FROM Employee
                                         -> //
+--------+----------+-----------+-----+--------+---------+
| Emp_No | Emp_Name | Address   | sex | Salary | Dept_No |
+--------+----------+-----------+-----+--------+---------+
| ACC01  | AHANA    | MANGALORE | F   |  35000 | 1       |
| ACC02  | SANTHOSH | MANGALORE | M   |  30000 | 1       |
| CSE01  | JAMES    | BANGALORE | M   |  50000 | 5       |
| CSE02  | HEARN    | BANGALORE | M   |  70000 | 5       |
| CSE03  | EDWARD   | MYSORE    | M   |  50000 | 5       |
| CSE04  | PAVAN    | MANGALORE | M   |  15000 | 5       |
| CSE05  | GIRISH   | MYSORE    | M   |  20000 | 5       |
| CSE06  | NEHA     | BANGALORE | F   |  80000 | 5       |
| ECE01  | PREETI   | BANGALORE | F   |  40000 | 3       |
| IT01   | NAGESH   | BANGALORE | M   |  50000 | 2       |
+--------+----------+-----------+-----+--------+---------+
10 rows in set (0.0004 sec)
```

```
/*3. Write a procedure to list all the employee names belonging to a
particular department given the Dept_No.*/

DELIMITER %%
CREATE PROCEDURE list_Emp(IN X INT)
    BEGIN
        SELECT * FROM Employee
```

```
        WHERE Dept_No=X;
    END%%
```

```
MySQL  localhost:3306 ssl  companydb  SQL > DELIMITER %%
MySQL  localhost:3306 ssl  companydb  SQL > CREATE PROCEDURE list_Emp(IN X INT)
                                      -> BEGIN
                                      -> SELECT * FROM Employee
                                      ->         WHERE Dept_No=X;
                                      -> END%%
Query OK, 0 rows affected (0.0079 sec)
```

```
CALL list_Emp(1)%%
```

```
MySQL  localhost:3306 ssl  companydb  SQL >
MySQL  localhost:3306 ssl  companydb  SQL > CALL list_Emp(1)%%
+--------+----------+-----------+-----+--------+---------+
| Emp_No | Emp_Name | Address   | sex | Salary | Dept_No |
+--------+----------+-----------+-----+--------+---------+
| ACC01  | AHANA    | MANGALORE | F   |  35000 | 1       |
| ACC02  | SANTHOSH | MANGALORE | M   |  30000 | 1       |
+--------+----------+-----------+-----+--------+---------+
2 rows in set (0.0024 sec)

Query OK, 0 rows affected, 2 warnings (0.0024 sec)
```

```
/*4. Write a procedure to display the number of employees whose salar
y is greater than 30K.*/
DELIMITER &&
CREATE PROCEDURE num_Emp()
    BEGIN
        SELECT COUNT(*) FROM Employee
        WHERE Salary>30000;
    END&&
CALL num_Emp()&&
```

```
MySQL  localhost:3306 ssl  companydb  SQL > DELIMITER &&
MySQL  localhost:3306 ssl  companydb  SQL > CREATE PROCEDURE num_Emp()
                                       ->     BEGIN
                                       ->         SELECT COUNT(*) FROM Employee
                                       ->         WHERE Salary>30000;
                                       ->     END&&
Query OK, 0 rows affected (0.0097 sec)
MySQL  localhost:3306 ssl  companydb  SQL > CALL num_Emp()&&
+----------+
| COUNT(*) |
+----------+
|        7 |
+----------+
1 row in set (0.0012 sec)

Query OK, 0 rows affected (0.0012 sec)
```

PROBLEM 2

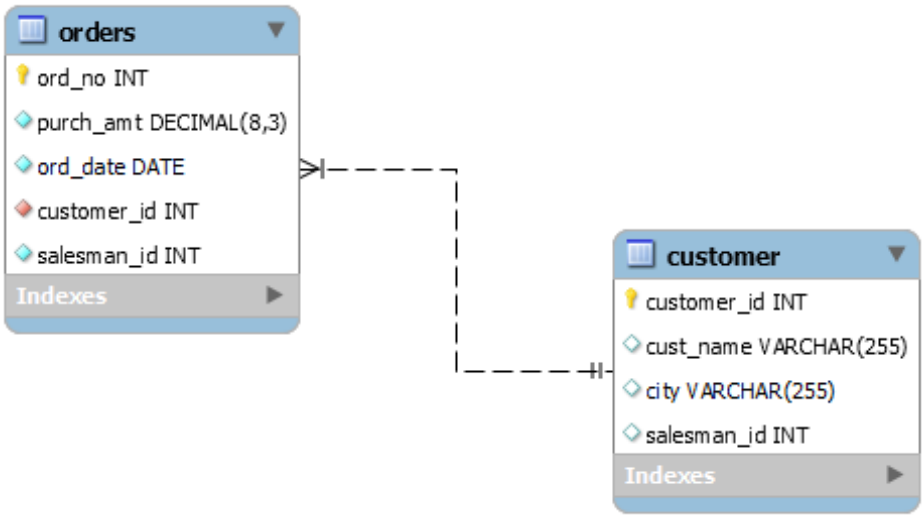
Question 2

Orders (ord_no, purch_amt, ord_date, customer_id, salesman_id)
Customer (customer_id, cust_name, city, salesman_id)

Insert at least 8 tuples in both relation so that all queries result at least 1 tuple

1. Use SQL functions to find the highest purchase amount ordered by each customer with their ID and highest purchase amount.
2. Use SQL functions to find the number of customers in each city.
3. Write a user defined function to display customer details given the customer_id.
4. Write a user defined function to decrease the purch_amt of an order by 5% given the ord_no.

ER Diagram :



```sql
CREATE DATABASE orderDB;
USE orderDB;

CREATE TABLE customer(
    customer_id int NOT NULL,
    cust_name varchar(255),
    city varchar(255),
    salesman_id int,
    UNIQUE (customer_id),
    PRIMARY KEY (customer_id)
);

INSERT INTO customer
 VALUES
    (3002 , 'Nick Rimando' , 'New York' , 5001),
    (3007 , 'Brad Davis' , 'New York'  , 5001),
    (3005 , 'Graham Zusi' , 'California' , 5002),
    (3008 , 'Julian Green' , 'London' , 5002),
    (3004 , 'Fabian Johnson' , 'Paris' , 5006),
    (3009 , 'Geoff Cameron' , 'Berlin' , 5003),
    (3003 , 'Jozy Altidor' , 'Moscow' , 5007),
    (3001 , 'Brad Guzan' , 'London' , 5005);

CREATE TABLE orders(
    ord_no INT NOT NULL,
    purch_amt DECIMAL(8,3) NOT NULL,
    ord_date DATE NOT NULL,
    customer_id INT NOT NULL,
    salesman_id INT NOT NULL,
    PRIMARY KEY (ord_no),
    FOREIGN KEY (customer_id) REFERENCES customer(customer_id)
    );

INSERT INTO orders
 VALUES
    ('70001', '150.5', '2012-10-05', '3005', '5002'),
    ('70009', '270.65', '2012-09-10', '3001', '5005'),
```

```
    ('70002', '65.26', '2012-10-05', '3002', '5001'),
    ('70004', '110.5', '2012-08-17', '3009', '5003'),
    ('70007', '948.5', '2012-09-10', '3005', '5002'),
    ('70005', '2400.6', '2012-07-27', '3007', '5001'),
    ('70008', '5760', '2012-09-10', '3002', '5001'),
    ('70010', '1983.43', '2012-10-10', '3004', '5006'),
    ('70003', '2480.4', '2012-10-10', '3009', '5003'),
    ('70012', '250.45', '2012-06-27', '3008', '5002'),
    ('70011', '75.29', '2012-08-17', '3003', '5007'),
    ('70013', '3045.6', '2012-04-25', '3002', '5001');


        /*------------- Queries---------------- */

SET GLOBAL log_bin_trust_function_creators = 1;

SELECT * FROM customer;
```

```
MySQL  localhost:3306 ssl  orderdb  SQL > SELECT * FROM customer;
+-------------+---------------+------------+-------------+
| customer_id | cust_name     | city       | salesman_id |
+-------------+---------------+------------+-------------+
|        3001 | Brad Guzan    | London     |        5005 |
|        3002 | Nick Rimando  | New York   |        5001 |
|        3003 | Jozy Altidor  | Moscow     |        5007 |
|        3004 | Fabian Johnson| Paris      |        5006 |
|        3005 | Graham Zusi   | California |        5002 |
|        3007 | Brad Davis    | New York   |        5001 |
|        3008 | Julian Green  | London     |        5002 |
|        3009 | Geoff Cameron | Berlin     |        5003 |
+-------------+---------------+------------+-------------+
8 rows in set (0.0007 sec)
```

```
SELECT * FROM orders;
```

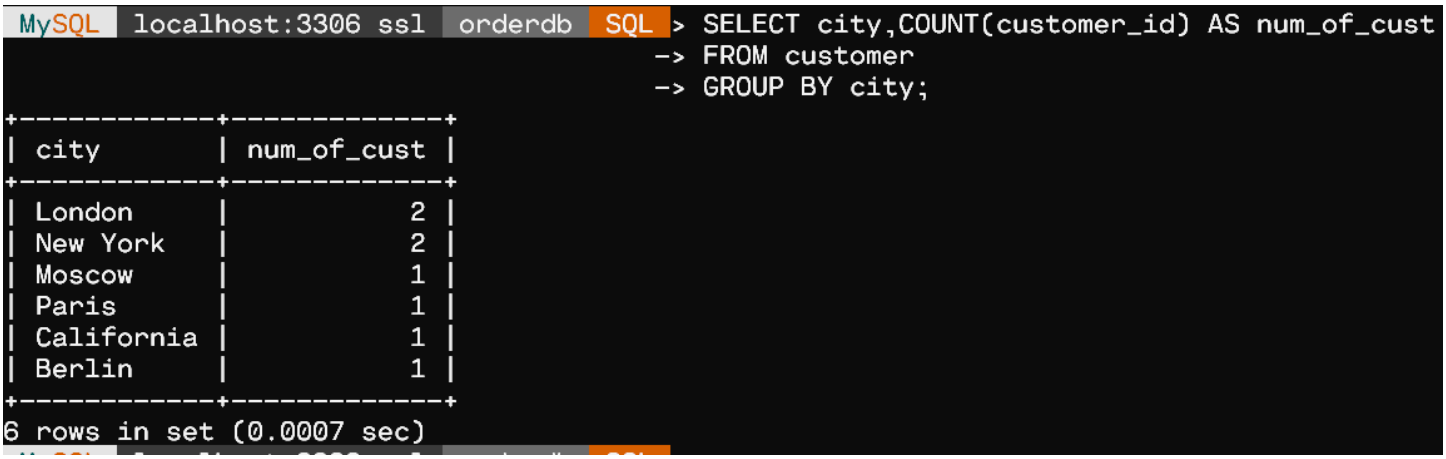
```
MySQL  localhost:3306 ssl  orderdb  SQL > SELECT * FROM orders;
+--------+-----------+------------+-------------+-------------+
| ord_no | purch_amt | ord_date   | customer_id | salesman_id |
+--------+-----------+------------+-------------+-------------+
|  70001 |   150.500 | 2012-10-05 |        3005 |        5002 |
|  70002 |    65.260 | 2012-10-05 |        3002 |        5001 |
|  70003 |  2480.400 | 2012-10-10 |        3009 |        5003 |
|  70004 |   110.500 | 2012-08-17 |        3009 |        5003 |
|  70005 |  2400.600 | 2012-07-27 |        3007 |        5001 |
|  70007 |   948.500 | 2012-09-10 |        3005 |        5002 |
|  70008 |  5760.000 | 2012-09-10 |        3002 |        5001 |
|  70009 |   270.650 | 2012-09-10 |        3001 |        5005 |
|  70010 |  1983.430 | 2012-10-10 |        3004 |        5006 |
|  70011 |    75.290 | 2012-08-17 |        3003 |        5007 |
|  70012 |   250.450 | 2012-06-27 |        3008 |        5002 |
|  70013 |  3045.600 | 2012-04-25 |        3002 |        5001 |
+--------+-----------+------------+-------------+-------------+
12 rows in set (0.0009 sec)
```

```sql
/*  1. Use SQL functions to find the highest purchase amount ordered by
       each customer with their ID and highest purchase amount */

SELECT customer_id,MAX(purch_amt) AS highest_purch_amt
FROM orders
GROUP BY customer_id;
```

```
MySQL  localhost:3306 ssl  orderdb  SQL > SELECT customer_id,MAX(purch_amt) AS highest_purch_amt
                                      -> FROM orders
                                      -> GROUP BY customer_id;
+-------------+-------------------+
| customer_id | highest_purch_amt |
+-------------+-------------------+
|        3001 |           270.650 |
|        3002 |          5760.000 |
|        3003 |            75.290 |
|        3004 |          1983.430 |
|        3005 |           948.500 |
|        3007 |          2400.600 |
|        3008 |           250.450 |
|        3009 |          2480.400 |
+-------------+-------------------+
8 rows in set (0.0004 sec)
```

```sql
-  2. Use SQL functions to find the number of customers in each city.
```

```sql
SELECT city,COUNT(customer_id) AS num_of_cust
FROM customer
GROUP BY city;
```



- 3. Write a user defined function to display customer details given the customer_id.

```sql
DELIMITER %%

CREATE FUNCTION display_cust(c int)
RETURNS varchar(255)
    BEGIN
        DECLARE ret_val varchar(255);
        SELECT cust_name INTO ret_val FROM customer
        WHERE customer_id=c;
        RETURN ret_val;
    END%%

SELECT display_cust(3002)%%
```

```
MySQL  localhost:3306 ssl  orderdb  SQL > DELIMITER %%
MySQL  localhost:3306 ssl  orderdb  SQL >
MySQL  localhost:3306 ssl  orderdb  SQL > CREATE FUNCTION display_cust(c int)
                                        -> RETURNS varchar(255)
                                        ->     BEGIN
                                        ->         DECLARE ret_val varchar(255);
                                        ->         SELECT cust_name INTO ret_val FROM customer
                                        ->         WHERE customer_id=c;
                                        ->         RETURN ret_val;
                                        ->     END%%
Query OK, 0 rows affected (0.0082 sec)
MySQL  localhost:3306 ssl  orderdb  SQL >
MySQL  localhost:3306 ssl  orderdb  SQL > SELECT display_cust(3002);
                                        -> %%
+--------------------+
| display_cust(3002) |
+--------------------+
| Nick Rimando       |
+--------------------+
1 row in set (0.0023 sec)
```

```
-  4. Write a user defined function to decrease the purch_amt of an o
rder by 5% given the ord_no.
DELIMITER $$

CREATE FUNCTION decrease_purch_amt( ord_no int)
RETURNS DECIMAL(8,3) DETERMINISTIC
BEGIN
    DECLARE Reduced_purch_amt DECIMAL(8,3);
    SELECT 0.95 * purch_amt INTO Reduced_purch_amt FROM orders WHERE
orders.ord_no = ord_no;
    RETURN Reduced_purch_amt;
END$$

SELECT decrease_purch_amt(70009)$$
```

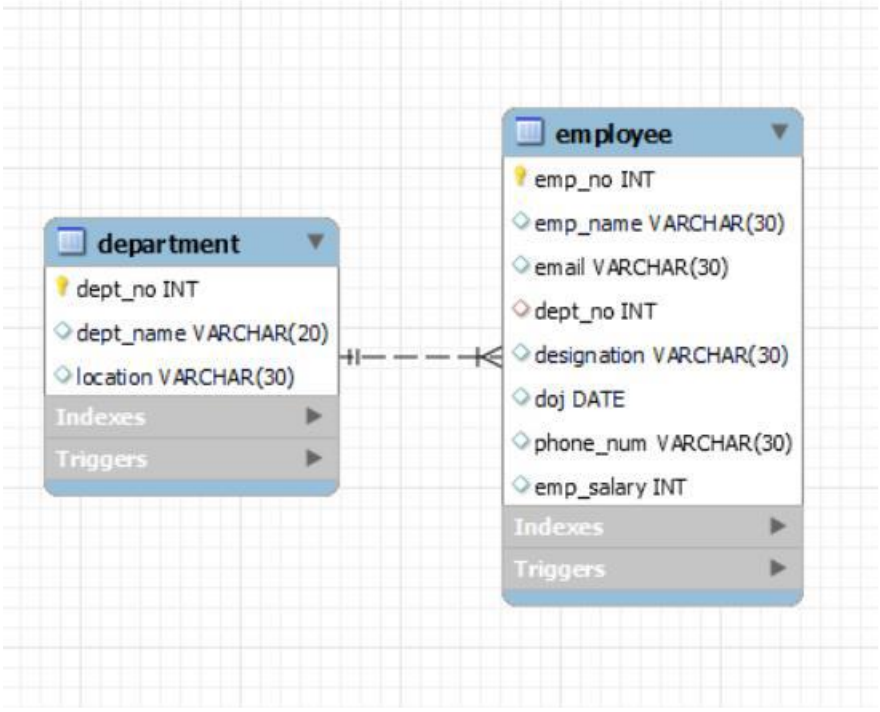
```
MySQL  localhost:3306 ssl  orderdb  SQL > DELIMITER $$
MySQL  localhost:3306 ssl  orderdb  SQL >
MySQL  localhost:3306 ssl  orderdb  SQL > CREATE FUNCTION decrease_purch_amt( ord_no int)
                                        -> RETURNS DECIMAL(8,3) DETERMINISTIC
                                        -> BEGIN
                                        ->     DECLARE Reduced_purch_amt DECIMAL(8,3);
                                        ->     SELECT 0.95 * purch_amt INTO Reduced_purch_amt FROM orders WHERE orders.ord_no = ord_no;
                                        ->     RETURN Reduced_purch_amt;
                                        -> END$$
Query OK, 0 rows affected (0.0061 sec)
MySQL  localhost:3306 ssl  orderdb  SQL >
MySQL  localhost:3306 ssl  orderdb  SQL > SELECT decrease_purch_amt(70009)$$
+---------------------------+
| decrease_purch_amt(70009) |
+---------------------------+
|                   257.118 |
+---------------------------+
1 row in set (0.0011 sec)
```

PROBLEM 1

1. Consider the following tables namely "DEPARTMENTS" and "EMPLOYEES" Their schemas are as follows:

DEPARTMENTS (dept _no , dept_ name , dept_location);
EMPLOYEES (emp_id , emp_name , email, dept, designation, DOJ, phone_num, emp_salary);

ER-Diagram




```
CREATE DATABASE CompanyDB;
USE CompanyDB;
```

```
CREATE TABLE Departments  (
    dept_id VARCHAR (20) NOT NULL PRIMARY KEY,
    dept_name VARCHAR (20),
    dept_location VARCHAR (20)
);

CREATE TABLE Employees (
  emp_id INT NOT NULL PRIMARY KEY,
  emp_name VARCHAR(100),
  email VARCHAR(100),
  dept_id  INT,
  designation VARCHAR(100),
  doj DATE,
  phone_num VARCHAR(45),
  emp_salary DECIMAL(12,3)
);

create table messages
(
    id int(11) not null auto_increment primary key,
    message varchar(255) default null,
    time timestamp null default current_timestamp on update current_t
imestamp
);


/** ----------------------------- Queries ---------------------------
----**/

-- Create a trigger for:-

-- a) Calculating the number of rows inserted so far.

delimiter //

create trigger trig1A
 before insert on departments
  for each row
```

```
  set @dept_rows = @dept_rows+1//

set @dept_rows = 0//

delimiter ;

insert into departments
values
    ('1', 'sales', 'chennai'),
    ('2', 'marketing', 'bombay'),
    ('3', 'management', 'bengaluru');

SELECT @dept_rows;
```

```
 MySQL  localhost:3306 ssl  SQL > use companydb;
Default schema set to `companydb`.
Fetching table and column names from `companydb` for auto-completion... Press ^C to stop.
 MySQL  localhost:3306 ssl  companydb  SQL > delimiter //
 MySQL  localhost:3306 ssl  companydb  SQL >
 MySQL  localhost:3306 ssl  companydb  SQL > create trigger trig1A
                                          ->  before insert on departments
                                          ->   for each row
                                          ->   set @dept_rows = @dept_rows+1//
Query OK, 0 rows affected (0.0138 sec)
 MySQL  localhost:3306 ssl  companydb  SQL >
 MySQL  localhost:3306 ssl  companydb  SQL > set @dept_rows = 0//
Query OK, 0 rows affected (0.0002 sec)
 MySQL  localhost:3306 ssl  companydb  SQL >
 MySQL  localhost:3306 ssl  companydb  SQL > delimiter ;
 MySQL  localhost:3306 ssl  companydb  SQL >
 MySQL  localhost:3306 ssl  companydb  SQL > insert into departments
                                          -> values
                                          ->     ('1', 'sales', 'chennai'),
                                          ->     ('2', 'marketing', 'bombay'),
                                          ->     ('3', 'management', 'bengaluru');
Query OK, 3 rows affected (0.0086 sec)

Records: 3  Duplicates: 0  Warnings: 0
 MySQL  localhost:3306 ssl  companydb  SQL >
 MySQL  localhost:3306 ssl  companydb  SQL > SELECT @dept_rows;
+------------+
| @dept_rows |
+------------+
|          3 |
+------------+
1 row in set (0.0008 sec)
```

```
-- b) Displays message prior to insert operation on Employee Table
```

```
delimiter //

create trigger display_message before insert on employees
    for each row
    begin
    insert into messages (message) values('Inserted Row in Employees'
);
    end;//

delimiter ;

insert into employees
values
    ('100','timber', 'timber@gmail.com','1', 'salseman', '2019-01-
12', '9842042345', '100000'),
    ('101','jaanu', 'jaanu@gmail.com', '2', 'senior manager', '2020-
01-01', '9865161122', '15000'),
    ('102','jen', 'jen@gmail.com', '3', 'junior manager', '2019-10-
12', '9042842344', '60000'),
    ('103','jack', 'jack@gmail.com', '2', 'assistant', '2016-03-
01', '9498853055', '50000');

SELECT * FROM messages;
```

```
MySQL  localhost:3306 ssl  companydb  SQL > delimiter //
MySQL  localhost:3306 ssl  companydb  SQL >
MySQL  localhost:3306 ssl  companydb  SQL > create trigger display_message before insert on employees
                                        ->     for each row
                                        ->     begin
                                        ->     insert into messages (message) values('Inserted Row in Employees');
                                        ->     end;//
Query OK, 0 rows affected (0.0134 sec)
MySQL  localhost:3306 ssl  companydb  SQL >
MySQL  localhost:3306 ssl  companydb  SQL > delimiter ;
MySQL  localhost:3306 ssl  companydb  SQL >
MySQL  localhost:3306 ssl  companydb  SQL > insert into employees
                                        -> values
                                        ->     ('100','timber', 'timber@gmail.com','1', 'salseman', '2019-01-12', '9842042345', '100000'),
                                        ->     ('101','jaanu', 'jaanu@gmail.com', '2', 'senior manager', '2020-01-01', '9865161122', '15000'),
                                        ->     ('102','jen', 'jen@gmail.com', '3', 'junior manager', '2019-10-12', '9042842344', '60000'),
                                        ->     ('103','jack', 'jack@gmail.com', '2', 'assistant', '2016-03-01', '9498853055', '50000');
Query OK, 4 rows affected (0.0109 sec)

Records: 4  Duplicates: 0  Warnings: 0
MySQL  localhost:3306 ssl  companydb  SQL >
MySQL  localhost:3306 ssl  companydb  SQL > SELECT * FROM messages;
+----+--------------------------+---------------------+
| id | message                  | time                |
+----+--------------------------+---------------------+
|  1 | Inserted Row in Employees | 2021-09-16 17:29:17 |
|  2 | Inserted Row in Employees | 2021-09-16 17:29:17 |
|  3 | Inserted Row in Employees | 2021-09-16 17:29:17 |
|  4 | Inserted Row in Employees | 2021-09-16 17:29:17 |
+----+--------------------------+---------------------+
4 rows in set (0.0007 sec)
```

-
- c) Ensuring that the Employee Table does not contain duplicates or null value in the Name column

```
delimiter //
create trigger NULL_val_trigger before insert on employees
for each row
begin
    if (exists(select 1 from employees where emp_name = new.emp_name
or new.emp_name is null))
    then
    signal sqlstate '45001' set message_text = "INSERT failed due to
duplicate or null emp_name";
    end if;
end//

insert into employees
  values ('104', 'jen', 'xjen@gmail.com', '3', 'junior manager', '201
9-09-10', '7561994181', '62000')//

delimiter ;
select * from employees;
```

```
MySQL  localhost:3306 ssl  companydb  SQL > delimiter //
MySQL  localhost:3306 ssl  companydb  SQL > create trigger NULL_val_trigger before insert on employees
                                      -> for each row
                                      -> begin
                                      ->     if (exists(select 1 from employees where emp_name = new.emp_name or new.emp_name is null))
                                      ->     then
                                      ->     signal sqlstate '45001' set message_text = "INSERT failed due to duplicate or null emp_name";
                                      ->     end if;
                                      -> end//
Query OK, 0 rows affected (0.0116 sec)
MySQL  localhost:3306 ssl  companydb  SQL >
MySQL  localhost:3306 ssl  companydb  SQL > insert into employees
                                      ->   values ('104', 'jen', 'xjen@gmail.com', '3', 'junior manager', '2019-09-10', '7561994181', '62000')
//
ERROR: 1644 (45001): INSERT failed due to duplicate or null emp_name
MySQL  localhost:3306 ssl  companydb  SQL >
MySQL  localhost:3306 ssl  companydb  SQL > delimiter ;
MySQL  localhost:3306 ssl  companydb  SQL > select * from employees;
+--------+----------+-----------------+---------+---------------+------------+------------+------------+
| emp_id | emp_name | email           | dept_id | designation   | doj        | phone_num  | emp_salary |
+--------+----------+-----------------+---------+---------------+------------+------------+------------+
|    100 | timber   | timber@gmail.com |       1 | salseman      | 2019-01-12 | 9842042345 | 100000.000 |
|    101 | jaanu    | jaanu@gmail.com  |       2 | senior manager | 2020-01-01 | 9865161122 |  15000.000 |
|    102 | jen      | jen@gmail.com    |       3 | junior manager | 2019-10-12 | 9042842344 |  60000.000 |
|    103 | jack     | jack@gmail.com   |       2 | assistant     | 2016-03-01 | 9498853055 |  50000.000 |
+--------+----------+-----------------+---------+---------------+------------+------------+------------+
4 rows in set (0.0010 sec)
```

```
-
- d) To insert Employee details into Employee Table only of DOJ > 201
8

DELIMITER //
CREATE TRIGGER NOT_BELOW_2018
    BEFORE INSERT
    ON EMPLOYEES
    FOR EACH ROW
BEGIN
    IF(NEW.doj < '2018-01-01') THEN
        SIGNAL SQLSTATE VALUE '45000' SET MESSAGE_TEXT = 'EMPLOYEE BE
LOW 2018 DOJ CANNOT BE INSERTED';
    END IF;
END //
insert into employees
 values (104, 'raj', 'rajx@gmail.com', '2', 'field officer', '2017-
03-10','9680988342','34000')//
DELIMITER ;
```

```
MySQL  localhost:3306 ssl  companydb  SQL > DELIMITER //
MySQL  localhost:3306 ssl  companydb  SQL > CREATE TRIGGER NOT_BELOW_2018
                                      ->     BEFORE INSERT
                                      ->     ON EMPLOYEES
                                      ->     FOR EACH ROW
                                      -> BEGIN
                                      ->     IF(NEW.doj < '2018-01-01') THEN
                                      ->         SIGNAL SQLSTATE VALUE '45000' SET MESSAGE_TEXT = 'EMPLOYEE BELOW 2018 DOJ CANNOT BE INSERTED'
;
                                      ->     END IF;
                                      -> END //
Query OK, 0 rows affected (0.0110 sec)
MySQL  localhost:3306 ssl  companydb  SQL > insert into employees
                                      ->  values (104, 'raj', 'rajx@gmail.com', '2', 'field officer', '2017-03-10','9680988342','34000')//
ERROR: 1644 (45000): EMPLOYEE BELOW 2018 DOJ CANNOT BE INSERTED
MySQL  localhost:3306 ssl  companydb  SQL > DELIMITER ;
```

```
-
- e) To Prevent any Employee Named Tom to be inserted into the table

DELIMITER //
create trigger prevent_name before insert on employees
    for each row
    begin
    if new.emp_name = "tom" then signal sqlstate '45000' set message_
text= "tom is prohibited from insertion !!!";
    end if;
```

```
    end//
DELIMITER ;

insert into employees values
  ('107', 'tom', 'tommy@gmail.com', '6', 'assistant doctor', '2020-
02-22', '7373558197','25000');

select * from employees;
```

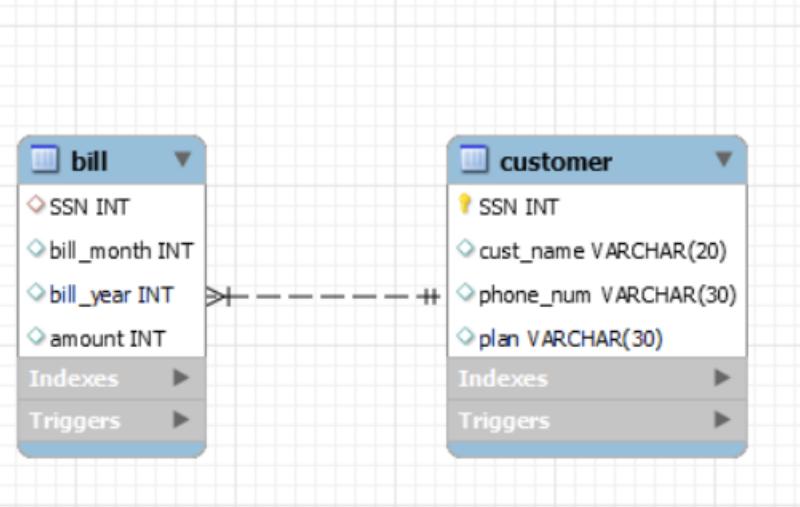
```
MySQL  localhost:3306 ssl  companydb  SQL > DELIMITER //
MySQL  localhost:3306 ssl  companydb  SQL > create trigger prevent_name before insert on employees
                                       ->     for each row
                                       ->     begin
                                       ->     if new.emp_name = "tom" then signal sqlstate '45000' set message_text= "tom is prohibited from in
sertion !!!";
                                       ->     end if;
                                       ->     end//
Query OK, 0 rows affected (0.0111 sec)
MySQL  localhost:3306 ssl  companydb  SQL > DELIMITER ;
MySQL  localhost:3306 ssl  companydb  SQL >
MySQL  localhost:3306 ssl  companydb  SQL > insert into employees values
                                       ->   ('107', 'tom', 'tommy@gmail.com', '6', 'assistant doctor', '2020-02-22', '7373558197','25000');
ERROR: 1644 (45000): tom is prohibited from insertion !!!
MySQL  localhost:3306 ssl  companydb  SQL >
MySQL  localhost:3306 ssl  companydb  SQL > select * from employees;
+--------+----------+-----------------+---------+---------------+------------+------------+------------+
| emp_id | emp_name | email           | dept_id | designation   | doj        | phone_num  | emp_salary |
+--------+----------+-----------------+---------+---------------+------------+------------+------------+
|    100 | timber   | timber@gmail.com |       1 | salseman      | 2019-01-12 | 9842042345 | 100000.000 |
|    101 | jaanu    | jaanu@gmail.com  |       2 | senior manager | 2020-01-01 | 9865161122 |  15000.000 |
|    102 | jen      | jen@gmail.com    |       3 | junior manager | 2019-10-12 | 9042842344 |  60000.000 |
|    103 | jack     | jack@gmail.com   |       2 | assistant     | 2016-03-01 | 9498853055 |  50000.000 |
+--------+----------+-----------------+---------+---------------+------------+------------+------------+
4 rows in set (0.0004 sec)
```

PROBLEM 2

2. Consider the following relational schema that manages the telephone bills of a mobile phone company.

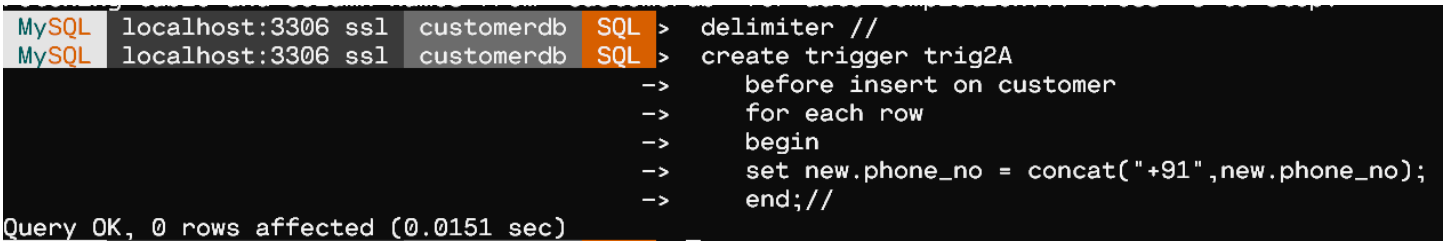
CUSTOMER (SSN, Name, PhoneNum, Plan)

BILL (SSN, Month, Year, amount)



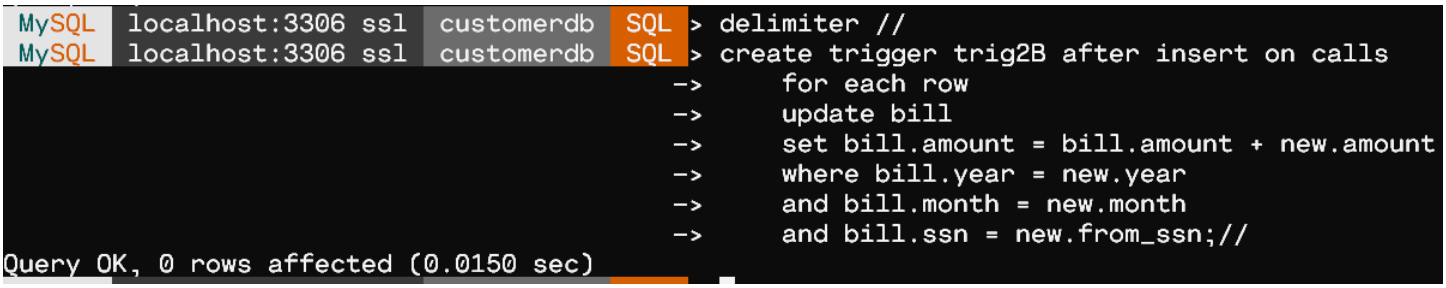

```
CREATE DATABASE customerDB;
USE customerDB;

create table customer
(
    ssn int primary key,
    name varchar(30),
    phone_no varchar(30),
    plan varchar(30)
);

create table bill
(
    ssn int not null,
    foreign key (ssn) references customer(ssn),
    month int not null,
```

```
    year int not null,
    amount int not null
);

create table calls
(
    from_ssn int,
    to_ssn int,
    month int,
    year int,
    amount int,
    foreign key(from_ssn) references customer(ssn),
    foreign key(to_ssn) references customer(ssn)
);

-
- a. That Adds +91 to all phone numbers as a prefix in the Customer T
able.

 delimiter //
 create trigger trig2A
    before insert on customer
    for each row
    begin
    set new.phone_no = concat("+91",new.phone_no);
    end;//
```




```
-
- b. Write a trigger that after each phone call updates the customer'
s bill.
delimiter //
```

```
create trigger trig2B after insert on calls
    for each row
    update bill
    set bill.amount = bill.amount + new.amount
    where bill.year = new.year
    and bill.month = new.month
    and bill.ssn = new.from_ssn;//
```

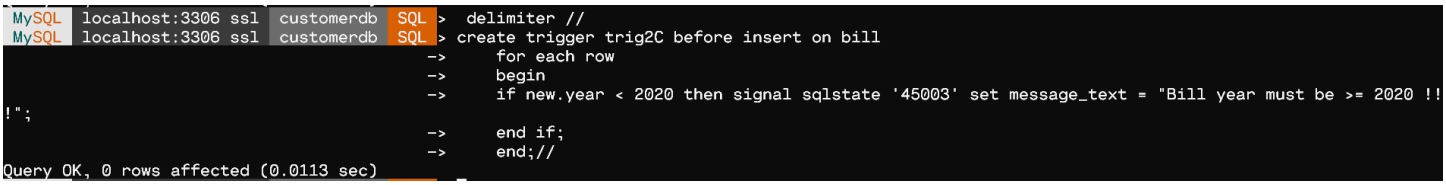


-

- c. Write a trigger to not insert bill older than 2020 (year > 2020)

.

```
 delimiter //
create trigger trig2C before insert on bill
    for each row
    begin
    if new.year < 2020 then signal sqlstate '45003' set message_text
= "Bill year must be >= 2020 !!!";
    end if;
    end;//
```



CODE

```
/*
CSLR-51 : DBMS LAB-7

Roll no. : 106119100
Name : Rajneesh Pandey
Section : CSE-B
*/

#include <bits/stdc++.h>
using namespace std;

vector<string> ans;
vector<int> mapps;
vector<int> nodes;
string relationship_alpha;
int relationship_alpha_len;
int dependency;

unordered_map<int, int> depends;
unordered_map<char, int> alpha_to_int;
unordered_map<int, char> int_to_alpha;

string mask_to_string(const int &mask)
{
    string str = "";
    for (int i = 0; i < relationship_alpha_len; i++)
    {
        if ((mask >> i) & 1)
            str += int_to_alpha[i];
    }
```

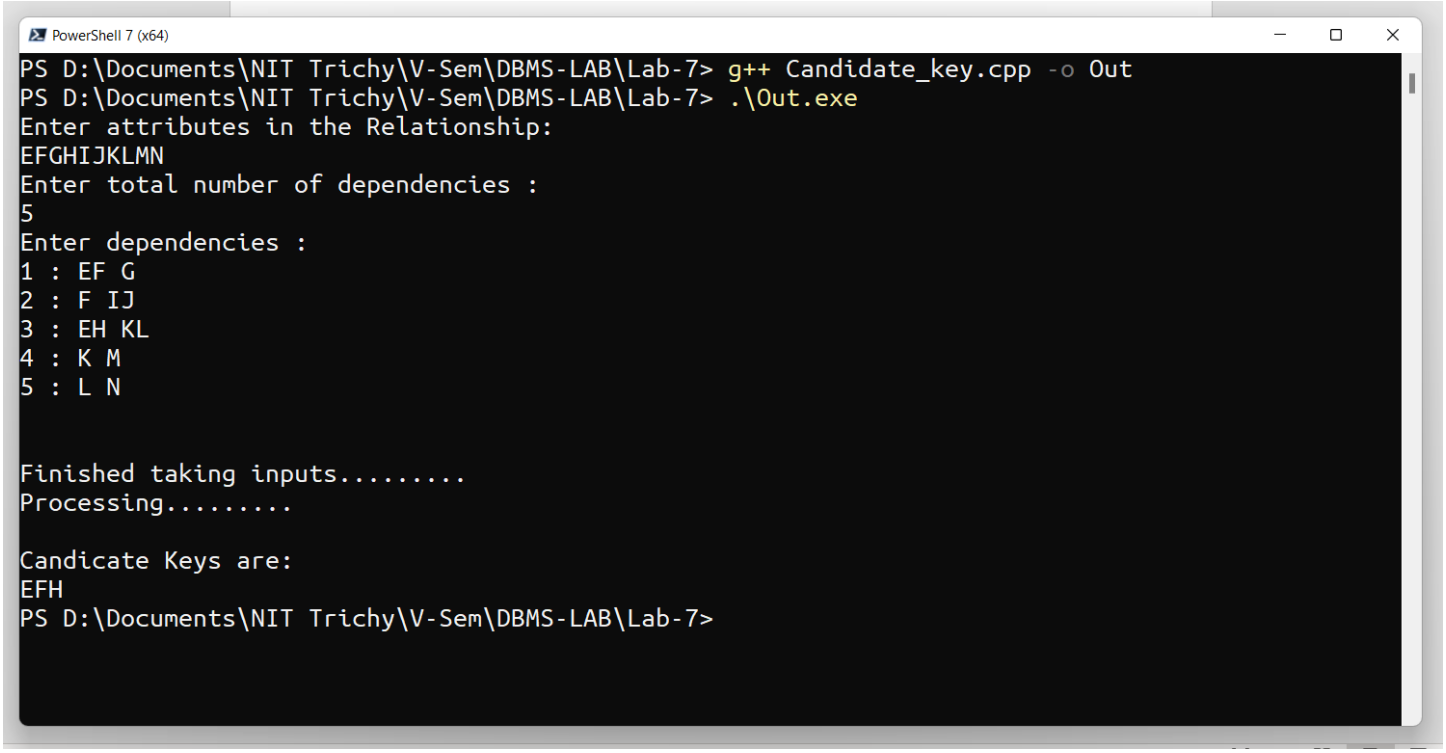
```
    return str;
}

int string_to_mask(const string &s)
{
    int mask = 0;
    for (auto &ch : s)
    {
        mask |= (1 << alpha_to_int[ch]);
    }

    return mask;
}

void init()
{

    cout << "Enter attributes in the Relationship: \n";
    cin >> relationship_alpha;

    relationship_alpha_len = relationship_alpha.length();

    mapps.assign(1 << relationship_alpha_len, 0);

    for (int i = 0; i < (1 << relationship_alpha_len); i++)
        mapps[i] = i;

    for (int i = 0; i < relationship_alpha_len; i++)
    {
        alpha_to_int[relationship_alpha[i]] = i;
        int_to_alpha[i] = relationship_alpha[i];
    }
    cout << "Enter total number of dependencies :\n";
    cin >> dependency;
    cout << "Enter dependencies : \n";
    for (int i = 0; i < dependency; i++)
    {
        cout<< i + 1 << " : ";
```

```
        string lhs, rhs;
        cin >> lhs >> rhs;
        depends[string_to_mask(lhs)] = depends[string_to_mask(lh
s)] | string_to_mask(rhs);
    }
    cout << "\n\nFinished taking inputs.........\n";
    cout << "Processing.........\n";
}
/* C+ = C; while (there is changes to C+)
    { do (for each functional dependency X-
>Y in F) { if (X⊆C+) then C+= C+∪Y } }
*/
int get_closures(int mask)
{
    int c = mask;
    int prevc = 0;
    while (c != prevc)
    {
        prevc = c;
        for (auto &ele : depends)
        {
            if ((c & ele.first) == ele.first)
                c |= ele.second;
        }
    }
    return c;
}
void get_closures_all()
{
    for (int i = 0; i < (1 << relationship_alpha_len); i++)
    {
        mapps[i] = get_closures(i);
    }
}
void get_keys()
{
    bool found = false;
    int total = (1 << relationship_alpha_len) - 1;
```

```
    for (int len = 1; len <= relationship_alpha_len; len++)
    {
        vector<int> perm(relationship_alpha_len, 0);
        for (int t = relationship_alpha_len - len; t < relations
hip_alpha_len; perm[t++] = 1)
            ;
        do
        {
            int mask = 0;
            for (int i = 0; i < relationship_alpha_len; i++)
                if (perm[i])
                    mask |= (1 << i);
            if (mapps[mask] == total)
            {
                found = true;
                ans.push_back(mask_to_string(mask));
            }
        } while (next_permutation(perm.begin(), perm.end()));
        if (found)
            break;
    }
    cout << "\nCandicate Keys are: \n";
    for (auto &cands : ans)
        cout << cands << " ";
}
int main()
{
    init();
    get_closures_all();
    get_keys();
    return 0;
}
```

Problem 1

1. Consider the relation scheme R = {E, F, G, H, I, J, K, L, M, N} and the set of functional dependencies {{E, F} -> {G}, {F} -> {I, J}, {E, H} -> {K, L}, K -> {M}, L -> {N}} on R. What is the key for R?



Problem 2

Consider a relation scheme R = (A, B, C, D, E, H) on which the following functional dependencies hold: {A–>B, BC–> D, E–>C, D–>A}. What are the candidate keys of R? write a c program to find it.

Ex. No.: 08 Date: 20-10-21 Week:8

Design a simple database for Online Shopping Cart using Python to access the back-end MySQL database. The online shopping cart must contain the following modules.

```
import mysql.connector
from datetime import date, datetime
mydb = mysql.connector.connect(host="localhost", user='root',
password='12345')

mycursor = mydb.cursor()
mycursor.execute("DROP DATABASE shopDB")
mycursor.execute("CREATE DATABASE shopDB")
mycursor.execute("USE shopDB")
mycursor.execute("Create table product(id int PRIMARY KEY, name
varchar(20),category varchar(20), quantity int, price float, discount
float, manf date,exp date)")


cont = "T"
opt = 0
while (cont == "t" or cont == "T"):
    opt = int(input("1.
insert\n2.find\n3.search_in_category\n4.update\n5.provide_discount\n6.delet
e\n7.notification\nEnter your choice: "))
    if (opt == 1):
        insert_record()
    elif(opt == 2):
        find_record()
    elif(opt == 3):
        search_category()
    elif(opt == 4):
        update_record()
    elif(opt == 5):
        provide_discount()
    elif(opt == 6):
```

```
        delete_record()
    elif(opt == 7):
        notify()
    else:
        print("invalid option")
    cont = input("do you want to continue(T/F): ")
```

1.The insert module must be able to accept the prod_id(primary key),product name, category, quantity, price, discount, date of manufacture and date of expiry and store it in the database.

```
def insert_record():
        prod_id = int(input("Enter id: "))
        name = input("Enter name: ")
        category = input("Enter category: ")
        quantity = int(input("Enter quantity: "))
        price = float(input("Enter price in RS: "))
        discount = float(input("Enter discount in %: "))
        inputDate = input("Enter the date in format 'dd/mm/yy' : ")
        day1, month1, year1 = (inputDate.split('/'))
        manf = date(int(year1), int(month1), int(day1))
        inputDate = input("Enter the date in format 'dd/mm/yy' : ")
        day, month, year = inputDate.split('/')
        exp = date(int(year), int(month), int(day))
        sql = "INSERT INTO product(id, name, category, quantity, price,
discount,manf, exp) values (%s, %s, %s, %s, %s, %s, %s, %s)"
        val = (prod_id, name, category, quantity, price, discount, manf,
exp)
        mycursor.execute(sql, val)
        mydb.commit()
        print("record inserted")
```

Input/ Output :

```
PS D: \Documents\NIT Trichy\V-Sem\DBMS-LAB\Lab-8 > conda activate base
PS D: \Documents\NIT Trichy\V-Sem\DBMS-LAB\Lab-8 > python .\shopping.py
1.insert
2.find
3.search_in_category
4.update
5.provide_discount
6.delete
7.notification
Enter your choice: 1
Enter id: 1
Enter name: mobile
Enter category: electronics
Enter quantity: 5
Enter price: 10000.0
Enter discount: 10.0
Enter the date in format dd/mm/yy: 12/06/2007
Enter the date in format dd/mm/yy: 30/07/2027
record inserted
do you want to continue(T/F): T
1.insert
2.find
3.search_in_category
4.update
5.provide_discount
6.delete
7.notification
Enter your choice: 1
Enter id: 3
Enter name: earphones
Enter category: accessories
Enter quantity: 10
Enter price: 2000.0
Enter discount: 5.0
Enter the date in format dd/mm/yy: 11/06/2012
Enter the date in format dd/mm/yy: 30/10/2015
record inserted
do you want to continue(T/F): T
1.insert
2.find
3.search_in_category
4.update
5.provide_discount
6.delete
7.notification
Enter your choice: 1
Enter id: 4
Enter name: router
Enter category: electronics
Enter quantity: 3
Enter price: 5000.0
Enter discount: 2.0
Enter the date in format dd/mm/yy: 03/10/2009
Enter the date in format dd/mm/yy: 03/10/2014
record inserted
do you want to continue(T/F): F
```

2.The find module must be able to accept the name of the product and display all the details of the product.

```
def find_record():
        name = input("Enter name: ")
        sql = f'SELECT * FROM product WHERE name = "{name}"'
        mycursor.execute(sql)
        myresult = mycursor.fetchall()
        if (len(myresult) == 0):
            print("no records")
        else:
            for x in myresult:
                print(x)
```

Input/Output :

```
PS D:\Documents\NIT Trichy\V-Sem\DBMS-LAB\Lab-8> conda activate base
PS D:\Documents\NIT Trichy\V-Sem\DBMS-LAB\Lab-8> python .\shopping.py
1.insert
2.find
3.search_in_category
4.update
5.provide_discount
6.delete
7.notification
Enter your choice: 2
Enter name: router
(4, 'router', 'electronics', 3, 5000.0, 2.0,
datetime.date(2009, 10, 03), datetime.date(2014, 10, 03))
do you want to continue(T/F): F
```

3.The search_in_category module must be able to accept the name of the category and display all the details of the products in the categories.

```
def search_category():
        category = input("Enter category: ")
        sql = f'SELECT * FROM product WHERE category = "{category}"'
        # que = (category)
        print(sql)
        mycursor.execute(sql)
        myresult = mycursor.fetchall()
        if (len(myresult) == 0):
            print("no records")
        else:
            for x in myresult:
                print(x)
```

Input/Output :

```
PS D:\Documents\NIT Trichy\V-Sem\DBMS-LAB\Lab-8> conda activate base
PS D:\Documents\NIT Trichy\V-Sem\DBMS-LAB\Lab-8> python .\shopping.py
1.insert
2.find
3.search_in_category
4.update
5.provide_discount
6.delete
7.notification
Enter your choice: 3
Enter category: electronics
(1, 'mobile', 'electronics', 5, 10000.0, 10.0,
datetime.date(2007, 06, 12), datetime.date(2027, 07, 30))
(4, 'router', 'electronics', 3, 5000.0, 2.0,
datetime.date(2009, 10, 03), datetime.date(2014, 10, 03))
do you want to continue(T/F): F
```

4.The update module must be able to update the price of the products.

```
def update_record():
        prod_id = int(input("Enter id: "))
        price = float(input("Enter price: "))
        sql = f'UPDATE product SET price = "{price}" WHERE id =
"{prod_id}"'
        # val = (price, prod_id)
        mycursor.execute(sql)
        mydb.commit()
        print("record updated")
```

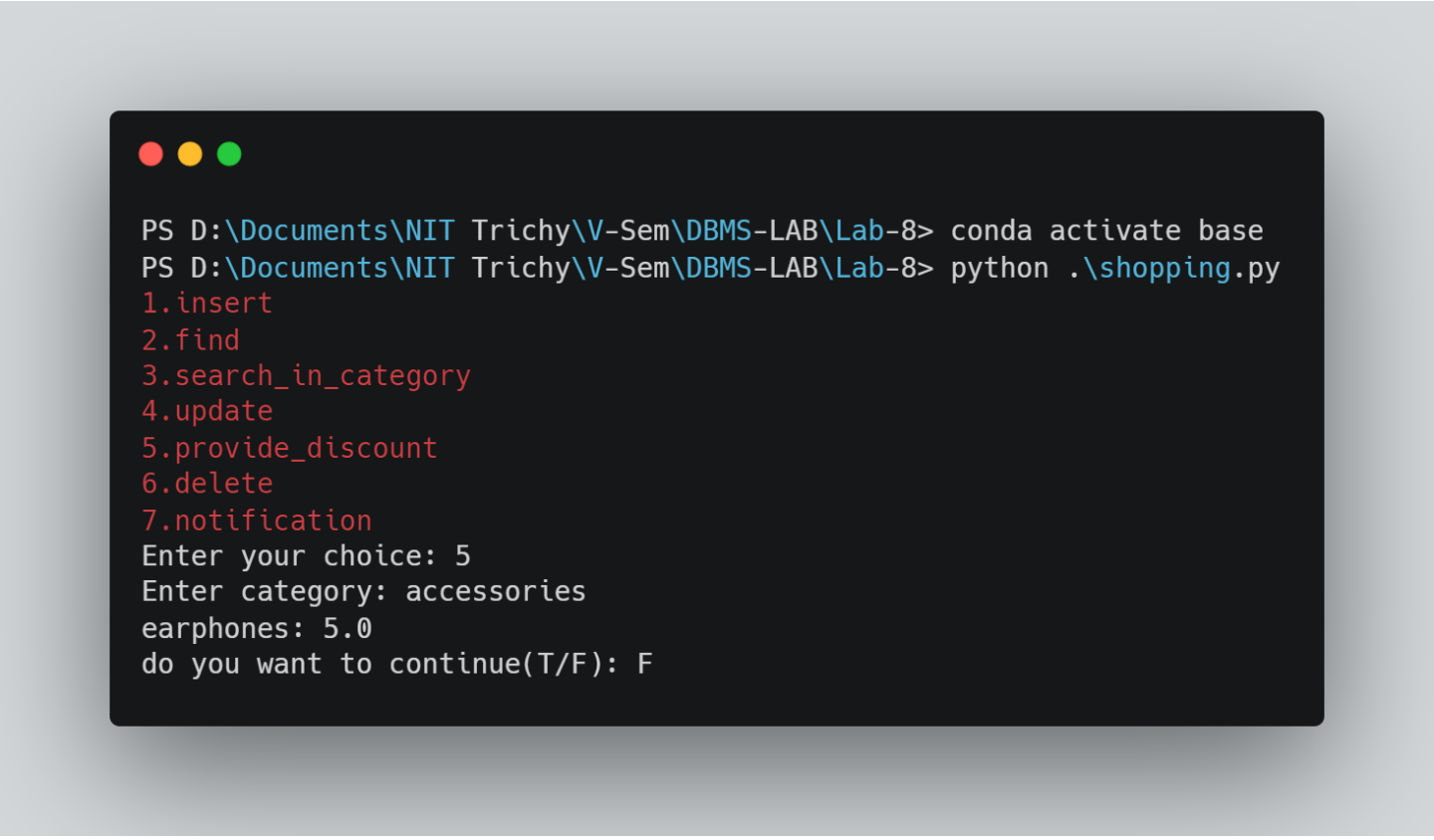
Input/Output :

```
PS D:\Documents\NIT Trichy\V-Sem\DBMS-LAB\Lab-8> conda activate base
PS D:\Documents\NIT Trichy\V-Sem\DBMS-LAB\Lab-8> python .\shopping.py
1.insert
2.find
3.search_in_category
4.update
5.provide_discount
6.delete
7.notification
Enter your choice: 4
Enter id: 1
Enter price: 15000.0
record updated
do you want to continue(T/F): F
```

5.The provide_discount module must be able to provide the discount of the products in a particular category.

```
def provide_discount():
        category = input("Enter category: ")
        sql = f'SELECT name, discount FROM product WHERE category =
"{category}"'
        # que = (category)
        mycursor.execute(sql)
        myresult = mycursor.fetchall()
        if (len(myresult) == 0):
            print("no records")
        else:
            for x in myresult:
                print(x[0]+' : %d'%x[1])
```

Input/Output :

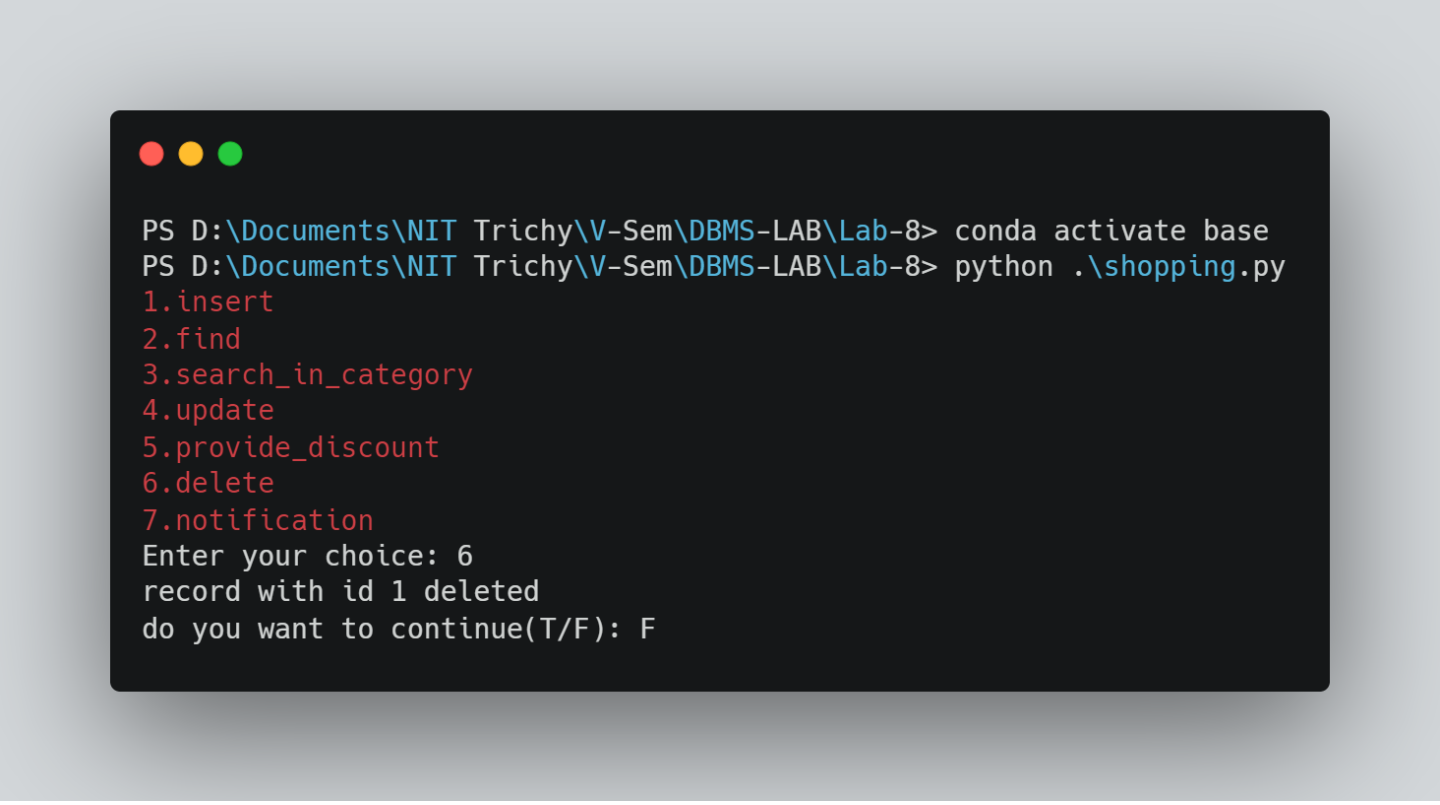


6.The delete_module must be able to delete/ cancel the product if the expiry date is not valid.

```
def delete_record():
        mycursor.execute('SELECT * FROM product')
        myresult = mycursor.fetchall()
        if (len(myresult) == 0):
            print("no records")
        else:
            for x in myresult:
                if (x[7] < datetime.now().date()):
                    print("record with id %d deleted" % x[0])
        sql = f"DELETE FROM product WHERE id = {x[0]}"
        mycursor.execute(sql)
```

```
        mydb.commit()
```

Input/Output :



```
PS D:\Documents\NIT Trichy\V-Sem\DBMS-LAB\Lab-8> conda activate base
PS D:\Documents\NIT Trichy\V-Sem\DBMS-LAB\Lab-8> python .\shopping.py
1.insert
2.find
3.search_in_category
4.update
5.provide_discount
6.delete
7.notification
Enter your choice: 6
record with id 1 deleted
do you want to continue(T/F): F
```

7.The notification_module must indicate free shipping if the total product price exceeds Rs.1000.

```
def notify():
        prod_id = int(input("Enter id: "))
        sql = f'SELECT price, discount FROM product WHERE id = {prod_id}'
        que = (prod_id)
        mycursor.execute(sql, que)
        myresult = mycursor.fetchall()
        if (len(myresult) == 0):
            print("no records")
        else:
             s = 0
             for x in myresult:
                s += x[0]
```

```
            s -= (x[0]*x[1]*0.01)
        if (s > 1000):
          print("Free shipping")
        else:
          print("no free shipping")
```

Input/Output :

```
PS D:\Documents\NIT Trichy\V-Sem\DBMS-LAB\Lab-8> conda activate base
PS D:\Documents\NIT Trichy\V-Sem\DBMS-LAB\Lab-8> python .\shopping.py
1.insert
2.find
3.search_in_category
4.update
5.provide_discount
6.delete
7.notification
Enter your choice: 7
Enter id: 3
Free shipping
do you want to continue(T/F): F
```

1)

a. Demonstrating DTD:

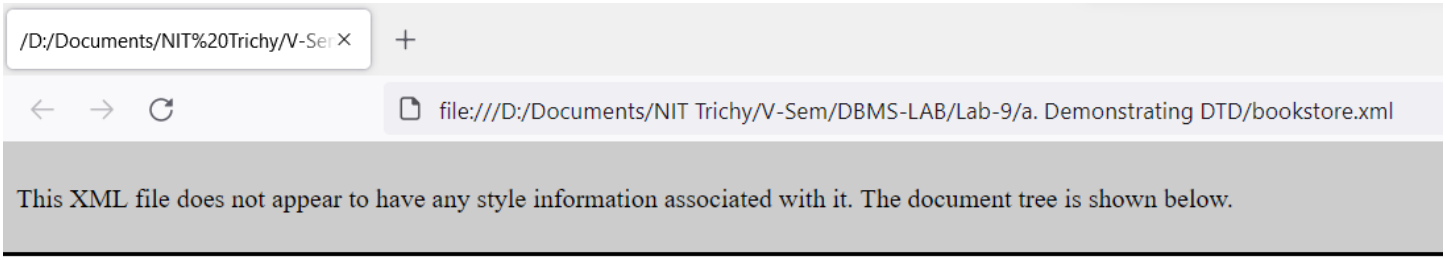
1. filename : *bookstore.dtd*

```
<!ELEMENT bookstore (book+)>
<!ELEMENT book (title,author,category,isbn,publisher,edition,price)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT category (#PCDATA)>
<!ELEMENT isbn (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT edition (#PCDATA)>
<!ELEMENT price (#PCDATA)>
```

2. filename : *bookstore.xml*

```
<?xml version="1.0"?>
<!DOCTYPE bookstore SYSTEM "bookstore.dtd">
    <bookstore>
            <book>
                <title>XML Developer's Guide</title>
                <author>Gambardella, Matthew</author>
                <category>textbook</category>
                <isbn>111111</isbn>
                <publisher>wiley</publisher>
```

```
            <edition>second</edition>
            <price>20.00</price>
        </book>
        <book>
            <title>Maeve Ascendant</title>
            <author>Corets, Eva</author>
            <category>textbook</category>
            <isbn>222222</isbn>
            <publisher>tata</publisher>
            <edition>second</edition>
            <price>60.00</price>
        </book>
        <book>
            <title>Midnight Rain</title>
            <author>Ralls, Kim</author>
            <category>fiction</category>
            <isbn>333333</isbn>
            <publisher>O'relilly</publisher>
            <edition>second</edition>
            <price>100.00</price>
        </book>
</bookstore>
```




```
−<bookstore>
  −<book>
      <title>XML Developer's Guide</title>
      <author>Gambardella, Matthew</author>
      <category>textbook</category>
      <isbn>111111</isbn>
      <publisher>wiley</publisher>
      <edition>second</edition>
      <price>20.00</price>
    </book>
  −<book>
      <title>Maeve Ascendant</title>
      <author>Corets, Eva</author>
      <category>textbook</category>
      <isbn>222222</isbn>
      <publisher>tata</publisher>
      <edition>second</edition>
      <price>60.00</price>
    </book>
  −<book>
      <title>Midnight Rain</title>
      <author>Ralls, Kim</author>
      <category>fiction</category>
      <isbn>333333</isbn>
      <publisher>O'relilly</publisher>
      <edition>second</edition>
      <price>100.00</price>
    </book>
</bookstore>
```

b. Demonstrating XSD:

1. filename : *bookstore.xml*

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="bookstore">
        <xs:complexType>
```

```
            <xs:sequence>
               <xs:element name="book" minOccurs="1" maxOccurs="unbounded" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="book">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="title" minOccurs="1" maxOccurs="1" />
                <xs:element ref="author" minOccurs="1" maxOccurs="1" />
                <xs:element ref="category" minOccurs="1" maxOccurs="1" />
                <xs:element ref="isbn" minOccurs="1" maxOccurs="1" />
                <xs:element ref="publisher" minOccurs="1" maxOccurs="1" />
                <xs:element ref="edition" minOccurs="1" maxOccurs="1" />
                <xs:element ref="price" minOccurs="1" maxOccurs="1" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="title" type="xs:string" />
    <xs:element name="author" type="xs:string" />
    <xs:element name="category" type="xs:string" />
    <xs:element name="isbn" type="xs:string" />
    <xs:element name="publisher" type="xs:string" />
    <xs:element name="edition" type="xs:string" />
    <xs:element name="price" type="xs:string" />
</xs:schema>
```

2. filename : *bookstore.css*

```
bookstore {
  color: orange;
}
book {
  color: Red;
}
title {
  color: blue;
  font-weight: bold;
  margin-left: 10pt;
  display: block;
}
author {
  color: red;
  font-weight: bold;
  margin-left: 10pt;
}
category {
  color:grey;
  font-weight: bold;
  margin-left: 10pt;
}
isbn {
  color: green;
  font-weight: bold;
  margin-left: 10pt;
}
edition {
  color: red;
  font-weight: bold;
  margin-left: 10pt;
```

```
}
publisher {
  color: green;
  font-weight: bold;
  margin-left: 10pt;
}
```

3. filename : *bookstore.xsl*

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="bookstore.css"?>
<bookstore xmlns:xsi="http:www.w3.org/2001/xmlschema-instance"
xsi:noNamespaceSchemaLocation="bookstore.xsd">
    <book>
        <title>XML Developer's Guide</title>
        <author>Gambardella, Matthew</author>
        <category>textbook</category>
        <isbn>111111</isbn>
        <publisher>wiley</publisher>
        <edition>second</edition>
        <price>20.00</price>
    </book>
    <book>
        <title>Maeve Ascendant</title>
        <author>Corets, Eva</author>
        <category>textbook</category>
        <isbn>222222</isbn>
        <publisher>tata</publisher>
        <edition>second</edition>
        <price>60.00</price>
    </book>
    <book>
```

```
        <title>Midnight Rain</title>
        <author>Ralls, Kim</author>
        <category>fiction</category>
        <isbn>333333</isbn>
        <publisher>O'relilly</publisher>
        <edition>second</edition>
        <price>100.00</price>
    </book>
</bookstore>
```

c. Demonstrating XSL:

1. filename : *bookstore.xsl*

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
```

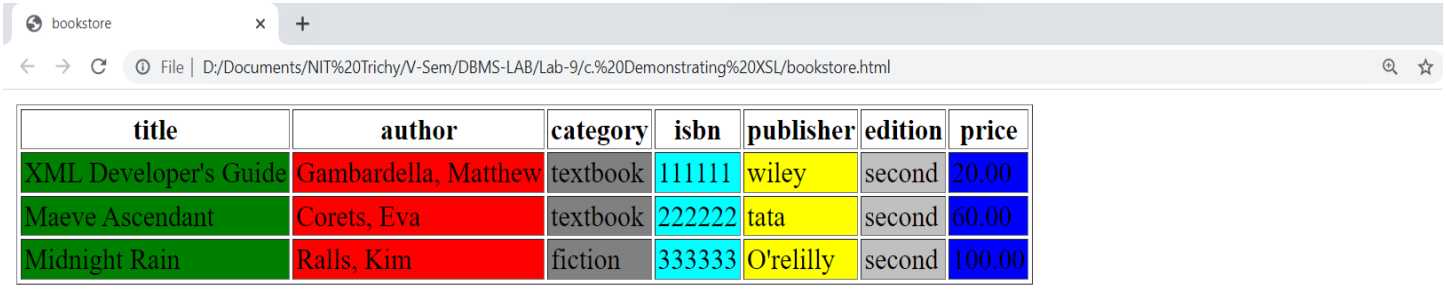
```
<html>
    <head>
        <title>bookstore</title>
    </head>
    <body>
        <table border="1">
            <tr>
                <th>title</th>
                <th>author</th>
                <th>category</th>
                <th>isbn</th>
                <th>publisher</th>
                <th>edition</th>
                <th>price</th>
            </tr>
            <xsl:for-each select="/bookstore/book">
                <tr>
                    <td bgcolor="green">
                        <xsl:value-of select="title" />
                    </td>
                    <td bgcolor="red">
                        <xsl:value-of select="author" />
                    </td>
                    <td bgcolor="grey">
                        <xsl:value-of select="category" />
                    </td>
                    <td bgcolor="cyan">
                        <xsl:value-of select="isbn" />
                    </td>
                    <td bgcolor="yellow">
                        <xsl:value-of select="publisher" />
                    </td>
```

```
                        <td bgcolor="silver">
                            <xsl:value-of select="edition" />
                        </td>
                        <td bgcolor="blue">
                            <xsl:value-of select="price" />
                        </td>
                    </tr>
                </xsl:for-each>
            </table>
        </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

2. filename : *bookstore.xml*

```
<?xml-stylesheet type="text/xsl"  href="bookstore.xsl"?>
<bookstore>
    <book>
        <title>XML Developer's Guide</title>
        <author>Gambardella, Matthew</author>
        <category>textbook</category>
        <isbn>111111</isbn>
        <publisher>wiley</publisher>
        <edition>second</edition>
        <price>20.00</price>
    </book>
    <book>
        <title>Maeve Ascendant</title>
        <author>Corets, Eva</author>
        <category>textbook</category>
        <isbn>222222</isbn>
```

```
        <publisher>tata</publisher>
        <edition>second</edition>
        <price>60.00</price>
    </book>
    <book>
        <title>Midnight Rain</title>
        <author>Ralls, Kim</author>
        <category>fiction</category>
        <isbn>333333</isbn>
        <publisher>O'relilly</publisher>
        <edition>second</edition>
        <price>100.00</price>
    </book>
</bookstore>
```

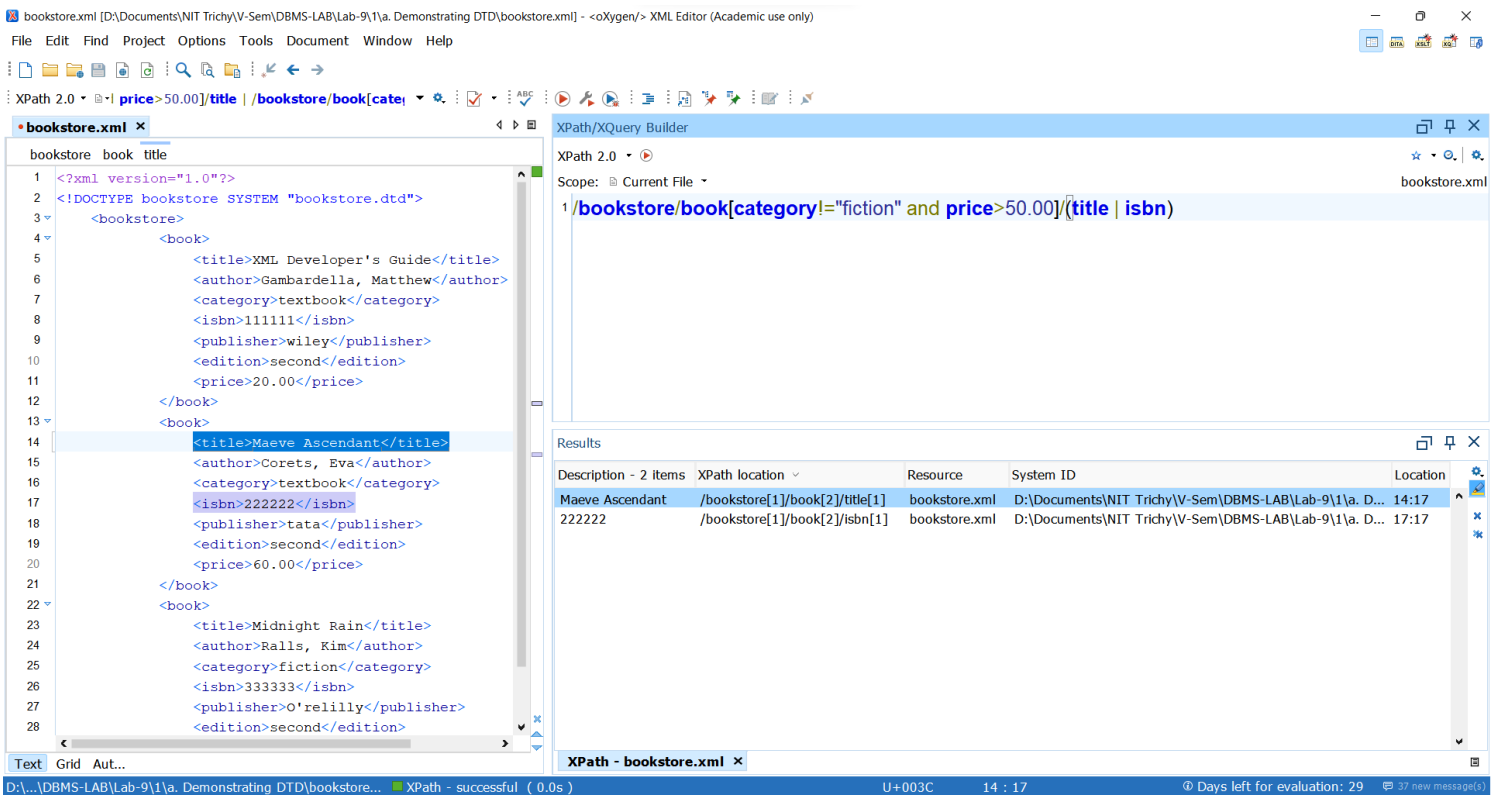


title	author	category	isbn	publisher	edition	price
XML Developer's Guide	Gambardella, Matthew	textbook	111111	wiley	second	20.00
Maeve Ascendant	Corets, Eva	textbook	222222	tata	second	60.00
Midnight Rain	Ralls, Kim	fiction	333333	O'relilly	second	100.00

2) Perform the following queries using XPath :

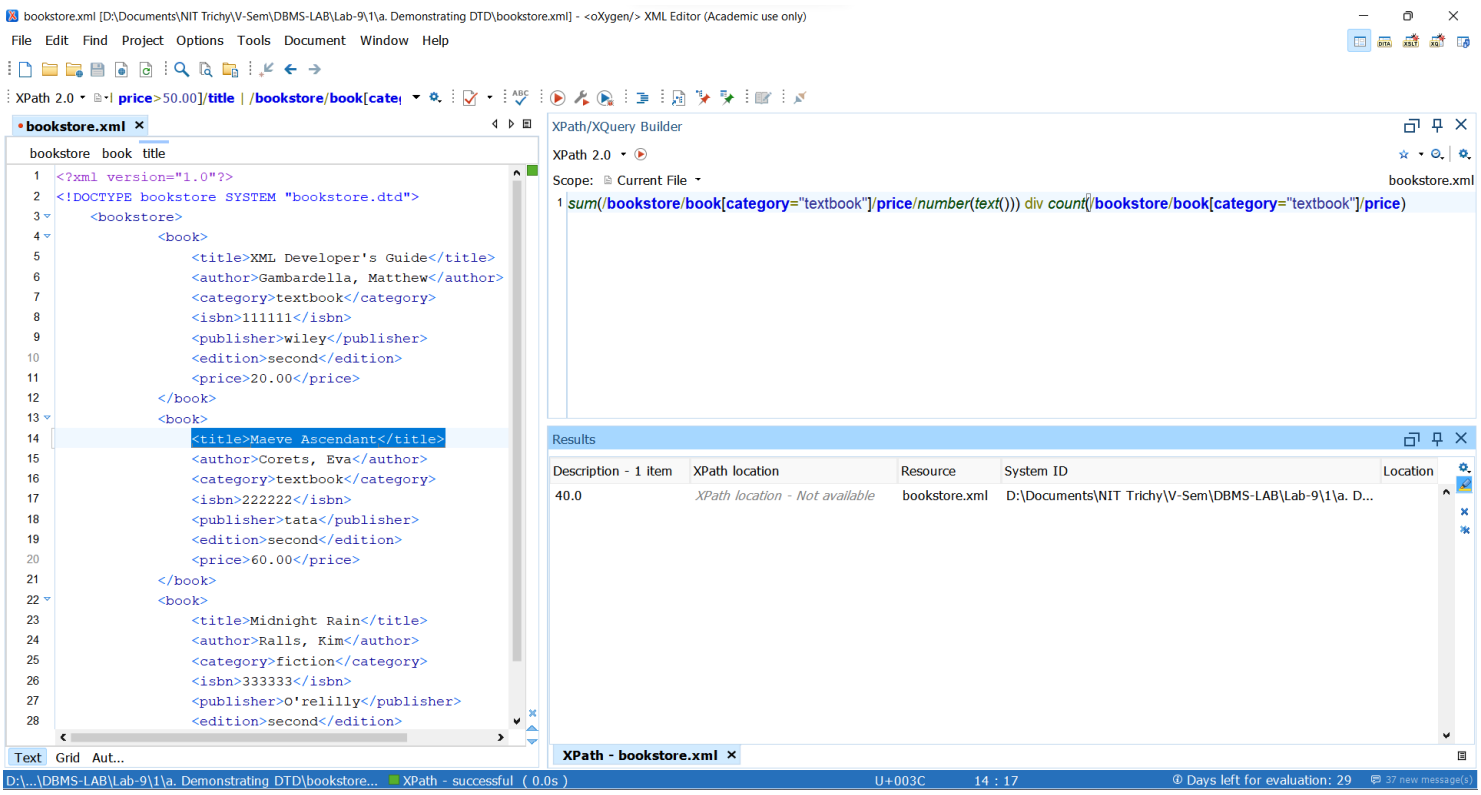
(i) Find the title and price of non-fiction books with a price more than 50 USD

Xpath query : /**bookstore**/**book**[**category**!="fiction" and **price**>50.00]/(**title** | **isbn**)



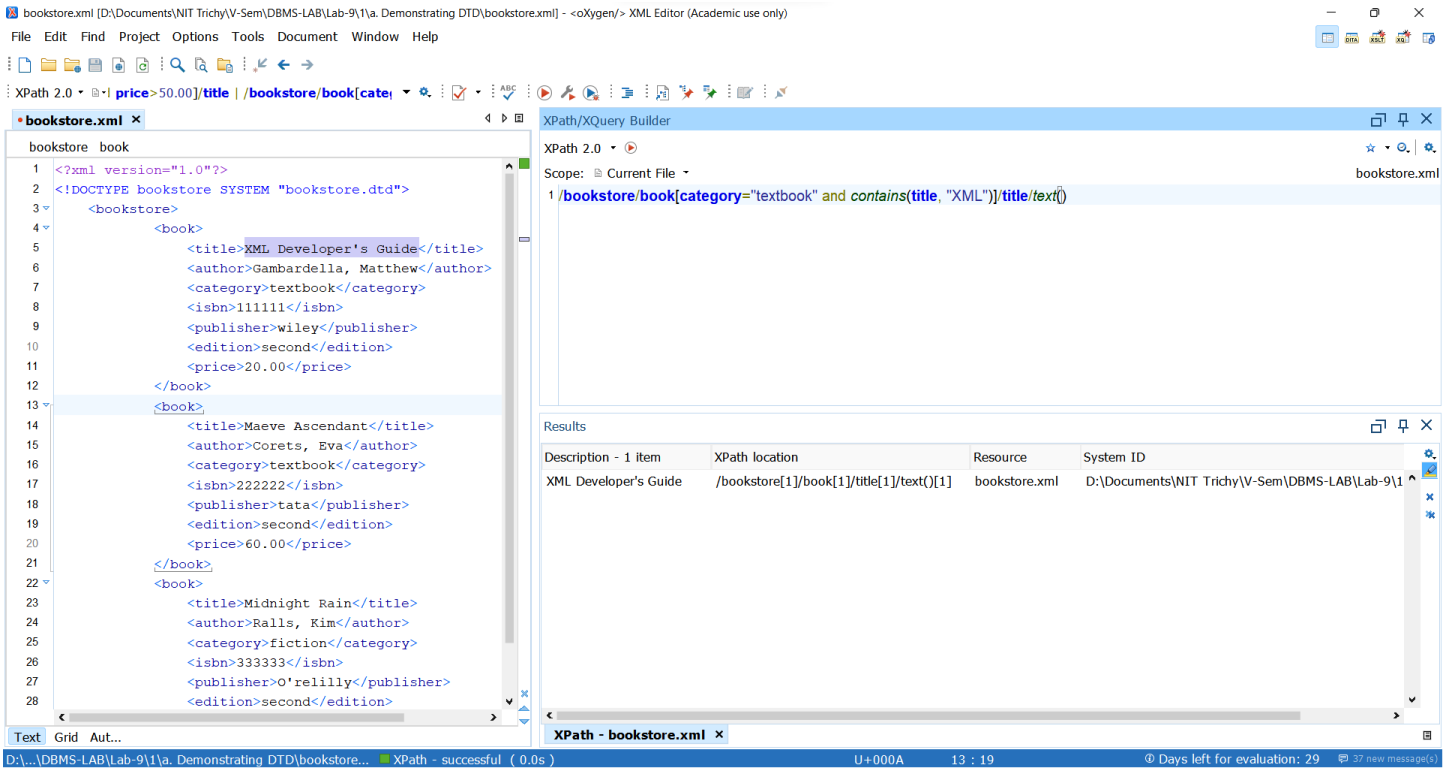
(ii) Find average price of textbooks

Xpath query : *sum*(/**bookstore**/**book**[**category**="textbook"]/**price**/*number*(*text*())) div *count*(/**bookstore**/**book**[**category**="textbook"]/**price**)



(iii) Find the titles of textbooks on XML

Xpath query : /**bookstore**/**book**[**category**="textbook" and *contains*(**title**, "XML")]/**title**/*text*()

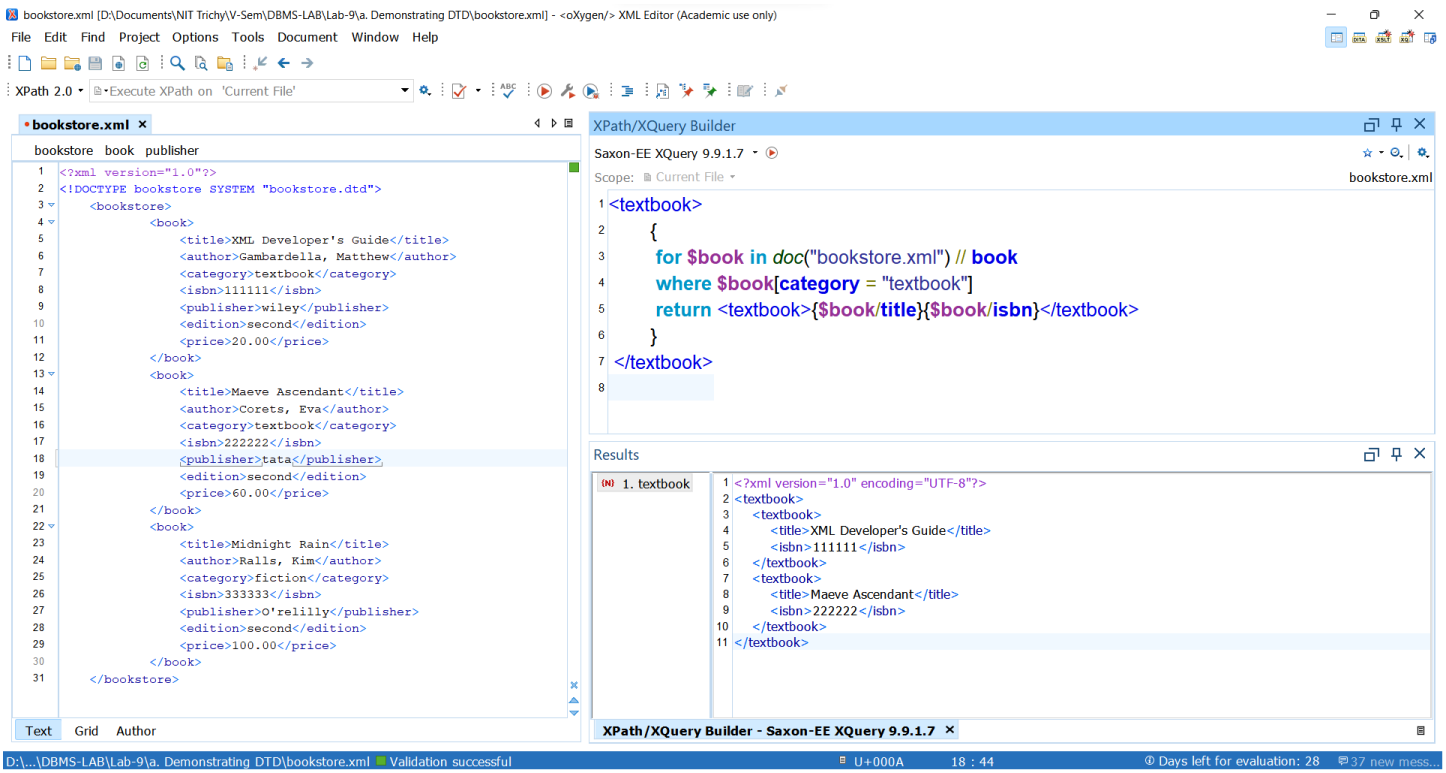


1) Perform the following queries using XQuery:

(i) Create a new document which contain only the isbn and title of textbooks

XQuery

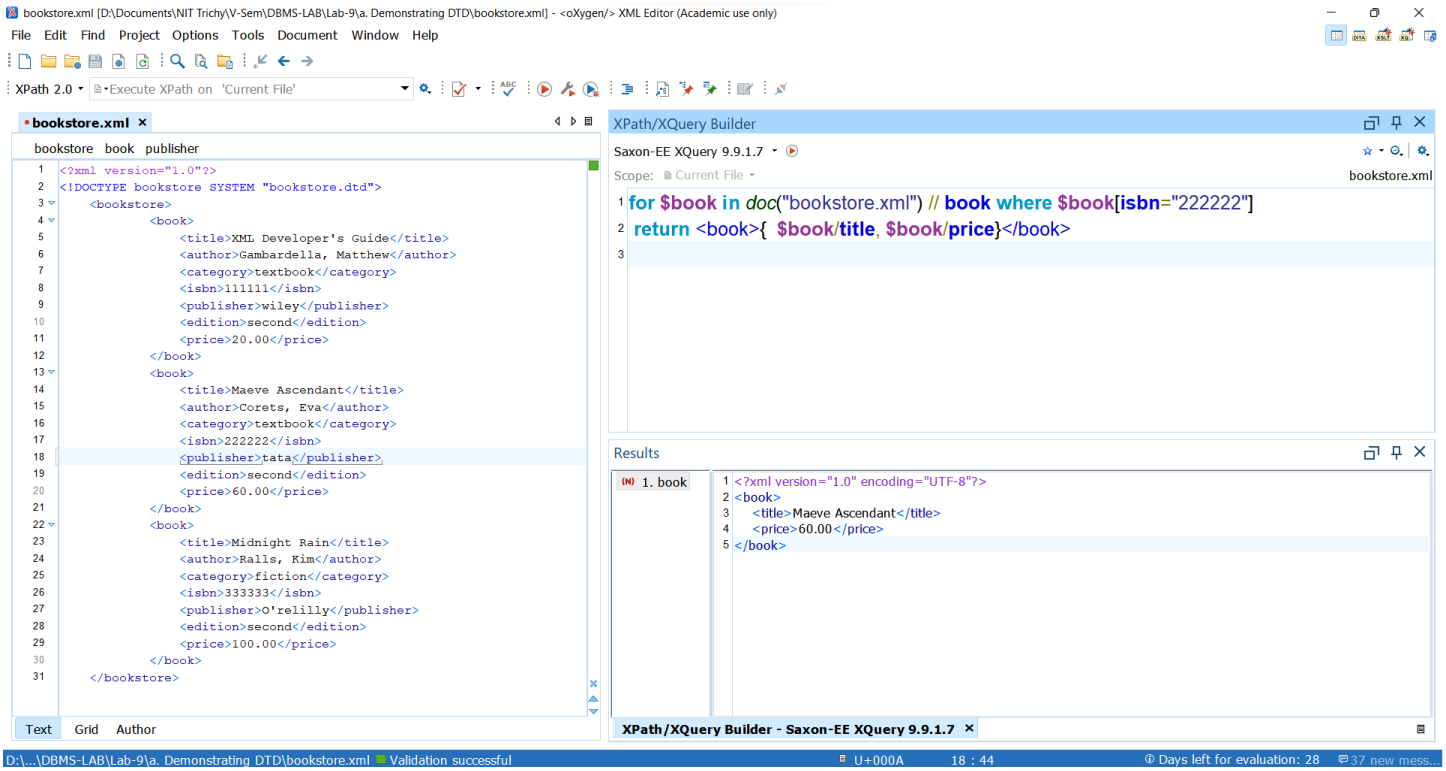
```
<textbook>
    {
     for $book in doc("bookstore.xml") // book
     where $book[category = "textbook"]
     return <textbook>{$book/title}{$book/isbn}</textbook>
     }
</textbook>
```



(ii) Find the title and price of the book with isbn “222222”.

XQuery :

```
for $book in doc("bookstore.xml") // book where $book[isbn="222222"]
 return <book>{  $book/title, $book/price}</book>
```

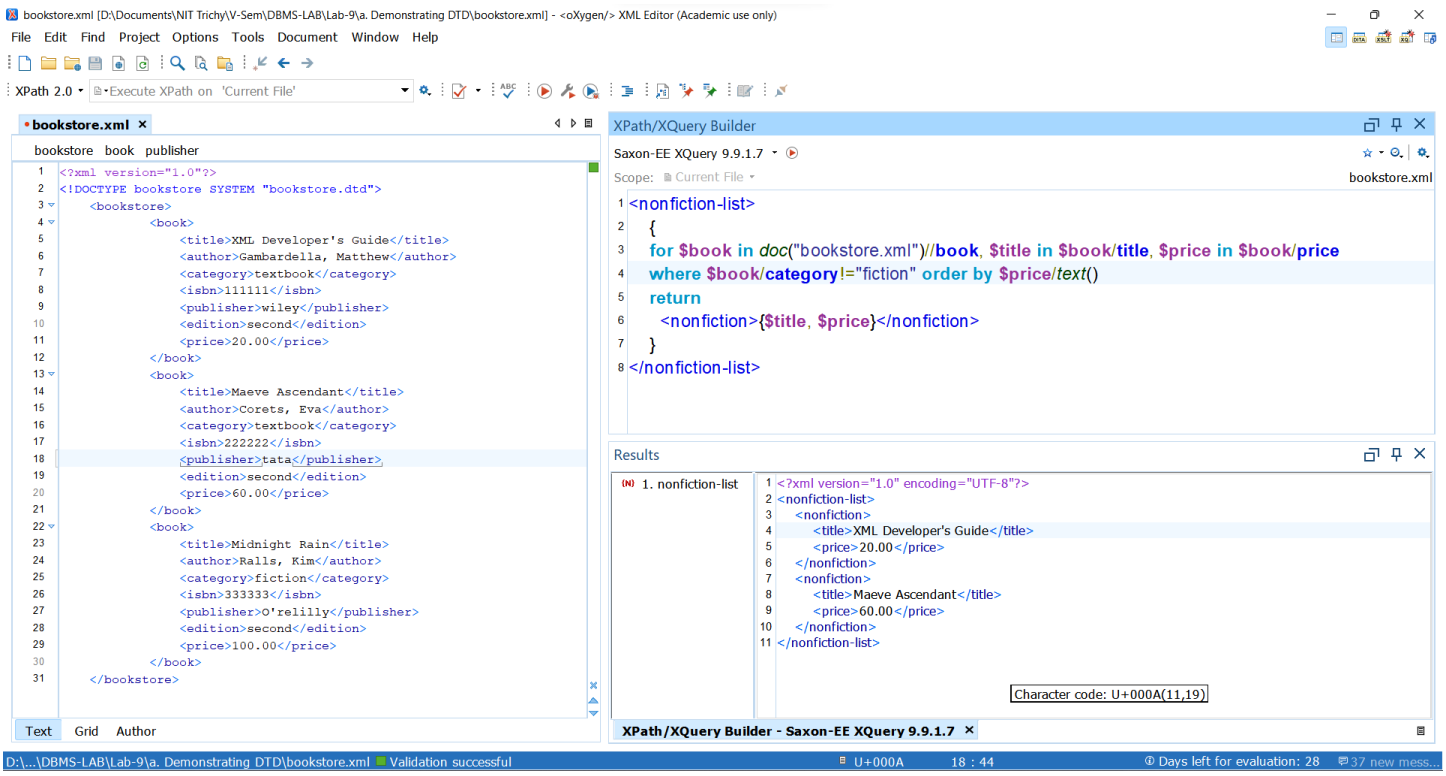


(iii) Produce a list of non-fictions with their title and price, sorted by price

XQuery :

```
<nonfiction-list>
   {
   for $book in doc("bookstore.xml")//book, $title in $book/title, $price in $book/price
```

```
    where $book/category!="fiction" order by $price/text()
    return
      <nonfiction>{$title, $price}</nonfiction>
    }
</nonfiction-list>
```

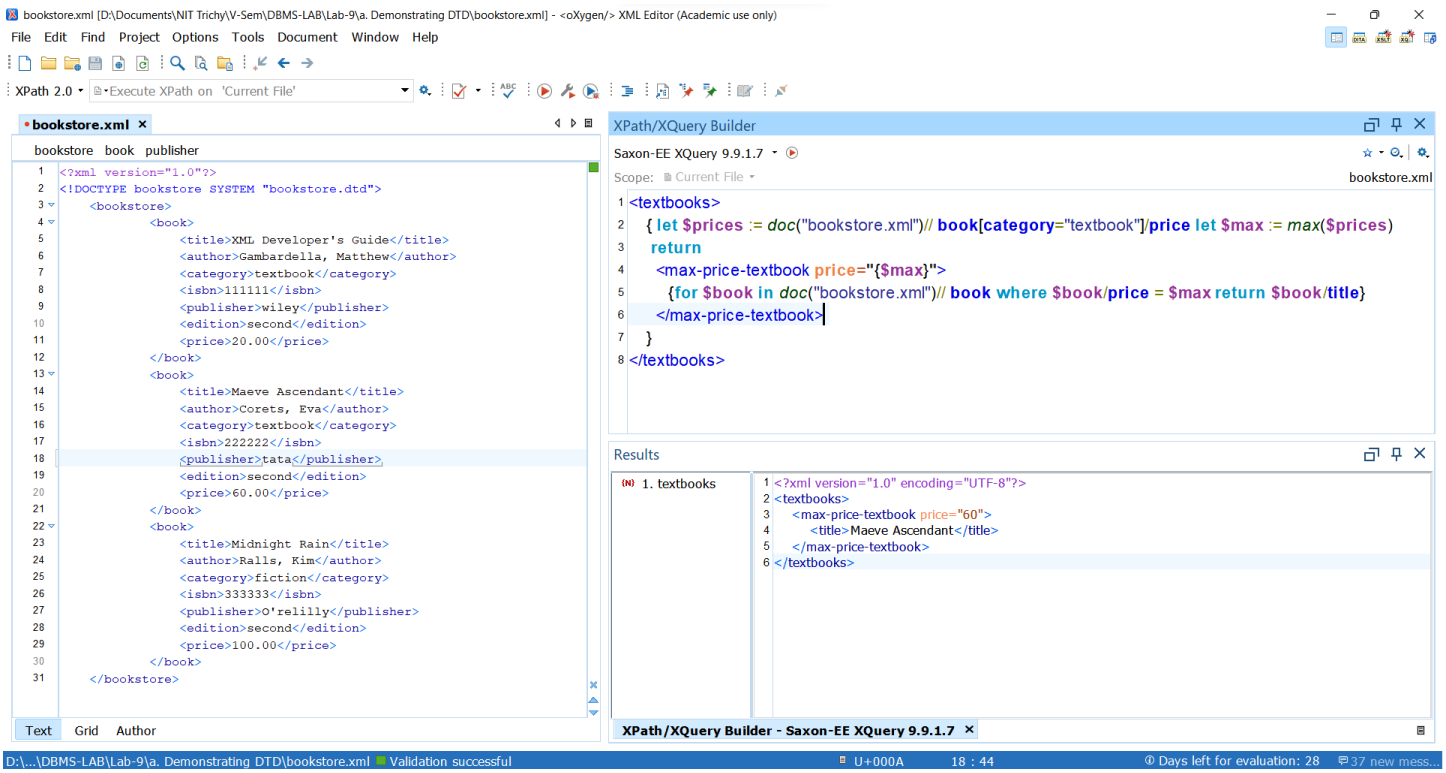


(iv) Find title of the textbook with highest price

XQuery :

```
<textbooks>
  { let $prices := doc("bookstore.xml")// book[category="textbook"]/price let $max := max($prices)
```

```
 return
<max-price-textbook price="{$max}">
  {for $book in doc("bookstore.xml")// book where $book/price = $max return $book/title}
</max-price-textbook>
}
</textbooks>
```

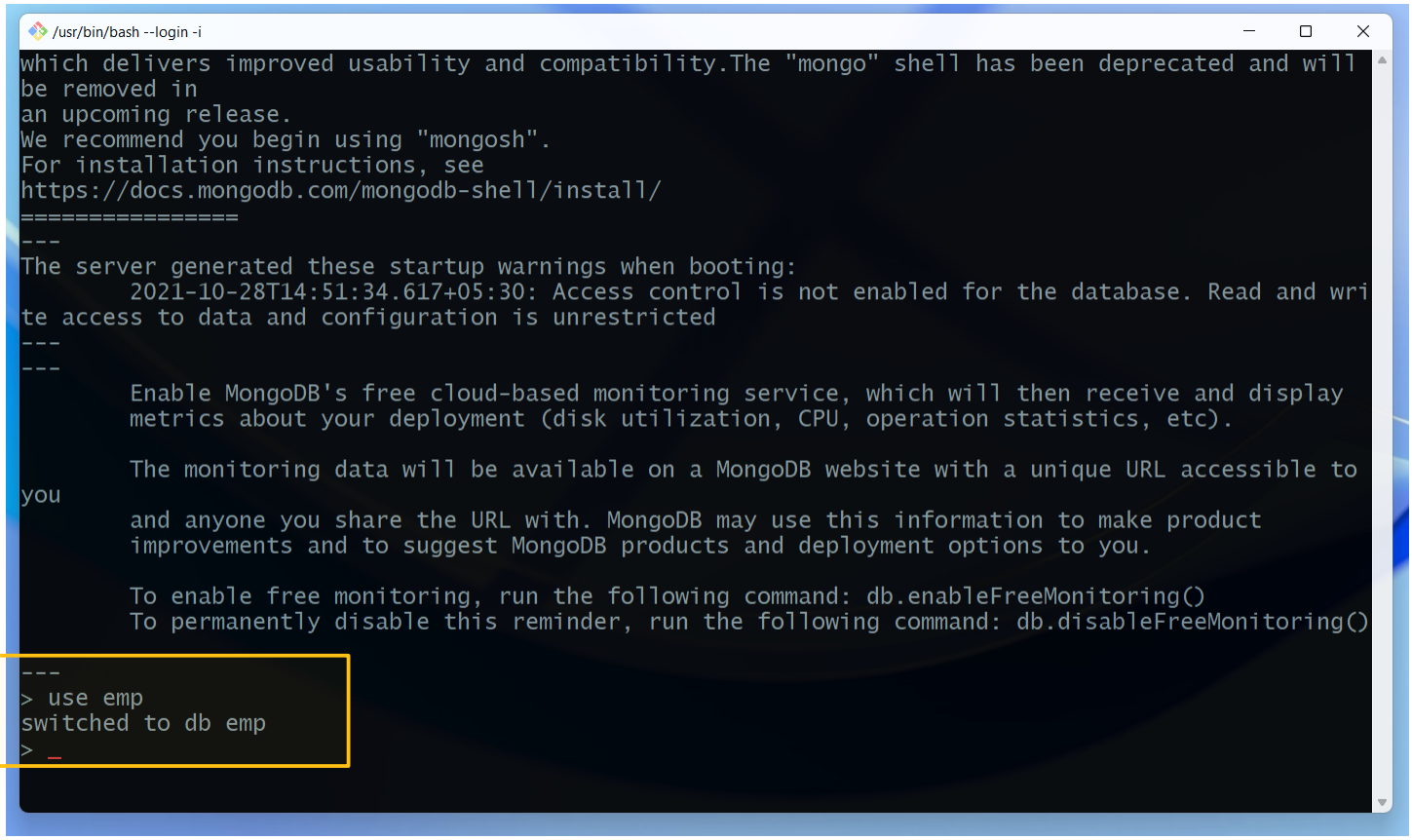


Ex. No.: 10 Date: 28-10-21 Week:10

-- Create database EMP and Make Collection With name "EMPL" and Follow Queries

--Created Database

```
use emp
switched to DB emp
```



Database created :

```
> show dbs
admin   0.000GB
config  0.000GB
emp     0.000GB
local   0.000GB
> _
```

```
--Create Collection
 db.createCollection("empl")
```

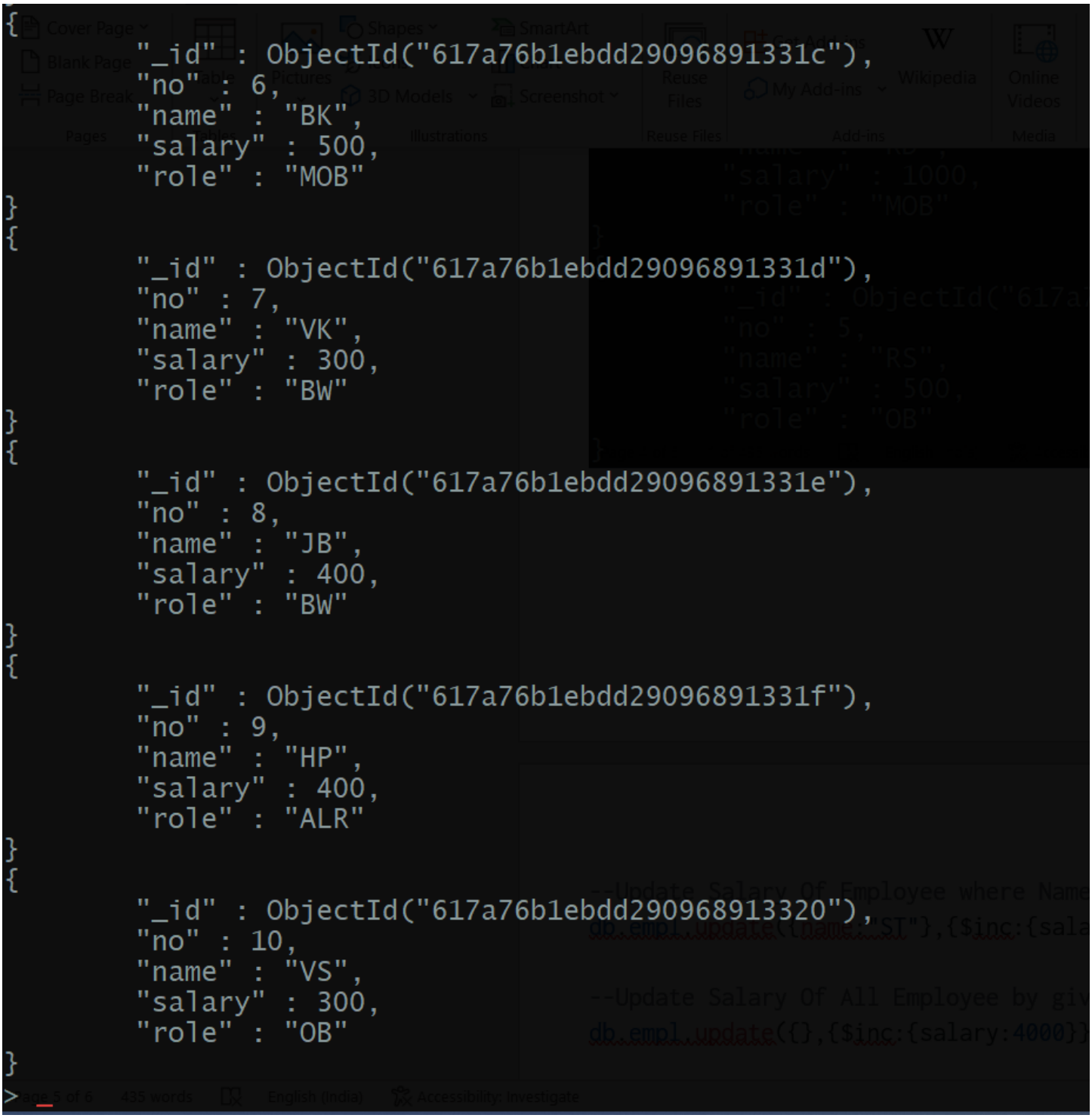
```
> db.createCollection("empl")
{ "ok" : 1 }
> _
```

```
--Insert Records Into EMPL Collection
 db.empl.insert([
 {no:1,name:"ST",salary:2000,role:"OB"},
 {no:2,name:"MSD",salary:1500,role:"WK"},
 {no:3,name:"YS",salary:1000,role:"ALR"},
 {no:4,name:"RD",salary:1000,role:"MOB"},
 {no:5,name:"RS",salary:500,role:"OB"},
 {no:6,name:"BK",salary:500,role:"MOB"},
 {no:7,name:"VK",salary:300,role:"BW"},
 {no:8,name:"JB",salary:400,role:"BW"},
 {no:9,name:"HP",salary:400,role:"ALR"},
 {no:10,name:"VS",salary:300,role:"OB"}])
```

```
> db.empl.insert([
...  {no:1,name:"ST",salary:2000,role:"OB"},
...  {no:2,name:"MSD",salary:1500,role:"WK"},
...  {no:3,name:"YS",salary:1000,role:"ALR"},
...  {no:4,name:"RD",salary:1000,role:"MOB"},
...  {no:5,name:"RS",salary:500,role:"OB"},
...  {no:6,name:"BK",salary:500,role:"MOB"},
...  {no:7,name:"VK",salary:300,role:"BW"},
...  {no:8,name:"JB",salary:400,role:"BW"},
...  {no:9,name:"HP",salary:400,role:"ALR"},
...  {no:10,name:"VS",salary:300,role:"OB"}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 10,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
```

```
--Display Data in Proper Format
db.empl.find().pretty()
```

```
> db.empl.find().pretty()
{
        "_id" : ObjectId("617a76b1ebdd290968913317"),
        "no" : 1,
        "name" : "ST",
        "salary" : 2000,
        "role" : "OB"
}
{
        "_id" : ObjectId("617a76b1ebdd290968913318"),
        "no" : 2,
        "name" : "MSD",
        "salary" : 1500,
        "role" : "WK"
}
{
        "_id" : ObjectId("617a76b1ebdd290968913319"),
        "no" : 3,
        "name" : "YS",
        "salary" : 1000,
        "role" : "ALR"
}
{
        "_id" : ObjectId("617a76b1ebdd29096891331a"),
        "no" : 4,
        "name" : "RD",
        "salary" : 1000,
        "role" : "MOB"
}
{
        "_id" : ObjectId("617a76b1ebdd29096891331b"),
        "no" : 5,
        "name" : "RS",
        "salary" : 500,
        "role" : "OB"
}
```




```
--Update Salary Of Employee where Name is "ST" by +8000
```

```
db.empl.update({name:"ST"},{$inc:{salary:8000}})
```

```
> db.empl.update({name:"ST"},{$inc:{salary:8000}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

```
--Update Salary Of All Employee by giving an increment of +4000 each
db.empl.update({},{$inc:{salary:4000}},{multi:true})
```

```
> db.empl.update({},{$inc:{salary:4000}},{multi:true})
WriteResult({ "nMatched" : 10, "nUpserted" : 0, "nModified" : 10 })
>
```

```
--update role of "MSD" as "C and WK"
db.empl.update({name:"MSD"},{$set:{role:"c and WK"}})
```

```
> db.empl.update({name:"MSD"},{$set:{role:"c and WK"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

```
--Add a New Field remark to document with name "RS" set Remark as WC
db.empl.update({name:"RS"},{$set:{remark:"WC"}})
```

```
> db.empl.update({name:"RS"},{$set:{remark:"WC"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

```
--Add a New Field As Number 11,name AK,Salary 10000,role coach without using
insert statement. But for Doing So You should have a Record Added with number
11.
db.empl.update({no:11},{$set:{no:11,name:"AK",salary:10000,role:"coach"}},{upser
t:true})
```

```
> db.empl.update({no:11},{$set:{no:11,name:"AK",salary:10000,role:"coch"}},{upsert:true})
WriteResult({
        "nMatched" : 0,
        "nUpserted" : 1,
        "nModified" : 0,
        "_id" : ObjectId("617a7867eb8bec42355c9b0d")
})
>
```

```
--remove added New Field
db.empl.update({name:"RS"},{$unset:{remark:"WC"}})
```

```
> db.empl.update({name:"RS"},{$unset:{remark:"WC"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> _
```

```
--Update the Field "RD" by multiplying with salary by 2
db.empl.update({name:"RD"},{$mul:{salary:2}})
```

```
> db.empl.update({name:"RD"},{$mul:{salary:2}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> _
```

```
--To Find Document From the empl collection where name begins with S
db.empl.find({name:/^S/})
```

```
> db.empl.find({name:/^S/})
{ "_id" : ObjectId("617a76b1ebdd290968913317"), "no" : 1, "name" : "ST", "salary" : 14000, "role" :
 "OB" }
> _
```

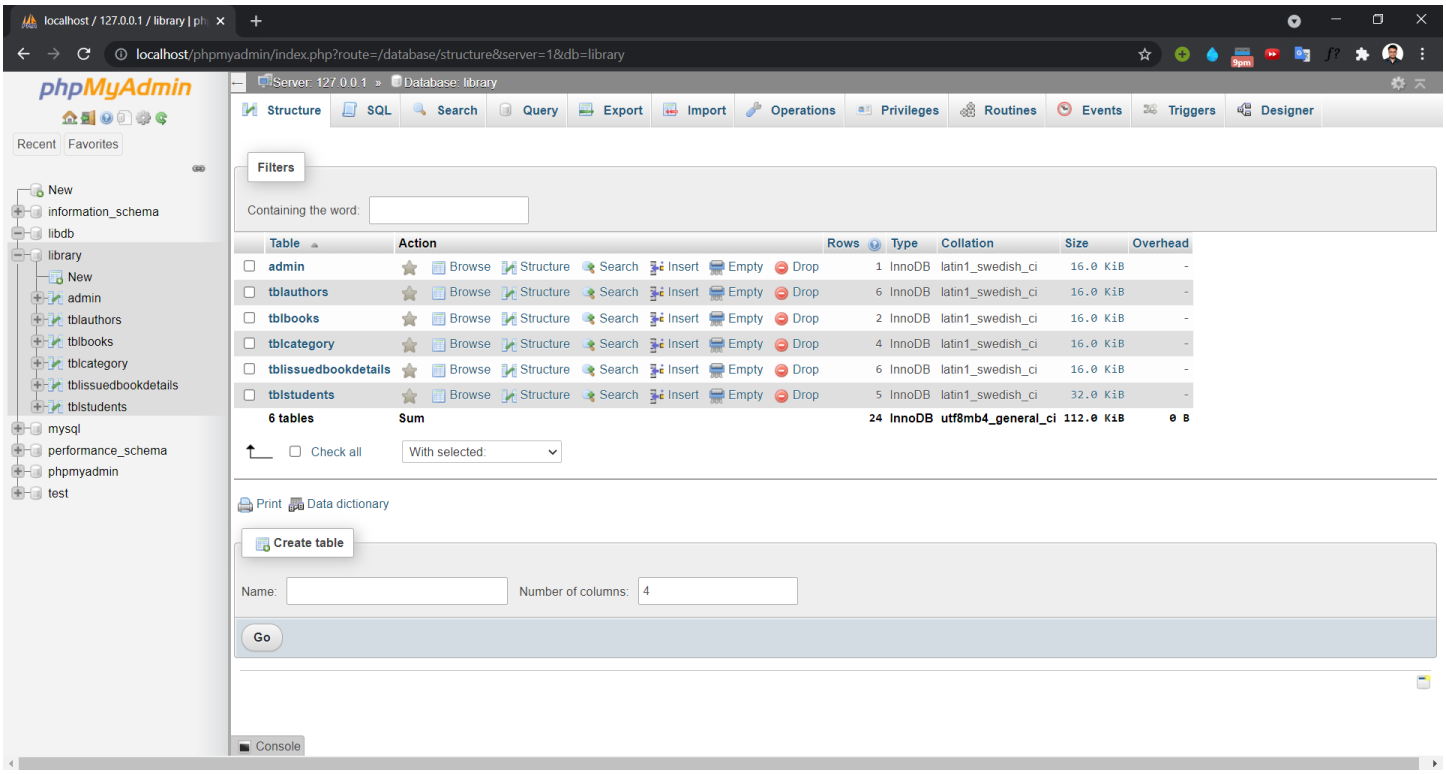
Database After all the query

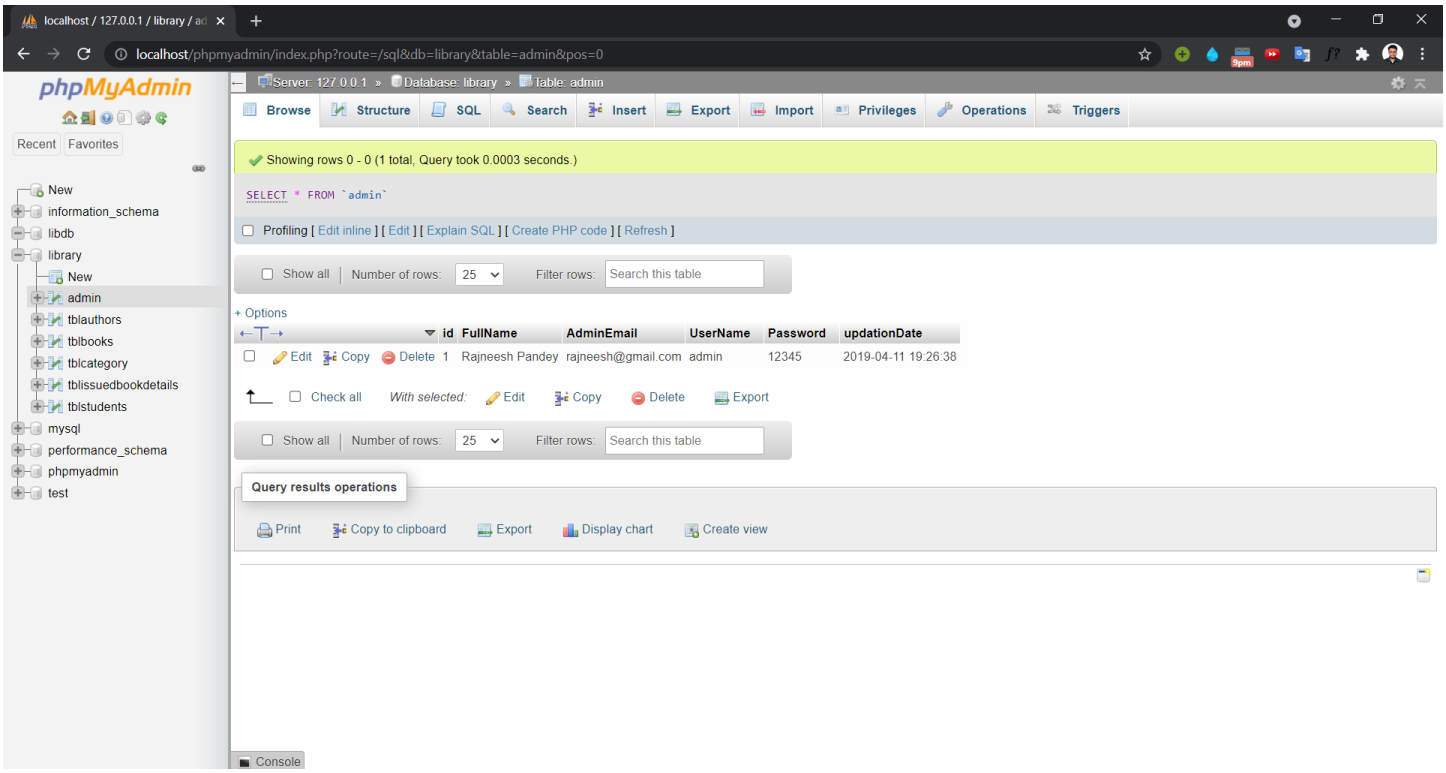
```
> db.empl.find().pretty()
{
        "_id" : ObjectId("617a883a71b2159c0e8fe3a6"),
        "no" : 1,
        "name" : "ST",
        "salary" : 14000,
        "role" : "OB"
}
{
        "_id" : ObjectId("617a883a71b2159c0e8fe3a7"),
        "no" : 2,
        "name" : "MSD",
        "salary" : 5500,
        "role" : "c and WK"
}
```

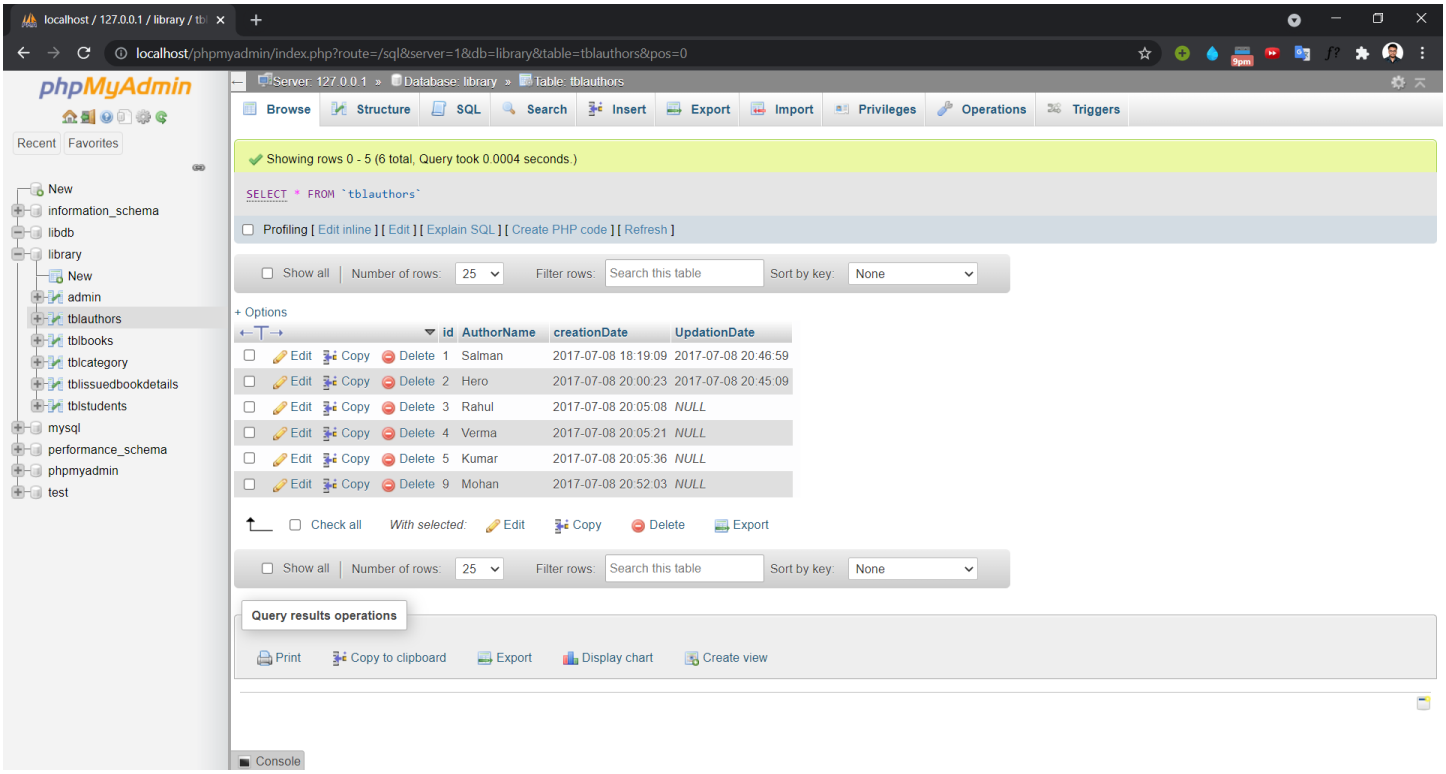
```
{
        "_id" : ObjectId("617a883a71b2159c0e8fe3a8"),
        "no" : 3,
        "name" : "YS",
        "salary" : 5000,
        "role" : "ALR"
}
{
        "_id" : ObjectId("617a883a71b2159c0e8fe3a9"),
        "no" : 4,
        "name" : "RD",
        "salary" : 10000,
        "role" : "MOB"
}
{
        "_id" : ObjectId("617a883a71b2159c0e8fe3aa"),
        "no" : 5,
        "name" : "RS",
        "salary" : 4500,
        "role" : "OB"
}
{
        "_id" : ObjectId("617a883a71b2159c0e8fe3ab"),
        "no" : 6,
        "name" : "BK",
        "salary" : 4500,
        "_id" : ObjectId("617a883a71b2159c0e8fe3ad"),
        "no" : 8,
        "name" : "JB",
        "salary" : 4400,
        "role" : "BW"
}
{
        "_id" : ObjectId("617a883a71b2159c0e8fe3ae"),
        "no" : 9,
        "name" : "HP",
        "salary" : 4400,
        "role" : "ALR"
}
{
        "_id" : ObjectId("617a883a71b2159c0e8fe3af"),
        "no" : 10,
        "name" : "VS",
        "salary" : 4300,
        "role" : "OB"
}
{
        "_id" : ObjectId("617a897aeb8bec42355c9b7b"),
        "no" : 11,
        "name" : "AK",
        "role" : "coach",
        "salary" : 10000
}
> _
```

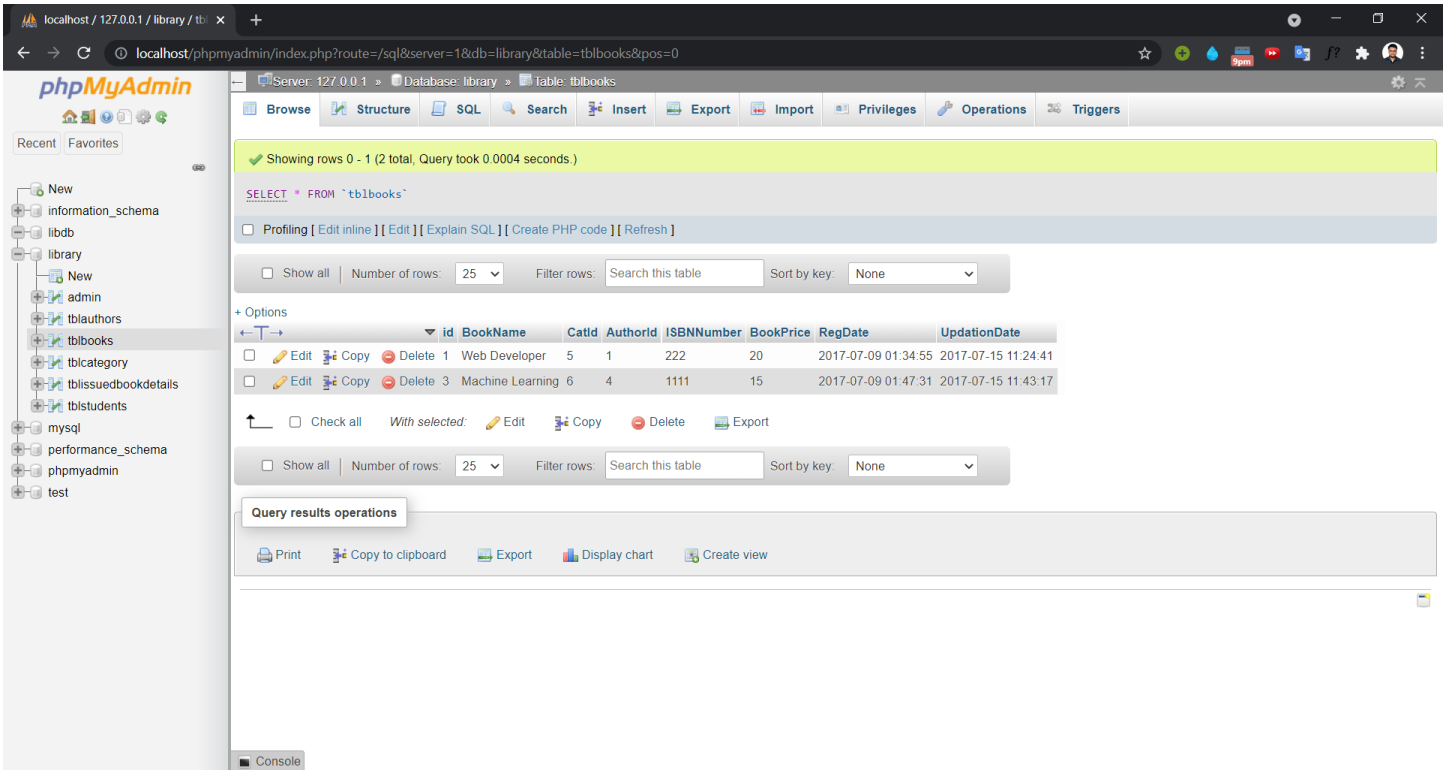
Ex. No.: 11 Date: 11-11-21 Week: 11

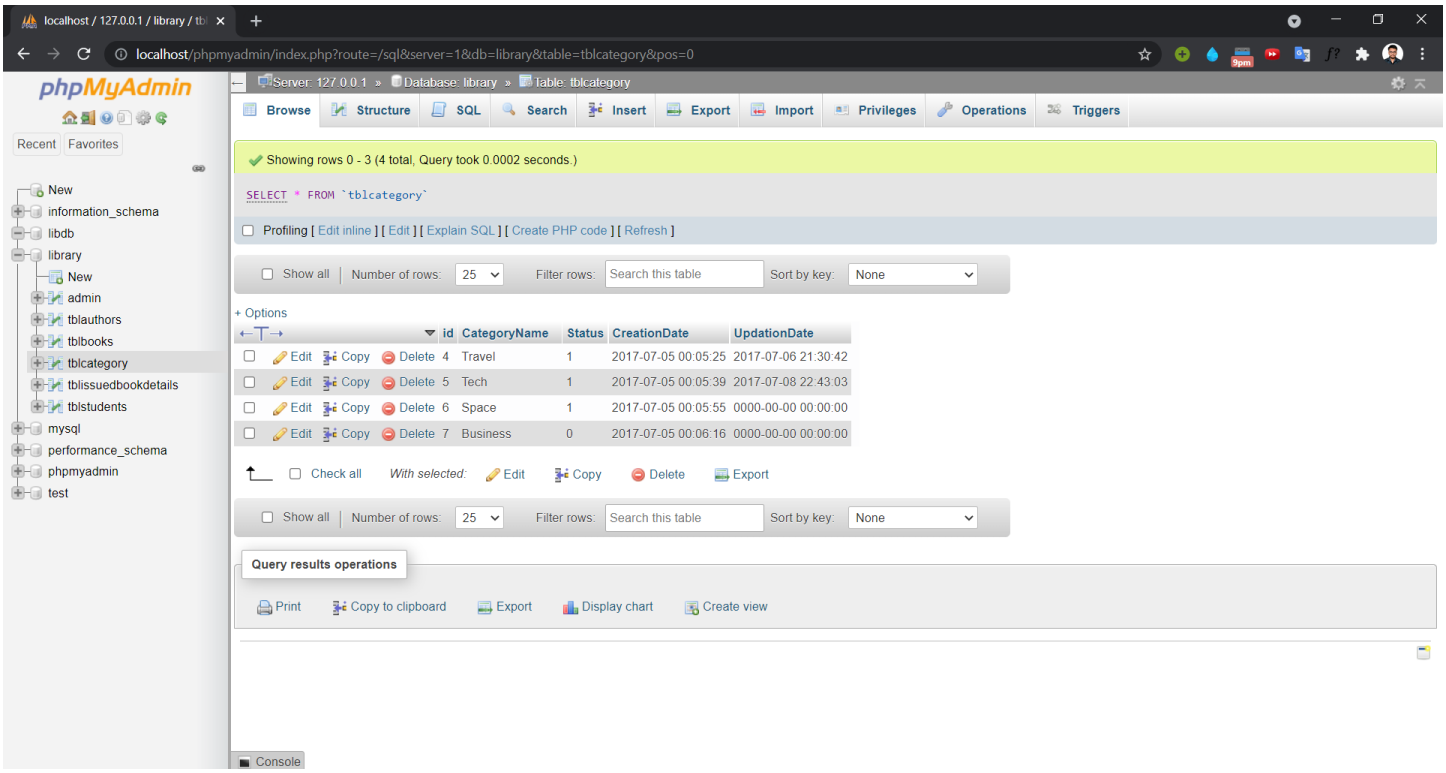
SQL Data Base for Library Management System

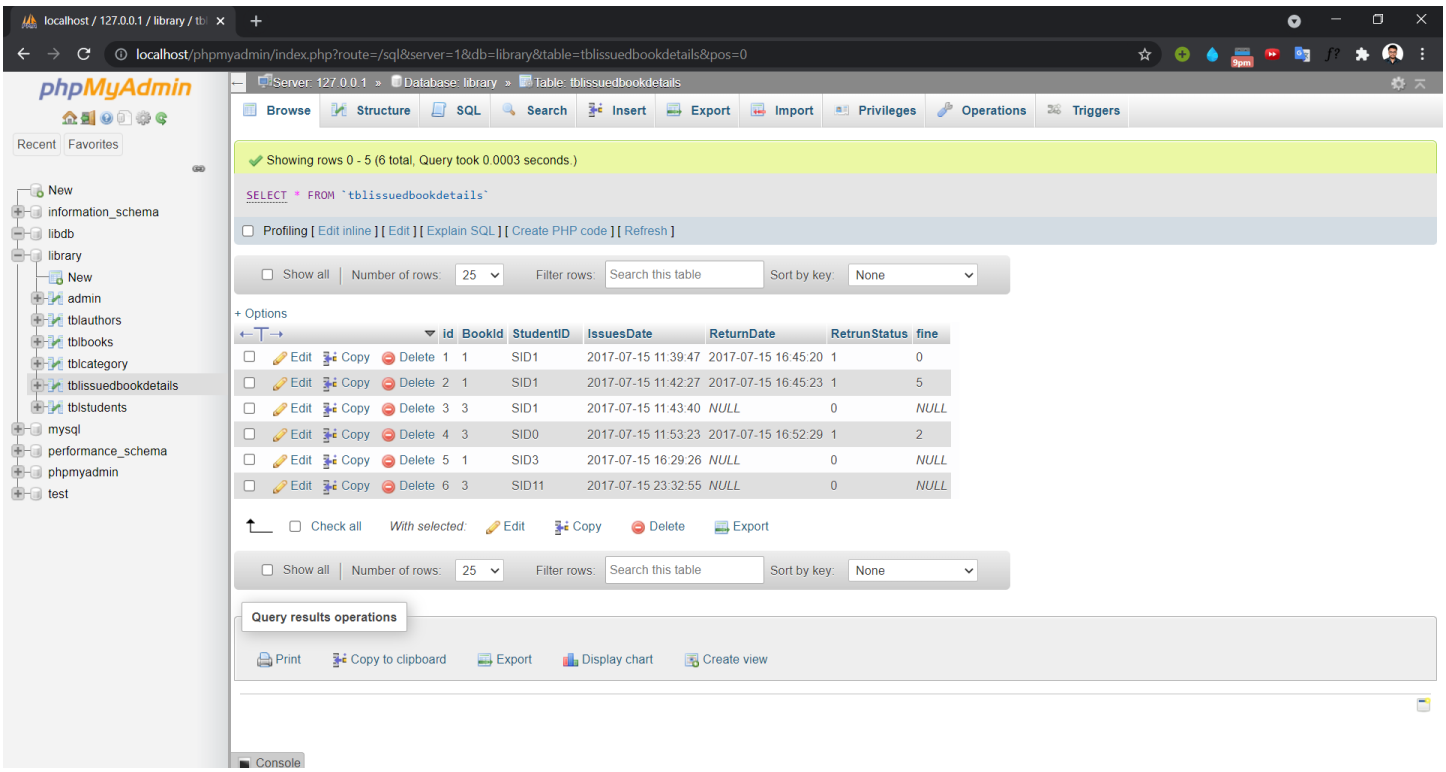


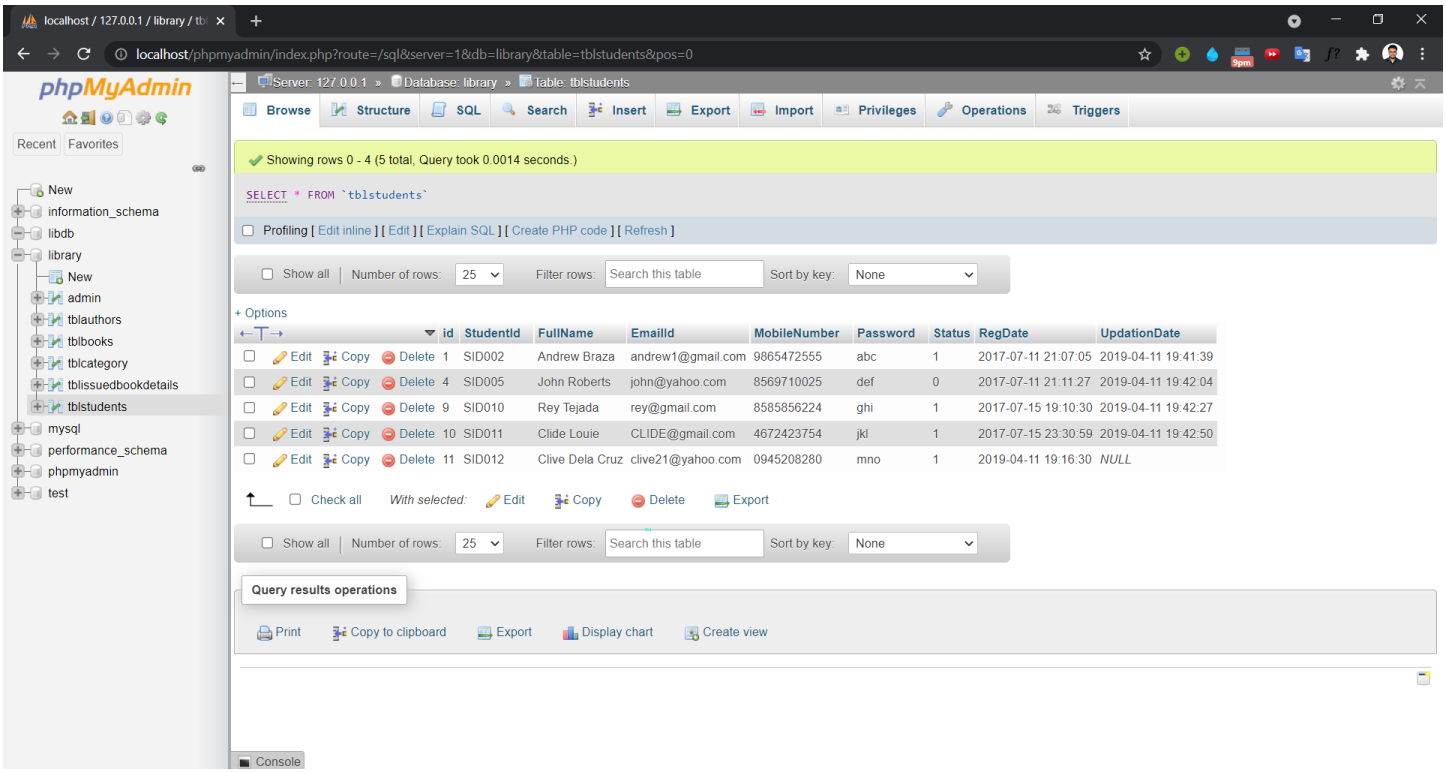

localhost / 127.0.0.1 / library / ad
localhost/phpmyadmin/index.php?route=/sql&db=library&table=admin&pos=0
phpMyAdmin
Recent Favorites
New
information_schema
libdb
library
New
admin
tblauthors
tblbooks
tblcategory
tblissuedbookdetails
tblstudents
mysql
performance_schema
phpmyadmin
test
Server: 127.0.0.1 » Database: library » Table: admin
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
Showing rows 0 - 0 (1 total, Query took 0.0003 seconds.)
SELECT * FROM `admin`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
+ Options
id FullName AdminEmail UserName Password updationDate
Edit Copy Delete 1 Rajneesh Pandey rajneesh@gmail.com admin 12345 2019-04-11 19:26:38
Check all With selected: Edit Copy Delete Export
Show all | Number of rows: 25 Filter rows: Search this table
Query results operations
Print Copy to clipboard Export Display chart Create view
Console


localhost / 127.0.0.1 / library / tbl
localhost/phpmyadmin/index.php?route=/sql&server=1&db=library&table=tblauthors&pos=0
phpMyAdmin
Recent Favorites
New
information_schema
libdb
library
New
admin
tblauthors
tblbooks
tblcategory
tblissuedbookdetails
tblstudents
mysql
performance_schema
phpmyadmin
test
Server: 127.0.0.1 » Database: library » Table: tblauthors
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
Showing rows 0 - 5 (6 total, Query took 0.0004 seconds.)
SELECT * FROM `tblauthors`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
+ Options
id AuthorName creationDate UpdationDate
Edit Copy Delete 1 Salman 2017-07-08 18:19:09 2017-07-08 20:46:59
Edit Copy Delete 2 Hero 2017-07-08 20:00:23 2017-07-08 20:45:09
Edit Copy Delete 3 Rahul 2017-07-08 20:05:08 NULL
Edit Copy Delete 4 Verma 2017-07-08 20:05:21 NULL
Edit Copy Delete 5 Kumar 2017-07-08 20:05:36 NULL
Edit Copy Delete 9 Mohan 2017-07-08 20:52:03 NULL
Check all With selected: Edit Copy Delete Export
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Query results operations
Print Copy to clipboard Export Display chart Create view
Console


localhost / 127.0.0.1 / library / tbl
localhost/phpmyadmin/index.php?route=/sql&server=1&db=library&table=tblbooks&pos=0
phpMyAdmin
Recent
Favorites
New
information_schema
libdb
library
New
admin
tblauthors
tblbooks
tblcategory
tblissuedbookdetails
tblstudents
mysql
performance_schema
phpmyadmin
test
Server: 127.0.0.1 » Database: library » Table: tblbooks
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Triggers
Showing rows 0 - 1 (2 total, Query took 0.0004 seconds.)
SELECT * FROM `tblbooks`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
Filter rows: Search this table
Sort by key: None
+ Options
id BookName CatId AuthorId ISBNNumber BookPrice RegDate UpdationDate
Edit Copy Delete 1 Web Developer 5 1 222 20 2017-07-09 01:34:55 2017-07-15 11:24:41
Edit Copy Delete 3 Machine Learning 6 4 1111 15 2017-07-09 01:47:31 2017-07-15 11:43:17
Check all
With selected:
Edit
Copy
Delete
Export
Show all
Number of rows: 25
Filter rows: Search this table
Sort by key: None
Query results operations
Print
Copy to clipboard
Export
Display chart
Create view
Console


localhost / 127.0.0.1 / library / tbl
localhost/phpmyadmin/index.php?route=/sql&server=1&db=library&table=tblcategory&pos=0
phpMyAdmin
Recent Favorites
New
information_schema
libdb
library
New
admin
tblauthors
tblbooks
tblcategory
tblissuedbookdetails
tblstudents
mysql
performance_schema
phpmyadmin
test
Server: 127.0.0.1 » Database: library » Table: tblcategory
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
Showing rows 0 - 3 (4 total, Query took 0.0002 seconds.)
SELECT * FROM `tblcategory`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all Number of rows: 25 Filter rows: Search this table Sort by key: None
+ Options
id CategoryName Status CreationDate UpdationDate
Edit Copy Delete 4 Travel 1 2017-07-05 00:05:25 2017-07-06 21:30:42
Edit Copy Delete 5 Tech 1 2017-07-05 00:05:39 2017-07-08 22:43:03
Edit Copy Delete 6 Space 1 2017-07-05 00:05:55 0000-00-00 00:00:00
Edit Copy Delete 7 Business 0 2017-07-05 00:06:16 0000-00-00 00:00:00
Check all With selected: Edit Copy Delete Export
Show all Number of rows: 25 Filter rows: Search this table Sort by key: None
Query results operations
Print Copy to clipboard Export Display chart Create view
Console


localhost / 127.0.0.1 / library / tbl
localhost/phpmyadmin/index.php?route=/sql&server=1&db=library&table=tblissuedbookdetails&pos=0
phpMyAdmin
Recent Favorites
New
information_schema
libdb
library
New
admin
tblauthors
tblbooks
tblcategory
tblissuedbookdetails
tblstudents
mysql
performance_schema
phpmyadmin
test
Server: 127.0.0.1 » Database: library » Table: tblissuedbookdetails
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
Showing rows 0 - 5 (6 total, Query took 0.0003 seconds.)
SELECT * FROM `tblissuedbookdetails`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all Number of rows: 25 Filter rows: Search this table Sort by key: None
+ Options
id BookId StudentID IssuesDate ReturnDate RetrunStatus fine
Edit Copy Delete 1 1 SID1 2017-07-15 11:39:47 2017-07-15 16:45:20 1 0
Edit Copy Delete 2 1 SID1 2017-07-15 11:42:27 2017-07-15 16:45:23 1 5
Edit Copy Delete 3 3 SID1 2017-07-15 11:43:40 NULL 0 NULL
Edit Copy Delete 4 3 SID0 2017-07-15 11:53:23 2017-07-15 16:52:29 1 2
Edit Copy Delete 5 1 SID3 2017-07-15 16:29:26 NULL 0 NULL
Edit Copy Delete 6 3 SID11 2017-07-15 23:32:55 NULL 0 NULL
Check all With selected: Edit Copy Delete Export
Show all Number of rows: 25 Filter rows: Search this table Sort by key: None
Query results operations
Print Copy to clipboard Export Display chart Create view
Console



Code :

```
-- phpMyAdmin SQL Dump
-- version 4.8.4
-- https://www.phpmyadmin.net/
--
-- Host: 127.0.0.1
-- Generation Time: Apr 11, 2019 at 04:13 PM
-- Server version: 10.1.37-MariaDB
-- PHP Version: 7.3.0

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;
```

```sql
--
-- Database: `library`
--
CREATE database library;
USE library;
-- --------------------------------------------------------
--
-- Table structure for table `admin`
--
CREATE TABLE `admin` (
  `id` int(11) NOT NULL,
  `FullName` varchar(100) DEFAULT NULL,
  `AdminEmail` varchar(120) DEFAULT NULL,
  `UserName` varchar(100) NOT NULL,
  `Password` varchar(100) NOT NULL,
  `updationDate` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00' ON UPDATE
CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
-- Dumping data for table `admin`
--
INSERT INTO `admin` (`id`, `FullName`, `AdminEmail`, `UserName`, `Password`,
`updationDate`) VALUES
(1, 'Rajneesh Pandey', 'rajneesh@gmail.com', 'admin', '12345', '2019-04-11
13:56:38');

-- --------------------------------------------------------
-- Table structure for table `tblauthors`

CREATE TABLE `tblauthors` (
  `id` int(11) NOT NULL,
  `AuthorName` varchar(159) DEFAULT NULL,
  `creationDate` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `UpdationDate` timestamp NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```sql
--
-- Dumping data for table `tblauthors`
--

INSERT INTO `tblauthors` (`id`, `AuthorName`, `creationDate`, `UpdationDate`)
VALUES
(1, 'Salman', '2017-07-08 12:49:09', '2017-07-08 15:16:59'),
(2, 'Hero', '2017-07-08 14:30:23', '2017-07-08 15:15:09'),
(3, 'Rahul', '2017-07-08 14:35:08', NULL),
(4, 'Verma', '2017-07-08 14:35:21', NULL),
(5, 'Kumar', '2017-07-08 14:35:36', NULL),
(9, 'Mohan', '2017-07-08 15:22:03', NULL);

-- --------------------------------------------------------
--
-- Table structure for table `tblbooks`
--
CREATE TABLE `tblbooks` (
  `id` int(11) NOT NULL,
  `BookName` varchar(255) DEFAULT NULL,
  `CatId` int(11) DEFAULT NULL,
  `AuthorId` int(11) DEFAULT NULL,
  `ISBNNumber` int(11) DEFAULT NULL,
  `BookPrice` int(11) DEFAULT NULL,
  `RegDate` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `UpdationDate` timestamp NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
-- Dumping data for table `tblbooks`
--
INSERT INTO `tblbooks` (`id`, `BookName`, `CatId`, `AuthorId`, `ISBNNumber`,
`BookPrice`, `RegDate`, `UpdationDate`) VALUES
(1, 'Web Developer', 5, 1, 222, 20, '2017-07-08 20:04:55', '2017-07-15 05:54:41'),
(3, 'Machine Learning', 6, 4, 1111, 15, '2017-07-08 20:17:31', '2017-07-15
06:13:17');
```

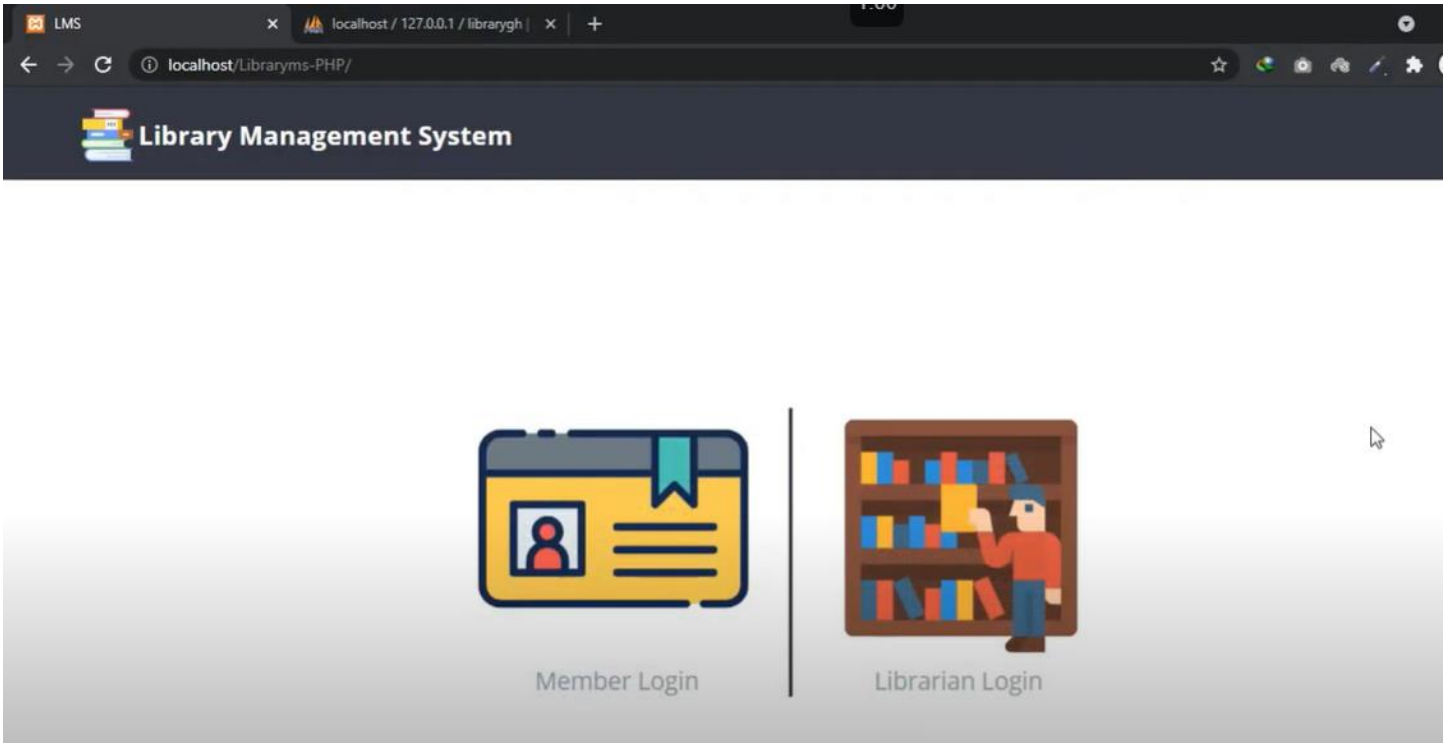
```sql
-- --------------------------------------------------------

--
-- Table structure for table `tblcategory`
--
CREATE TABLE `tblcategory` (
  `id` int(11) NOT NULL,
  `CategoryName` varchar(150) DEFAULT NULL,
  `Status` int(1) DEFAULT NULL,
  `CreationDate` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `UpdationDate` timestamp NULL DEFAULT '0000-00-00 00:00:00' ON UPDATE
CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
-- Dumping data for table `tblcategory`
--
INSERT INTO `tblcategory` (`id`, `CategoryName`, `Status`, `CreationDate`,
`UpdationDate`) VALUES
(4, 'Travel', 1, '2017-07-04 18:35:25', '2017-07-06 16:00:42'),
(5, 'Tech', 1, '2017-07-04 18:35:39', '2017-07-08 17:13:03'),
(6, 'Space', 1, '2017-07-04 18:35:55', '0000-00-00 00:00:00'),
(7, 'Business', 0, '2017-07-04 18:36:16', '0000-00-00 00:00:00');

-- --------------------------------------------------------
--
-- Table structure for table `tblissuedbookdetails`
--
CREATE TABLE `tblissuedbookdetails` (
  `id` int(11) NOT NULL,
  `BookId` int(11) DEFAULT NULL,
  `StudentID` varchar(150) DEFAULT NULL,
  `IssuesDate` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `ReturnDate` timestamp NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP,
  `RetrunStatus` int(1) DEFAULT NULL,
  `fine` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

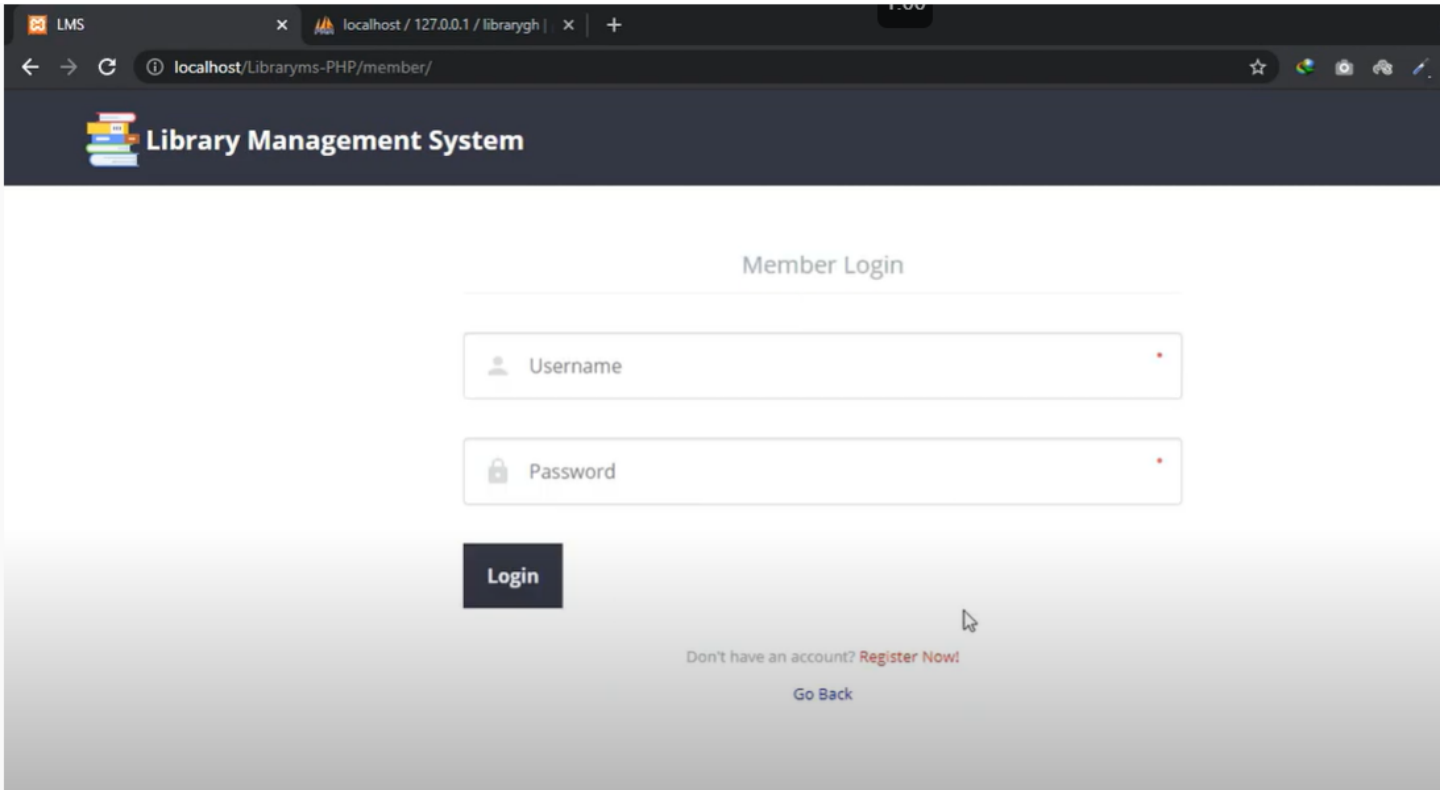
```sql
--
-- Dumping data for table `tblissuedbookdetails`
--

INSERT INTO `tblissuedbookdetails` (`id`, `BookId`, `StudentID`, `IssuesDate`,
`ReturnDate`, `RetrunStatus`, `fine`) VALUES
(1, 1, 'SID1', '2017-07-15 06:09:47', '2017-07-15 11:15:20', 1, 0),
(2, 1, 'SID1', '2017-07-15 06:12:27', '2017-07-15 11:15:23', 1, 5),
(3, 3, 'SID1', '2017-07-15 06:13:40', NULL, 0, NULL),
(4, 3, 'SID0', '2017-07-15 06:23:23', '2017-07-15 11:22:29', 1, 2),
(5, 1, 'SID3', '2017-07-15 10:59:26', NULL, 0, NULL),
(6, 3, 'SID11', '2017-07-15 18:02:55', NULL, 0, NULL);

-- --------------------------------------------------------
--
-- Table structure for table `tblstudents`
--
CREATE TABLE `tblstudents` (
  `id` int(11) NOT NULL,
  `StudentId` varchar(100) DEFAULT NULL,
  `FullName` varchar(120) DEFAULT NULL,
  `EmailId` varchar(120) DEFAULT NULL,
  `MobileNumber` char(11) DEFAULT NULL,
  `Password` varchar(120) DEFAULT NULL,
  `Status` int(1) DEFAULT NULL,
  `RegDate` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `UpdationDate` timestamp NULL DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
-- Dumping data for table `tblstudents`
--
INSERT INTO `tblstudents` (`id`, `StudentId`, `FullName`, `EmailId`,
`MobileNumber`, `Password`, `Status`, `RegDate`, `UpdationDate`) VALUES
```

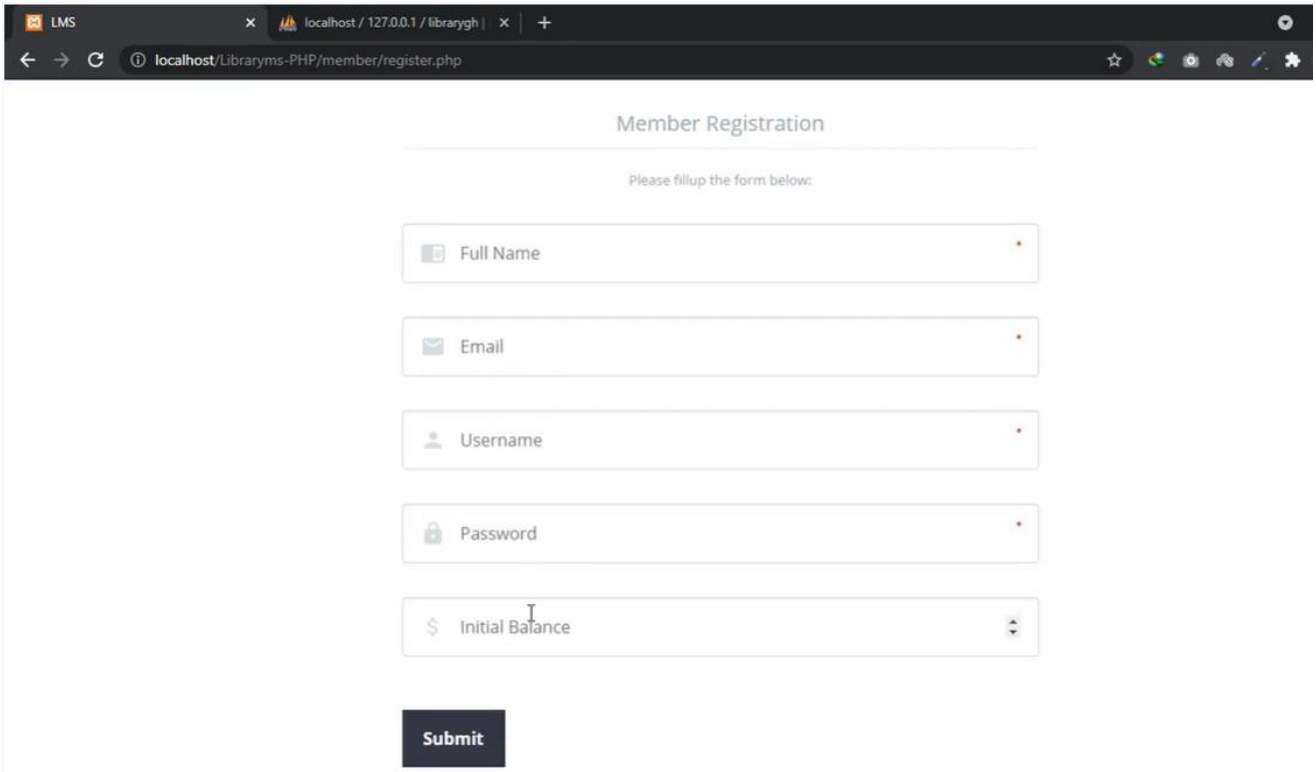
```
(1, 'SID002', 'Andrew Braza', 'andrew1@gmail.com', '9865472555', 'abc', 1, '2017-07-11 15:37:05', '2019-04-11 14:11:39'),
(4, 'SID005', 'John Roberts', 'john@yahoo.com', '8569710025', 'def', 0, '2017-07-11 15:41:27', '2019-04-11 14:12:04'),
(9, 'SID010', 'Rey Tejada', 'rey@gmail.com', '8585856224', 'ghi', 1, '2017-07-15 13:40:30', '2019-04-11 14:12:27'),
(10, 'SID011', 'Clide Louie', 'CLIDE@gmail.com', '4672423754', 'jkl', 1, '2017-07-15 18:00:59', '2019-04-11 14:12:50'),
(11, 'SID012', 'Clive Dela Cruz', 'clive21@yahoo.com', '0945208280', 'mno', 1, '2019-04-11 13:46:30', NULL);
--
-- Indexes for dumped tables
-
--
-- Indexes for table `admin`
--
ALTER TABLE `admin`
  ADD PRIMARY KEY (`id`);
--
-- Indexes for table `tblauthors`
--
ALTER TABLE `tblauthors`
  ADD PRIMARY KEY (`id`);
--
-- Indexes for table `tblbooks`
--
ALTER TABLE `tblbooks`
  ADD PRIMARY KEY (`id`);
--
-- Indexes for table `tblcategory`
ALTER TABLE `tblcategory`
  ADD PRIMARY KEY (`id`);
--
-- Indexes for table `tblissuedbookdetails`
--
ALTER TABLE `tblissuedbookdetails`
```

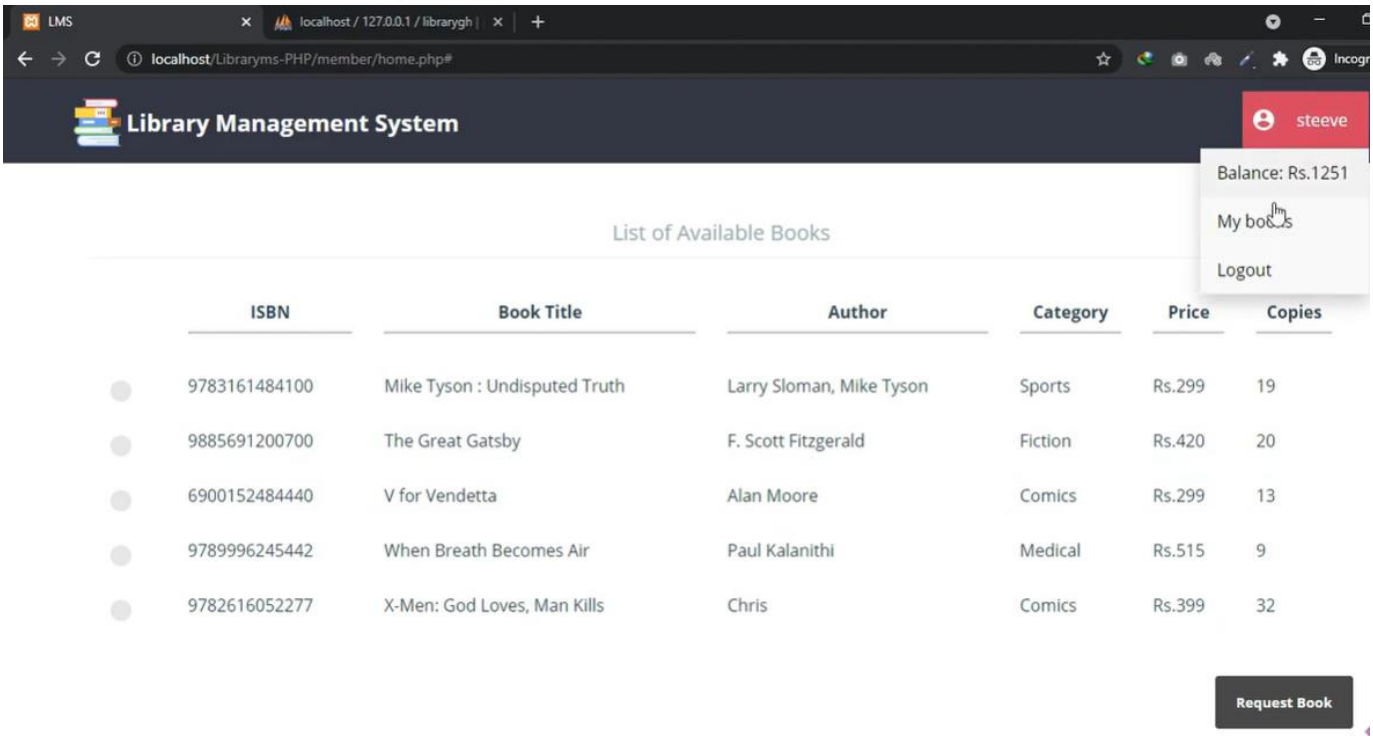
```
  ADD PRIMARY KEY (`id`);
-- Indexes for table `tblstudents`
--
ALTER TABLE `tblstudents`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `StudentId` (`StudentId`);
-- AUTO_INCREMENT for dumped tables
-- AUTO_INCREMENT for table `admin`
--
ALTER TABLE `admin`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;
--
-- AUTO_INCREMENT for table `tblauthors`
--
ALTER TABLE `tblauthors`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=10;
--
-- AUTO_INCREMENT for table `tblbooks`
--
ALTER TABLE `tblbooks`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;
--
-- AUTO_INCREMENT for table `tblcategory`
--
ALTER TABLE `tblcategory`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=8;
--
-- AUTO_INCREMENT for table `tblissuedbookdetails`
--
ALTER TABLE `tblissuedbookdetails`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=7;
--
-- AUTO_INCREMENT for table `tblstudents`
--
ALTER TABLE `tblstudents`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=12;
```

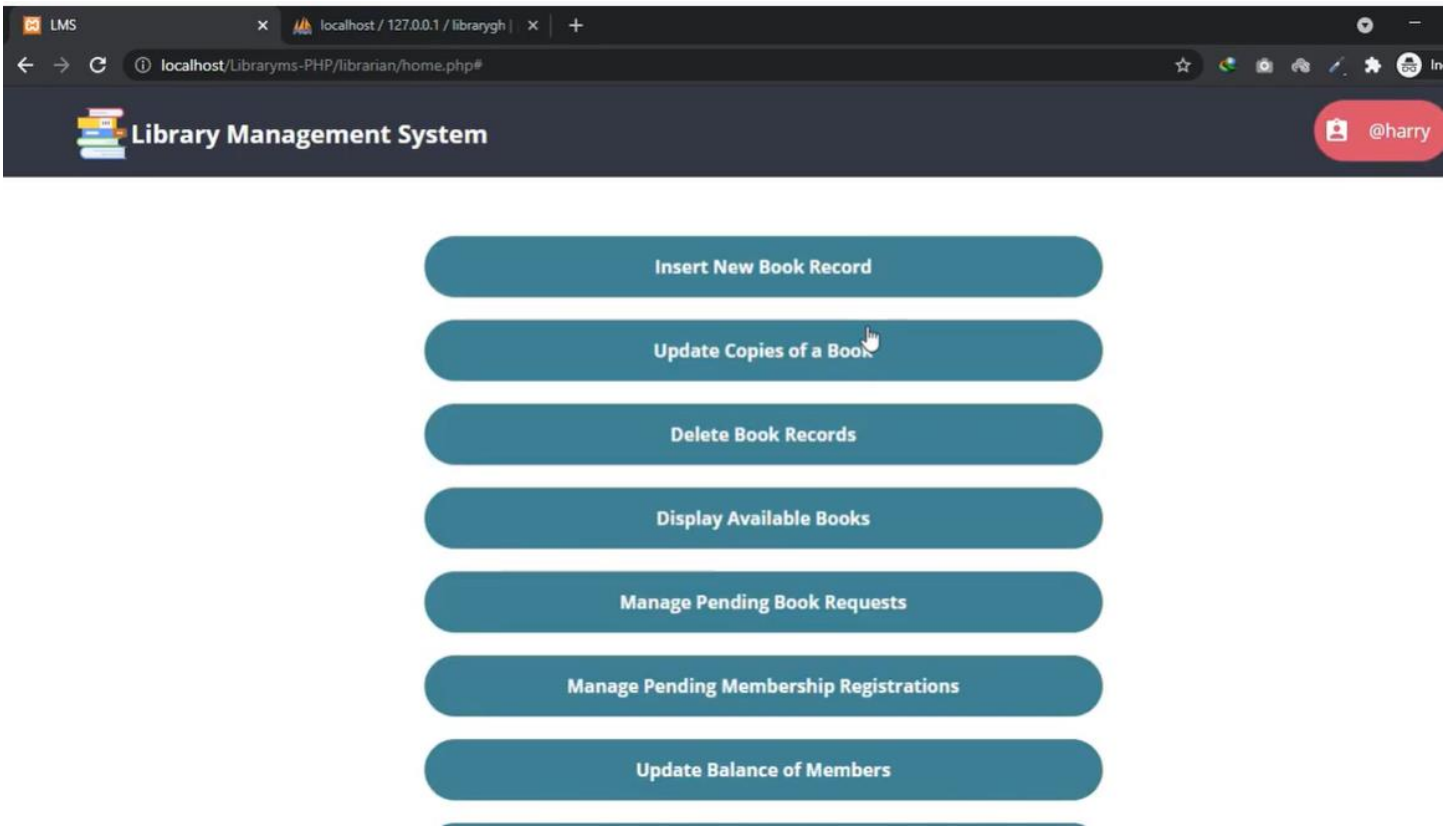
```
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

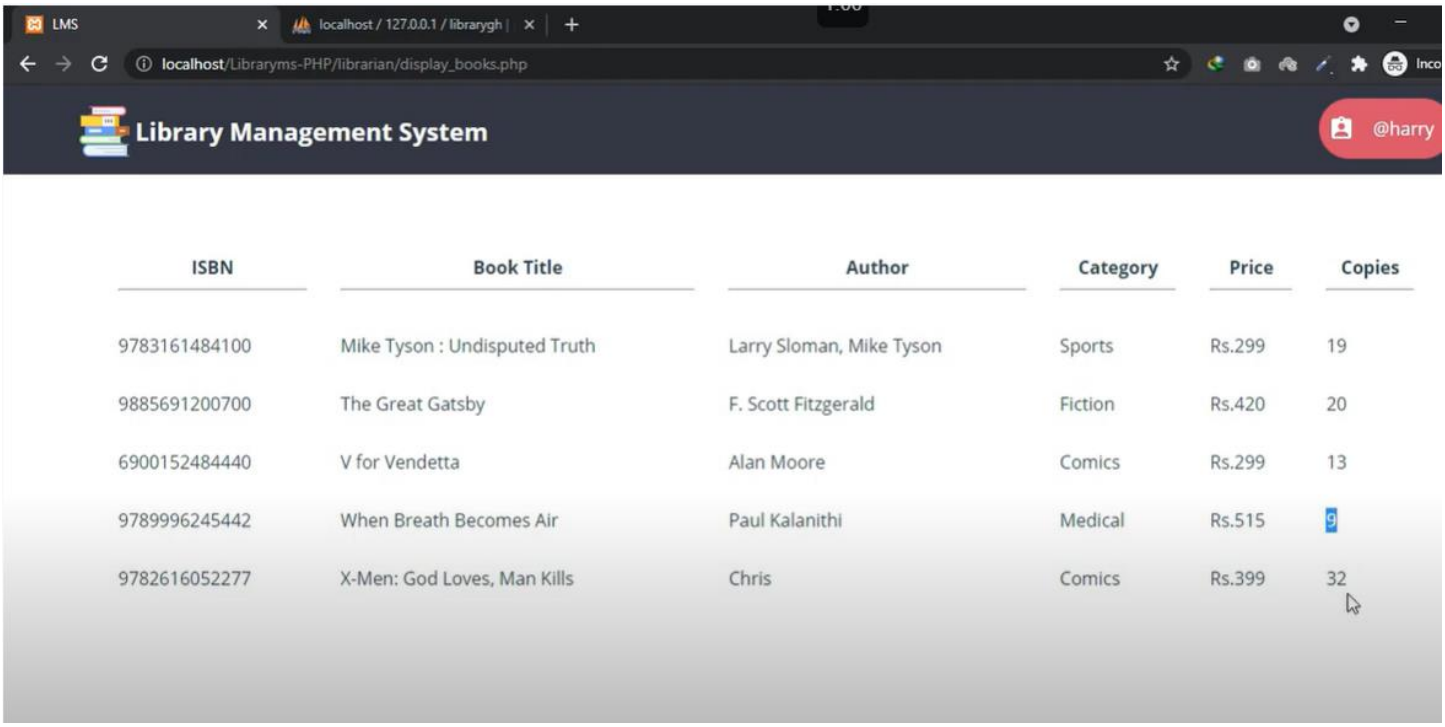



LMS
localhost / 127.0.0.1 / librarygh
localhost/Libraryms-PHP/member/
Library Management System
Member Login
Username
Password
Login
Don't have an account? Register Now!
Go Back

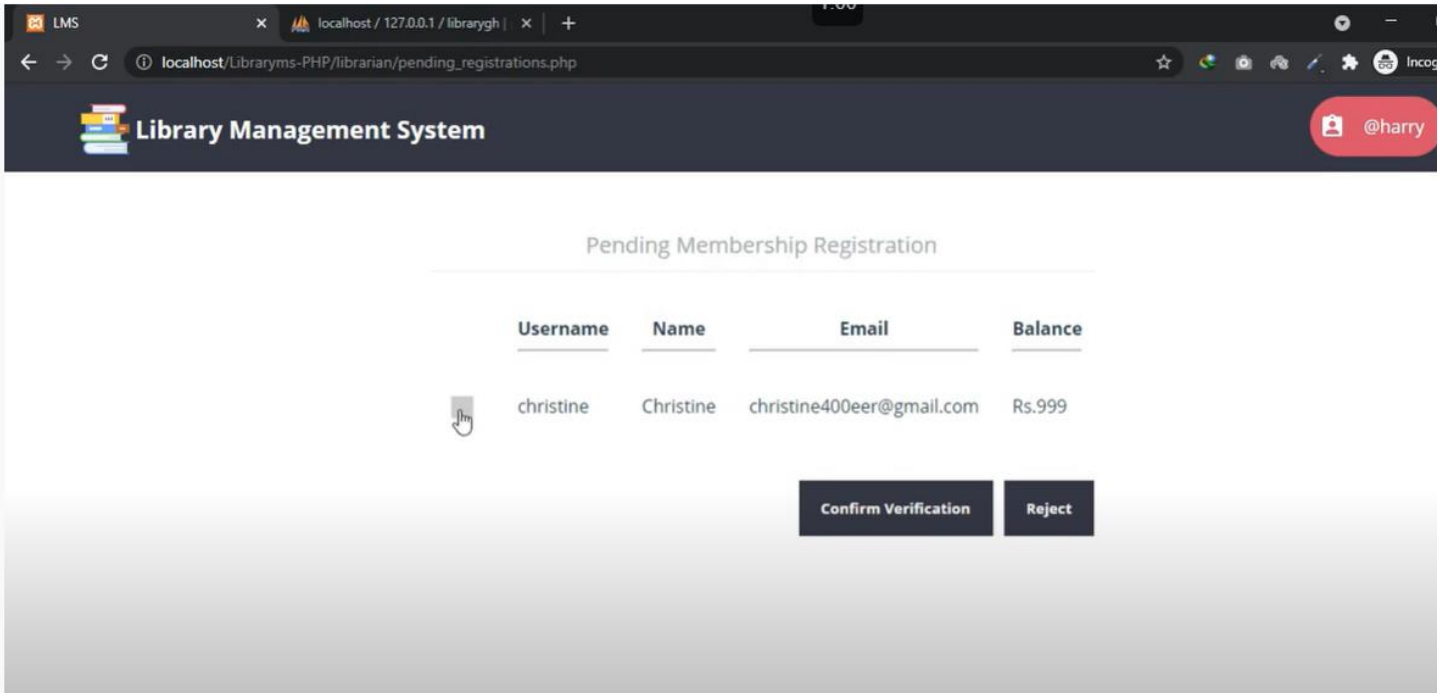

LMS
localhost / 127.0.0.1 / librarygh
localhost/Libraryms-PHP/member/register.php
Member Registration
Please fillup the form below:
Full Name
Email
Username
Password
Initial Balance
Submit


LMS
localhost / 127.0.0.1 / librarygh
localhost/Libraryms-PHP/member/home.php#
Library Management System
steeve
Balance: Rs.1251
My books
Logout
List of Available Books
ISBN
Book Title
Author
Category
Price
Copies
9783161484100
Mike Tyson : Undisputed Truth
Larry Sloman, Mike Tyson
Sports
Rs.299
19
9885691200700
The Great Gatsby
F. Scott Fitzgerald
Fiction
Rs.420
20
6900152484440
V for Vendetta
Alan Moore
Comics
Rs.299
13
9789996245442
When Breath Becomes Air
Paul Kalanithi
Medical
Rs.515
9
9782616052277
X-Men: God Loves, Man Kills
Chris
Comics
Rs.399
32
Request Book



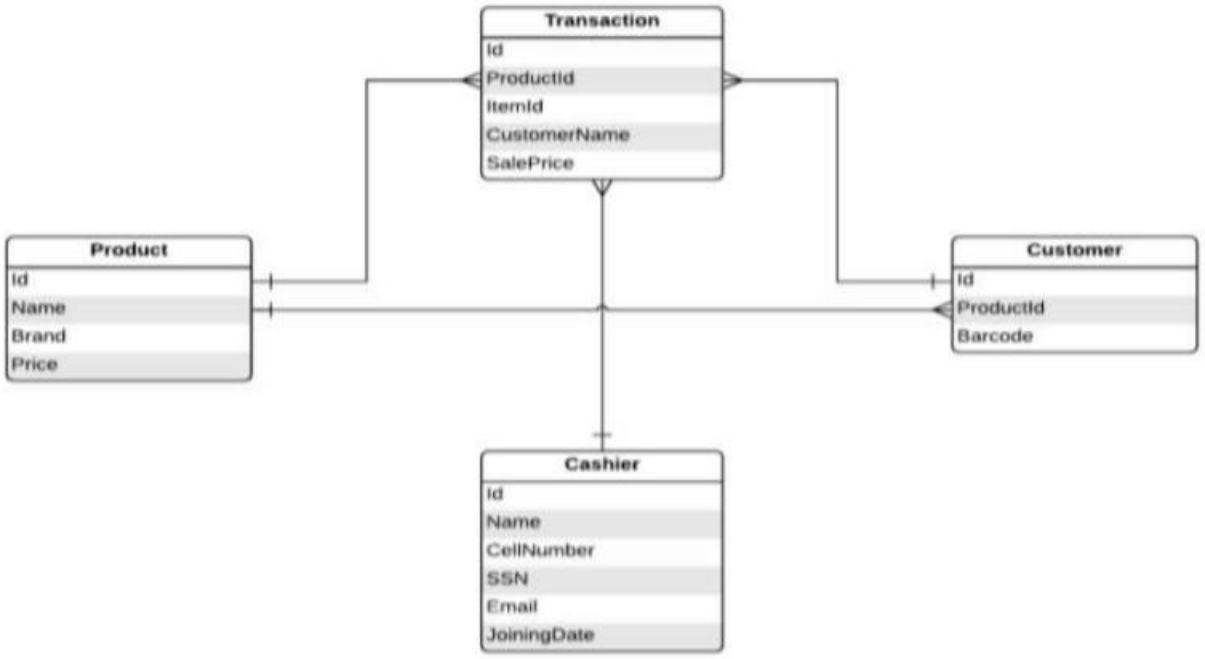


ISBN	Book Title	Author	Category	Price	Copies
9783161484100	Mike Tyson : Undisputed Truth	Larry Sloman, Mike Tyson	Sports	Rs.299	19
9885691200700	The Great Gatsby	F. Scott Fitzgerald	Fiction	Rs.420	20
6900152484440	V for Vendetta	Alan Moore	Comics	Rs.299	13
9789996245442	When Breath Becomes Air	Paul Kalanithi	Medical	Rs.515	9
9782616052277	X-Men: God Loves, Man Kills	Chris	Comics	Rs.399	32

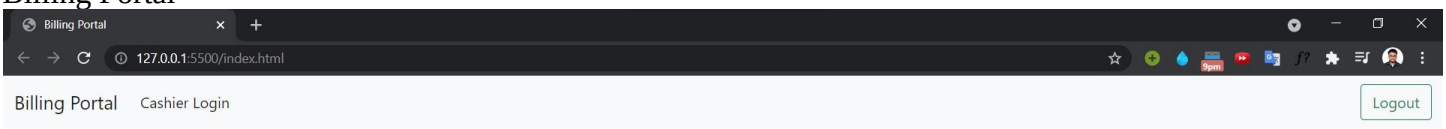

LMS
localhost / 127.0.0.1 / librarygh
localhost/Libraryms-PHP/librarian/pending_registrations.php
Library Management System
@harry
Pending Membership Registration
Username
Name
Email
Balance
christine
Christine
christine400eer@gmail.com
Rs.999
Confirm Verification
Reject

Ex. No.: 12 Date: 18-11-21 Week:12

ER Diagram :



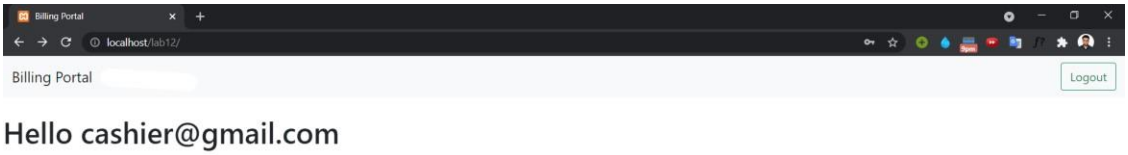
Billing Portal



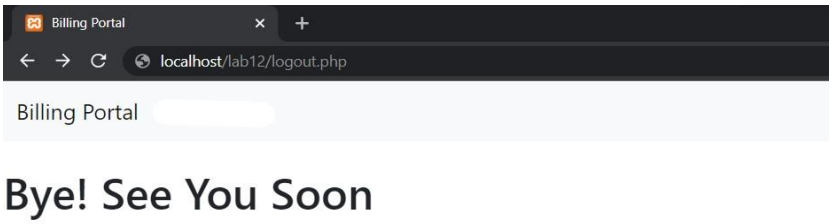
Welcome! to the Billing Portal

1. Login page for login.php

On Successfull Sign in Redirected to home.php



Logout.php



2. Page for the cashier to add products to the table.

DB Before:

```
_id: ObjectId("61961002b6b4bd4cb140ede3")
id: 1
name: "Atta"
brand: "ABC"
price: 100
```

```
_id: ObjectId("61961002b6b4bd4cb140ede4")
id: 2
name: "Rice"
brand: "XYZ"
price: 100
```

DB after:

```
_id: ObjectId("61961002b6b4bd4cb140ede3")
id: 1
name: "Atta"
brand: "ABC"
price: 100
```

```
_id: ObjectId("61961002b6b4bd4cb140ede4")
id: 2
name: "Rice"
brand: "XYZ"
price: 100
```

```
_id: ObjectId("6196932dc4496f288138dae6")
id: "3"
name: "Pulses"
brand: "Aashirwad"
price: "50"
```

3.Transaction page for cashier for billing the products.

Bill.php (real-time total bill calculation)

Updated in the DB:

```
  _id: ObjectId("61969a68c4496f288138dae7")
› product_id: Array
  customer_name: "Spider"
  sale_price: 350
```

Thankyou

Roll no. : **106119100**

Name : **Rajneesh Pandey**

Section : **CSE-B**

----------End of Record File----------