Software Engineering Assignment

MODULE: 5 (DataBase)

==========

1) Create two tables Student and Exam and link two tables through Primary Key and Foreign Key.

==========

```
CREATE TABLE Student
(
        Rollno int PRIMARY KEY,
        Name varchar(20),
        Branch varchar(20)
)
INSERT INTO student(Rollno,Name,Branch) VALUES(1,"Jay","Computer Science");
INSERT INTO student(Rollno,Name,Branch) VALUES(2,"Suhani","Electronic and Com");
INSERT INTO student(Rollno,Name,Branch) VALUES(3,"Kriti","Electronic and Com");

CREATE TABLE Exam
(
        Rollno int,
        S_code varchar(20),
        Marks int,
        P_code varchar(20),

        FOREIGN KEY(Rollno) REFERENCES student(Rollno)
)
INSERT INTO exam(Rollno,S_code,Marks,P_code) VALUES(1,"CS11",50,"CS");
INSERT INTO exam(Rollno,S_code,Marks,P_code) VALUES(1,"CS12",60,"CS");
INSERT INTO exam(Rollno,S_code,Marks,P_code) VALUES(2,"EC101",66,"EC");
INSERT INTO exam(Rollno,S_code,Marks,P_code) VALUES(2,"EC102",70,"EC");
```

```
INSERT INTO exam(Rollno,S_code,Marks,P_code) VALUES(3,"EC101",45,"EC");
INSERT INTO exam(Rollno,S_code,Marks,P_code) VALUES(3,"EC102",50,"EC");
```

==========

----> Create two tables Employee and Incentive and link two tables

==========

```
CREATE TABLE Employee
(
        Employee_id int PRIMARY KEY,
        First_name varchar(20),
        Last_name varchar(20),
        Salary int,
        Joining_date date,
        Department varchar(20)
)

INSERT INTO employee(Employee_id,First_name,Last_name,Salary,Joining_date,Department)
VALUES(1,"John","Abraham",1000000,"01-JAN-13 12.00.00 AM","Banking");
INSERT INTO employee(Employee_id,First_name,Last_name,Salary,Joining_date,Department)
VALUES(2,"Michael","Clarke",800000,"01-JAN-13 12.00.00 AM","Insurance");
INSERT INTO employee(Employee_id,First_name,Last_name,Salary,Joining_date,Department)
VALUES(3,"Roy","Thomas",700000,"01-FEB-13 12.00.00 AM","Banking");
INSERT INTO employee(Employee_id,First_name,Last_name,Salary,Joining_date,Department)
VALUES(4,"Tom","Jose",600000,"01-FEB-13 12.00.00 AM","Insurance");
INSERT INTO employee(Employee_id,First_name,Last_name,Salary,Joining_date,Department)
VALUES(5,"Jerry","Pinto",650000,"01-FEB-13 12.00.00 AM","Insurance");
INSERT INTO employee(Employee_id,First_name,Last_name,Salary,Joining_date,Department)
VALUES(6,"Philip","Mathew",750000,"01-JAN-13 12.00.00 AM","Services");
```

```
INSERT INTO employee(Employee_id,First_name,Last_name,Salary,Joining_date,Department)
VALUES(7,"TestName1","123",650000,"01-JAN-13 12.00.00 AM","Services");
INSERT INTO employee(Employee_id,First_name,Last_name,Salary,Joining_date,Department)
VALUES(8,"TestName2","Lname%",600000,"01-FEB-13 12.00.00 AM","Insurance");


CREATE TABLE Incentive
(
        Employee_ref_id int,
        Incentive_date date,
        Incentive_amount int,

        FOREIGN KEY(Employee_ref_id) REFERENCES employee(Employee_id)
)
INSERT INTO incentive(Employee_ref_id,Incentive_date,Incentive_amount) VALUES(1,"01-FEB-13",5000);
INSERT INTO incentive(Employee_ref_id,Incentive_date,Incentive_amount) VALUES(2,"01-FEB-13",3000);
INSERT INTO incentive(Employee_ref_id,Incentive_date,Incentive_amount) VALUES(3,"01-FEB-13",4000);
INSERT INTO incentive(Employee_ref_id,Incentive_date,Incentive_amount) VALUES(1,"01-JAN-13",4500);
INSERT INTO incentive(Employee_ref_id,Incentive_date,Incentive_amount) VALUES(2,"01-JAN-13",3500);
```

2) Get First_Name from employee table using Tom name “Employee Name”.

```
SELECT * FROM employee WHERE First_name="Tom"
```

3) Get FIRST_NAME, Joining Date, and Salary from employee table.

```
SELECT First_name, Joining_date, Salary FROM employee
```

4) Get all employee details from the employee table order by First_Name Ascending and Salary descending?

```
SELECT * FROM employee ORDER BY First_name ASC
SELECT * FROM employee ORDER BY Salary DESC
```

5) Get employee details from employee table whose first name contains 'J'.

```
SELECT * FROM employee WHERE First_name LIKE 'J%'
```

6) Get department wise maximum salary from employee table order by salary ascending?

```
SELECT * FROM employee ORDER BY Salary ASC
SELECT MAX(Salary), Department FROM employee WHERE Department="Banking"
SELECT MAX(Salary), Department FROM employee WHERE Department="Insurance"
SELECT MAX(Salary), Department FROM employee WHERE Department="Services"
```

7) Select first_name, incentive amount from employee and incentives table forthose employees who have incentives and incentive amount greater than 3000

```
SELECT employee.First_name, incentive.Incentive_amount FROM employee
INNER JOIN incentive ON employee.Employee_id=incentive.Employee_ref_id

SELECT * FROM incentive WHERE Incentive_amount>3000
```

8) Create After Insert trigger on Employee table which insert records in viewtable.

```
CREATE TABLE viewtable
(
    id int,

    fname varchar(20),
    lname varchar(20),
    salary int,
    jdate date,
    department varchar(20),
    date_time timestamp,
    action_performed varchar(40)
)
```

========== **TRIGGER START** ==========

```
DELIMITER $$

CREATE TRIGGER insert_trigger AFTER INSERT ON employee FOR EACH ROW

BEGIN

    INSERT INTO viewtable(id, fname, lname, salary, jdate, department, action_performed)
    VALUES(new.Employee_id, new.First_name, new.Last_name, new.Salary, new.Joining_date,
    new.Department, "Record Inserted!");

END
```

========== **TRIGGER END** ==========

==========

----> Create table given below: Salesperson and Customer

==========

```
CREATE TABLE Salesperson

(

    SNo int PRIMARY KEY,
    SNAME varchar(20),
    CITY varchar(20),
    COMM float

)
```

```
INSERT INTO salesperson(SNo,SNAME,CITY,COMM) VALUES(1001,"Peel","London",0.12);
INSERT INTO salesperson(SNo,SNAME,CITY,COMM) VALUES(1002,"Serres","San Jose",0.13);
INSERT INTO salesperson(SNo,SNAME,CITY,COMM) VALUES(1004,"Motika","London",0.11);
INSERT INTO salesperson(SNo,SNAME,CITY,COMM) VALUES(1007,"Rafkin","Barcelona",0.15);
INSERT INTO salesperson(SNo,SNAME,CITY,COMM) VALUES(1003,"Axelrod","New York",0.1);


CREATE TABLE Customer
(
        CNM int PRIMARY KEY,
        CNAME varchar(20),
        CITY varchar(20),
        RATING int,
        SNo int,
        FOREIGN KEY(SNo) REFERENCES salesperson(SNo)
)
INSERT INTO customer(CNM,CNAME,CITY,RATING,SNo) VALUES(201,"Hoffman","London",100,1001);
INSERT INTO customer(CNM,CNAME,CITY,RATING,SNo) VALUES(202,"Giovanne","Roe",200,1003);
INSERT INTO customer(CNM,CNAME,CITY,RATING,SNo) VALUES(203,"Liu","San Jose",300,1002);
INSERT INTO customer(CNM,CNAME,CITY,RATING,SNo) VALUES(204,"Grass","Barcelona",100,1002);
INSERT INTO customer(CNM,CNAME,CITY,RATING,SNo) VALUES(206,"Ciemens","London",300,1007);
INSERT INTO customer(CNM,CNAME,CITY,RATING,SNo) VALUES(207,"Pereira","Roe",100,1004);
```

9) Names and cities of all salespeople in London with commission above 0.12

```
SELECT SANAME,CITY FROM salesperson WHERE CITY='London' AND COMM > 0.12;
```

10) All salespeople either in Barcelona or in London

```
SELECT SNAME FROM salesperson WHERE CITY="London" OR "Barcelona"
```

11) All salespeople with commission between 0.10 and 0.12. (Boundary valuesshould be excluded).

```
SELECT SNAME,COMM FROM salesperson WHERE COMM BETWEEN 0.10 AND 0.12
```

12) All customers excluding those with rating <= 100 unless they are located in Rome.

```
SELECT * FROM customers WHERE RATING>100 OR CITY='Rome';
```

13) Write a SQL statement that displays all the information about all salespeople

```
SELECT * FROM salesperson
```

==========

-----> Create table given below: salesman and orders

==========

```
CREATE TABLE salesman
(
        salesman_id int PRIMARY KEY,
        name varchar(20),
        city varchar(20),
        commision float
)

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

```
CREATE TABLE orders
(
        ord_no int PRIMARY KEY,
        purch_amt int,
        ord_date date,
        customer_id int,
        salesman_id int,
        FOREIGN KEY(salesman_id) REFERENCES salesman(salesman_id)
)
INSERT INTO orders(ord_no,purch_amt,ord_date,customer_id,salesman_id)
VALUES(70001,150.5,"2012-10-05",3005,5002);
INSERT INTO orders(ord_no,purch_amt,ord_date,customer_id,salesman_id)
VALUES(70009,270.65,"2012-09-10",3001,5005);
INSERT INTO orders(ord_no,purch_amt,ord_date,customer_id,salesman_id)
VALUES(70002,65.26,"2012-10-05",3002,5001);
INSERT INTO orders(ord_no,purch_amt,ord_date,customer_id,salesman_id)
VALUES(70004,110.5,"2012-08-17",3009,5003);
INSERT INTO orders(ord_no,purch_amt,ord_date,customer_id,salesman_id)
VALUES(70007,948.5,"2012-09-10",3005,5002);
INSERT INTO orders(ord_no,purch_amt,ord_date,customer_id,salesman_id)
VALUES(70005,2400.6,"2012-07-27",3007,5001);
INSERT INTO orders(ord_no,purch_amt,ord_date,customer_id,salesman_id)
VALUES(70008,5760,"2012-09-10",3002,5001);
INSERT INTO orders(ord_no,purch_amt,ord_date,customer_id,salesman_id)
VALUES(70010,1983.43,"2012-10-10",3004,5006);
INSERT INTO orders(ord_no,purch_amt,ord_date,customer_id,salesman_id)
VALUES(70003,2480.4,"2012-10-10",3009,5003);
INSERT INTO orders(ord_no,purch_amt,ord_date,customer_id,salesman_id)
VALUES(70012,250.45,"2012-06-27",3008,5002);
```

```
INSERT INTO orders(ord_no,purch_amt,ord_date,customer_id,salesman_id)
VALUES(70011,75.29,"2012-07-17",3003,5007);
INSERT INTO orders(ord_no,purch_amt,ord_date,customer_id,salesman_id)
VALUES(70013,3045.6,"2012-04-25",3002,5001);
```

14) All orders for more than $1000.

```
SELECT * FROM orders WHERE purch_amt>1000;
```

15) From the following table, write a SQL query to find orders that are delivered by a salesperson with ID. 5001. Return ord_no, ord_date, purch_amt.

```
SELECT ord_no,purch_amt,ord_date FROM orders WHERE salesman_id=5001
```

==========

----> Create table item_mast

==========

```
CREATE TABLE item_mast
(
        PRO_ID int PRIMARY KEY,
        PRO_NAME varchar(40),
        PRO_PRICE float,
        PRO_COM int
)
```

========== **PROCEDURE START** ==========

```
DELIMITER $$
CREATE PROCEDURE insert_data(i int, j varchar(40), k float, l int)
BEGIN
        INSERT INTO item_mast(PRO_ID, PRO_NAME, PRO_PRICE, PRO_COM) VALUES(i,j,k,l);
END
```

```
CALL insert_data(101,"Mother Board",3200.00,15);
CALL insert_data(102,"Key Board",450.00,16);
CALL insert_data(103,"Zip Drive",250.00,14);
CALL insert_data(104,"Speaker",550.00,16);
CALL insert_data(105,"Monitor",5000.00,11);
CALL insert_data(106,"DVD drive",900.00,12);
CALL insert_data(107,"CD drive",800.00,12);
CALL insert_data(108,"Printer",2600.00,13);
CALL insert_data(109,"Refill Cartridge",350.00,13);
CALL insert_data(110,"Mouse",250.00,12);
```

========== **PROCEDURE END** ==========

16) From the following table, write a SQL query to select a range of products whose price is in the range Rs.200 to Rs.600. Begin and end values are included. Return pro_id, pro_name, pro_price, and pro_com.

```
SELECT * FROM item_mast WHERE PRO_PRICE BETWEEN 200 AND 600
```

17) From the following table, write a SQL query to calculate the average price for a manufacturer code of 16. Return avg.

```
SELECT AVG(PRO_PRICE) FROM item_mast WHERE PRO_COM=16
```

18) From the following table, write a SQL query to display the pro_name as 'Item Name' and pro_priceas 'Price in Rs.'

```
SELECT PRO_NAME AS Item_Name, PRO_PRICE AS Price_in_Rs FROM item_mast
```

19) From the following table, write a SQL query to find the items whose prices are higher than or equal to $250. Order the result by product price in descending, then product name in ascending. Return pro_name and pro_price.

```
SELECT PRO_NAME,PRO_PRICE FROM item_mast WHERE PRO_PRICE>=250
SELECT PRO_NAME FROM item_mast ORDER BY PRO_NAME ASC
SELECT PRO_PRICE FROM item_mast ORDER BY PRO_PRICE DESC
```

20) From the following table, write a SQL query to calculate average price of the items for each company. Return average price and company code.

```
SELECT AVG(PRO_PRICE), PRO_COM FROM item_mast WHERE PRO_COM=11
SELECT AVG(PRO_PRICE), PRO_COM FROM item_mast WHERE PRO_COM=12
SELECT AVG(PRO_PRICE), PRO_COM FROM item_mast WHERE PRO_COM=13
SELECT AVG(PRO_PRICE), PRO_COM FROM item_mast WHERE PRO_COM=14
SELECT AVG(PRO_PRICE), PRO_COM FROM item_mast WHERE PRO_COM=15
SELECT AVG(PRO_PRICE), PRO_COM FROM item_mast WHERE PRO_COM=16
```