Module: 5 Database

• Topics Covered Basics of Database

1. What do you understand By Database

- **A database is a structured collection of data that is organized and stored in a way that allows for efficient retrieval, management, and updating of information.**
- **The** Database **is an essential part of our life. We encounter several activities that involve our interaction with databases, for example in the bank, in the railway station, in school, in a grocery store, etc. These are the instances where we need to store a large amount of data in one place and fetch these data easily.**

2. What is Normalization?

- **Normalization is a process used in database design to organize tables and minimize redundancy and dependency by dividing large tables into smaller, related tables. Its primary goal is to structure the data in such a way that it reduces redundancy and anomalies when data is inserted, updated, or deleted.**
- Normalization **is the process of minimizing** redundancy **from a relation or set of relations. Redundancy in relation may cause insertion, deletion, and update anomalies. So, it helps to minimize the redundancy in relations.** Normal forms **are used to eliminate or reduce redundancy in database tables.**

3. What is Difference between DBMS and RDBMS?

- DBMS:

- **Data stored is in file format**
- **Individual access of data element**
- **No connection between data**
- **No support for distributed database**
- **Data stored is a small quantity**
- **DBMS support a single user**

- The software and hardware requirements are low
- Example: - XML, Microsoft Assess

➢ RDBMS:

- Relation database management system.
- Data Stored is in table format.
- Multiple data element is accessible together.
- Data in the form of a table are linked together.
- Support distributed database.
- Data is Stored in large amount.
- RDBMS supports multiple users.
- The software and hardware requirement are higher.
- Example: - Oracle, SQL, Server.

4. What is MF Cod Rule of RDBMS Systems?

➢ The MF Cod Rule of RDBMS Systems states that for a system to qualify as an RDBMS, it must be able to manage database entirely through the relational capabilities . Rule 0 of the MF Cod Rules states that the system must qualify as relational, as a database, and as a management system.

➢ a set of thirteen rules (numbered 0 to 12) that define a database to be a correct Relational Database Management System (RDBMS).

5. What do you understand by Data Redundancy?

➢ Data redundancy refers to the unnecessary repetition or duplication of data within a database or across different databases or systems. It occurs when the same piece of data is stored in multiple places. This redundancy can lead to several issues:

- Increased Storage Requirements
- Inconsistency
- Update Anomalies

- Decreased Performance

- Difficulty in Data Management

6. What is DDL Interpreter?

- ➢ A DDL (Data Definition Language) Interpreter is a component of a Database Management System (DBMS) that processes and executes Data Definition Language commands. DDL commands are used to define and manage the structure of databases and database objects such as tables, indexes, views, and schemas.
- ➢ It interprets the DDL (Data Definition Language) Instructions and stores the record in a data dictionary (in a table containing meta-data)

7. What is DML Compiler in SQL?

- ➢ The Data Manipulation Language, or DML for short, is the group of commands responsible for manipulating data in a database; this generally entails inserting, editing, or deleting rows in SQL tables.
- ➢ DML Compiler again as the name suggests compiles(or translates) the DML statements such as select, update, and delete statements into low-level instructions

- Query Parser

- Query Optimizer

- Execution Engine

8. What is SQL Key Constraints writing an Example of SQL Key Constraints

- ➢ Constraints are the rules that we can apply on the type of data in a table. That is, we can specify the limit on the type of data that can be stored in a particular column in a table using constraints.

- NOT NULL: This constraint tells that we cannot store a null value in a column. That is, if a column is specified as NOT NULL then we will not be able to store null in this particular column any more.

- **UNIQUE: This constraint when specified with a column, tells that all the values in the column must be unique. That is, the values in any row of a column must not be repeated.**

- **PRIMARY KEY: Database Tirth Patel 4 A primary key is a field which can uniquely identify each row in a table. And this constraint is used to specify a field in a table as primary key.**

- **FOREIGN KEY: A Foreign key is a field which can uniquely identify each row in another table. And this constraint is used to specify a field as foreign key.**

- **CHECK: This constraint helps to validate the values of a column to meet a particular condition. That is, it helps to ensure that the value stored in a column meets a specific condition.**

- **DEFAULT: This constraint specifies a default value for the column when no value is specified by the user.**

9. What is save Point? How to create a save Point write a Query?

- **A save point in SQL is a logical rollback point within a transaction.**

It allows you to specify a point in a transaction that you can roll back to without affecting the entire transaction.

Syntax: 'SAVEPOINT savepoint_name'

You can then perform various SQL operations Within the transaction. To roll back to a specific save point use 'ROLLBACK TO save_point_name'

10. What is trigger and how to create a Trigger in SQL?

A trigger in SQL is a special type of stored procedure that automatically executes in response to certain events on a particular table or view in a database. These events can include INSERT, UPDATE, DELETE operations or a combination thereof.

a special type of stored procedure that automatically runs when an event occurs in the database server.

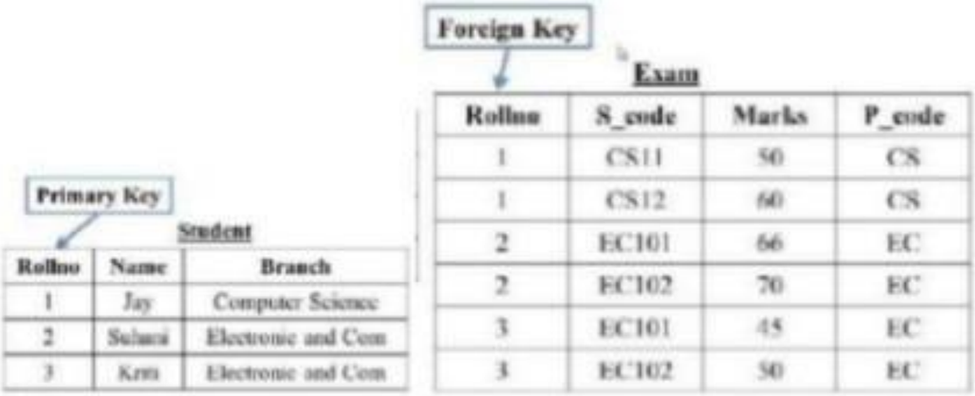
Example:

We are adding tuple to the ‘Donors’ table that is some Person has donated blood. So we can design a trigger that will automatically add the value of donated blood to the ‘blood_record’ table

We can define 6 types of triggers for each table:

- **AFTER INSERT: activated after data is inserted into the table.**
- **AFTER UPDATE: activated after data in the table is modified.**
- **AFTER DELETE: activated after data is deleted/removed from the table.**
- **BEFORE INSERT: activated before data is inserted into the table.**
- **BEFORE UPDATE: activated before data in the table is modified.**
- **BEFORE DELETE: activated before data is deleted/removed from the table.**

1. Create Table Name: Student and Exam



Foreign Key
Primary Key

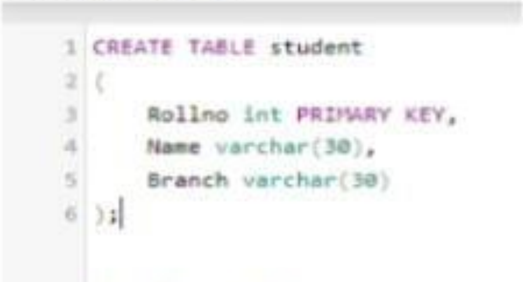
Exam

Rollno	S_code	Marks	P_code
1	CS11	50	CS
1	CS12	60	CS
2	EC101	66	EC
2	EC102	70	EC
3	EC101	45	EC
3	EC102	50	EC

Student

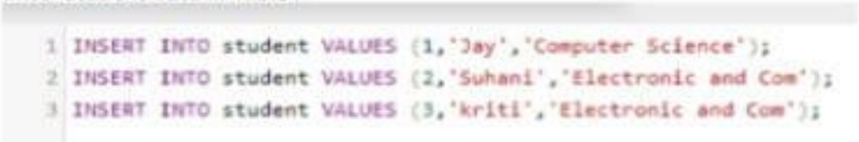
Rollno	Name	Branch
1	Jay	Computer Science
2	Suhani	Electronic and Com
3	Kriti	Electronic and Com

Create Table Student :



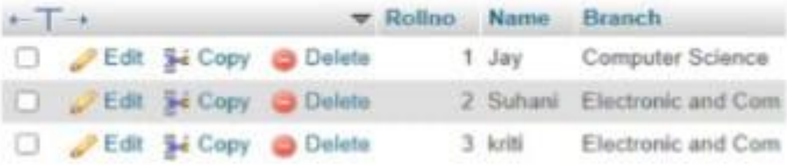
```
CREATE TABLE student
(
    Rollno int PRIMARY KEY,
    Name varchar(30),
    Branch varchar(30)
);
```

Enter Student Data in Table :



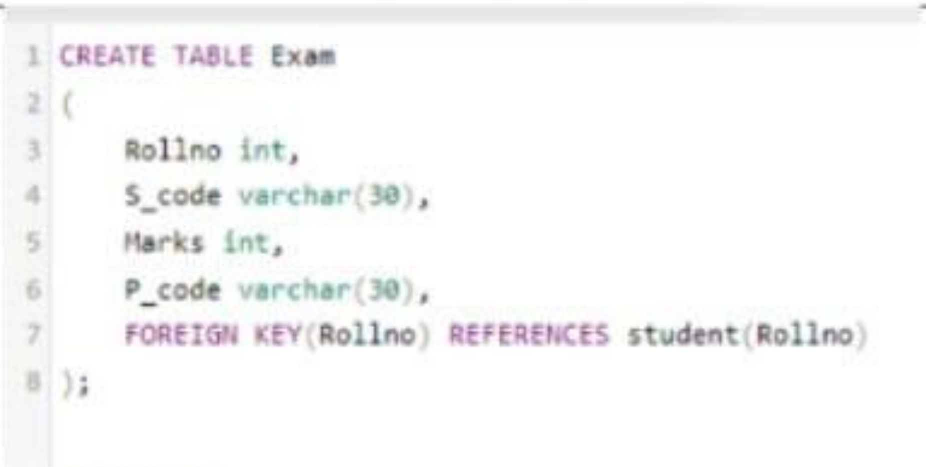
```
INSERT INTO student VALUES (1,'Jay','Computer Science');
INSERT INTO student VALUES (2,'Suhani','Electronic and Com');
INSERT INTO student VALUES (3,'kriti','Electronic and Com');
```

Student Table:



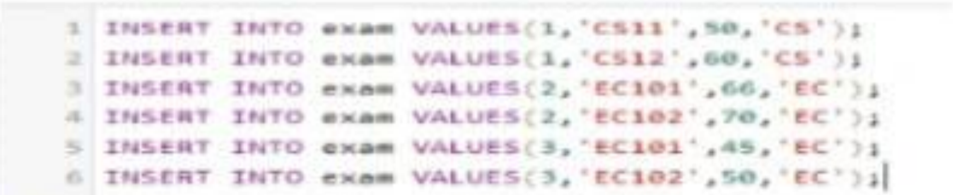
←T→	Rollno	Name	Branch
□ Edit Copy Delete	1	Jay	Computer Science
□ Edit Copy Delete	2	Suhani	Electronic and Com
□ Edit Copy Delete	3	kriti	Electronic and Com

Create Table Exam:



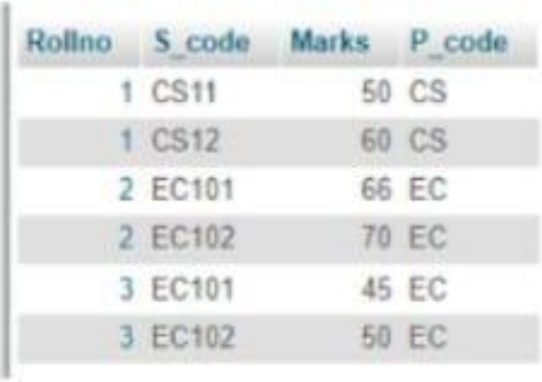
```
CREATE TABLE Exam
(
    Rollno int,
    S_code varchar(30),
    Marks int,
    P_code varchar(30),
    FOREIGN KEY(Rollno) REFERENCES student(Rollno)
);
```

Enter Data in Table :



```
INSERT INTO exam VALUES(1,'CS11',50,'CS');
INSERT INTO exam VALUES(1,'CS12',60,'CS');
INSERT INTO exam VALUES(2,'EC101',66,'EC');
INSERT INTO exam VALUES(2,'EC102',70,'EC');
INSERT INTO exam VALUES(3,'EC101',45,'EC');
INSERT INTO exam VALUES(3,'EC102',50,'EC');
```

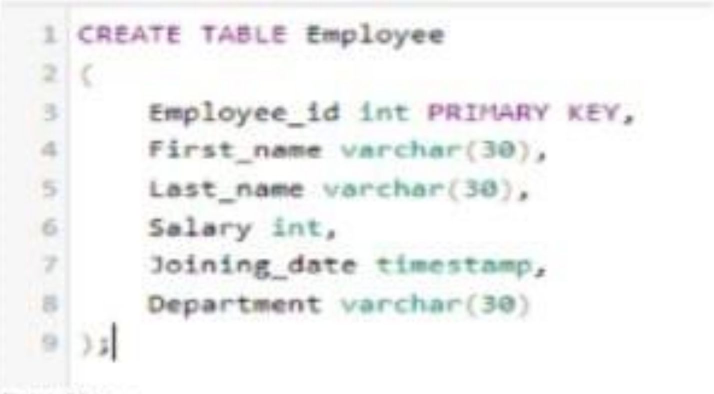
Exam Table :



Rollno	S_code	Marks	P_code
1	CS11	50	CS
1	CS12	60	CS
2	EC101	66	EC
2	EC102	70	EC
3	EC101	45	EC
3	EC102	50	EC

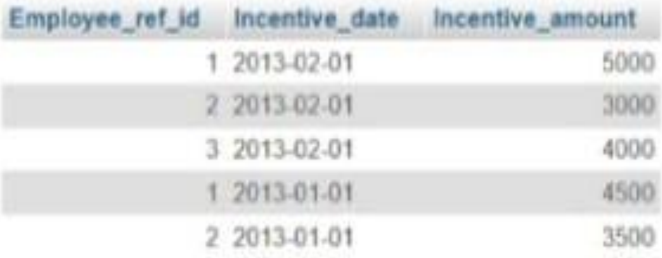
2 Create table given below: Employee and Incentive Table.

Create Table Employee:



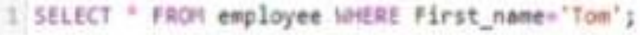
```
CREATE TABLE Employee
(
    Employee_id int PRIMARY KEY,
    First_name varchar(30),
    Last_name varchar(30),
    Salary int,
    Joining_date timestamp,
    Department varchar(30)
);
```

Enter Data :

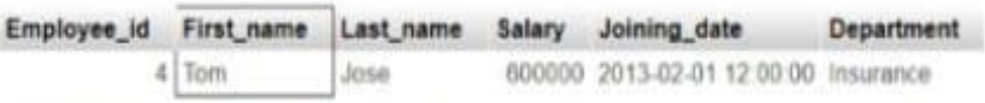


Employee_ref_id	Incentive_date	Incentive_amount
1	2013-02-01	5000
2	2013-02-01	3000
3	2013-02-01	4000
1	2013-01-01	4500
2	2013-01-01	3500

3. **Get First_Name from employee table using Tom name "Employee Name"**

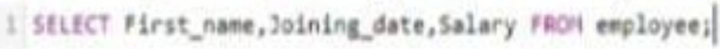


```
SELECT * FROM employee WHERE First_name='Tom';
```

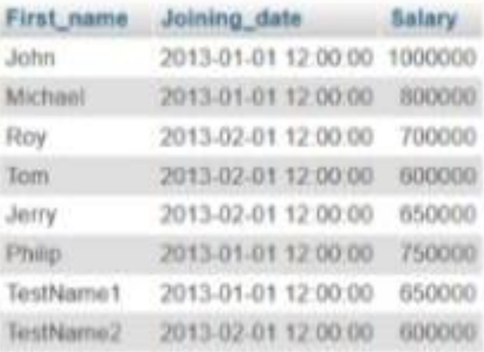


Employee_id	First_name	Last_name	Salary	Joining_date	Department
4	Tom	Jose	600000	2013-02-01 12:00:00	Insurance

4 **Get FIRST_NAME, Joining Date, and Salary from employee table.**

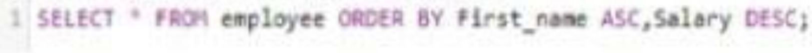


```
SELECT First_name,Joining_date,Salary FROM employee;
```

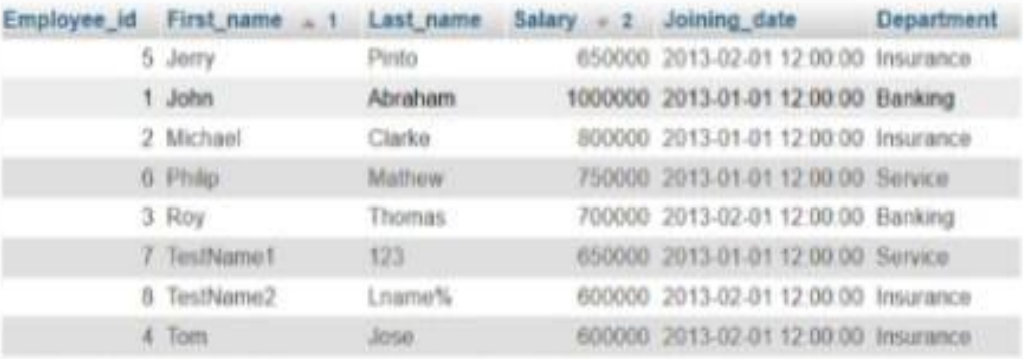


First_name	Joining_date	Salary
John	2013-01-01 12:00:00	1000000
Michael	2013-01-01 12:00:00	800000
Roy	2013-02-01 12:00:00	700000
Tom	2013-02-01 12:00:00	600000
Jerry	2013-02-01 12:00:00	650000
Philip	2013-01-01 12:00:00	750000
TestName1	2013-01-01 12:00:00	650000
TestName2	2013-02-01 12:00:00	600000

5 **Get all employee details from the employee table order by First_Name Ascending and Salary descending?**

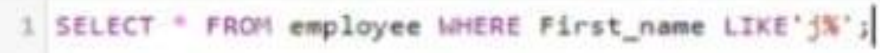


```
SELECT * FROM employee ORDER BY First_name ASC,Salary DESC;
```

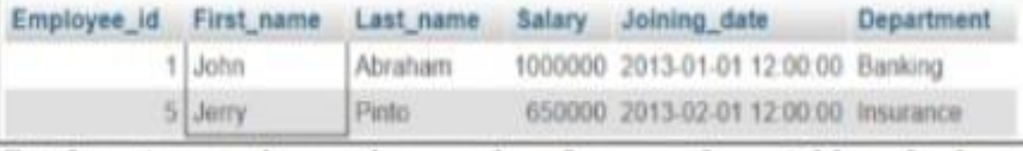


Employee_id	First_name ▲ 1	Last_name	Salary ▼ 2	Joining_date	Department
5	Jerry	Pinto	650000	2013-02-01 12:00:00	Insurance
1	John	Abraham	1000000	2013-01-01 12:00:00	Banking
2	Michael	Clarke	800000	2013-01-01 12:00:00	Insurance
6	Philip	Mathew	750000	2013-01-01 12:00:00	Service
3	Roy	Thomas	700000	2013-02-01 12:00:00	Banking
7	TestName1	123	650000	2013-01-01 12:00:00	Service
8	TestName2	Lname%	600000	2013-02-01 12:00:00	Insurance
4	Tom	Jose	600000	2013-02-01 12:00:00	Insurance

6 **Get employee details from employee table whose first name contains 'J'.**

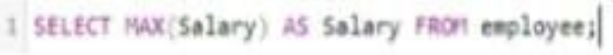


```
SELECT * FROM employee WHERE First_name LIKE'j%';
```

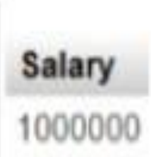


Employee_id	First_name	Last_name	Salary	Joining_date	Department
1	John	Abraham	1000000	2013-01-01 12:00:00	Banking
5	Jerry	Pinto	650000	2013-02-01 12:00:00	Insurance

7 **Get department wise maximum salary from employee table order by**

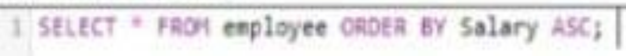


```sql
SELECT MAX(Salary) AS Salary FROM employee;
```

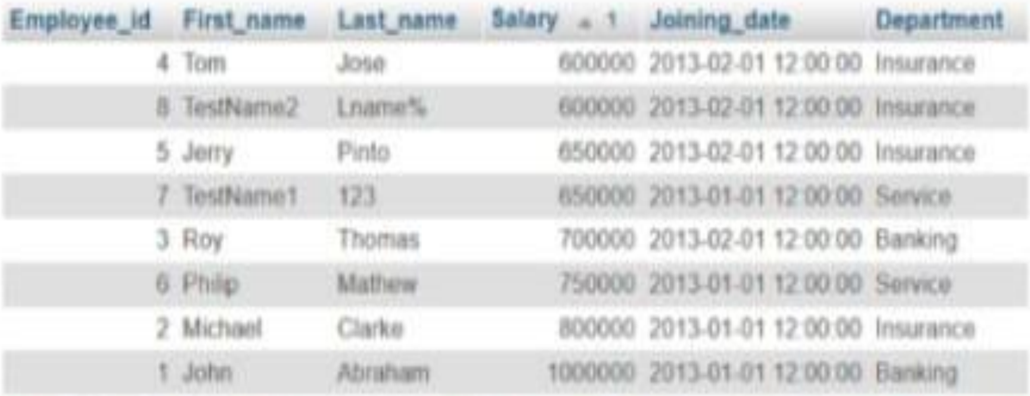


Salary
1000000

8 **salary ascending?**

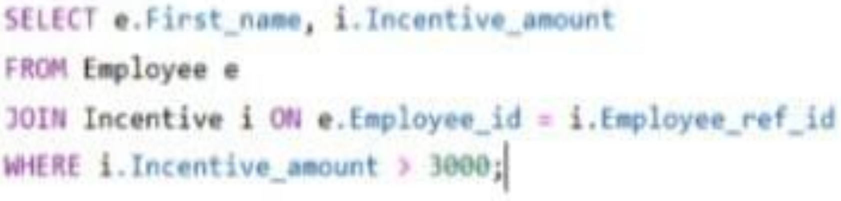


```sql
SELECT * FROM employee ORDER BY Salary ASC;
```

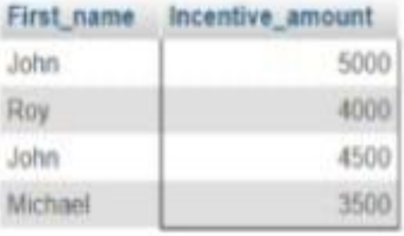


Employee_id	First_name	Last_name	Salary ▲ 1	Joining_date	Department
4	Tom	Jose	600000	2013-02-01 12:00:00	Insurance
8	TestName2	Lname%	600000	2013-02-01 12:00:00	Insurance
5	Jerry	Pinto	650000	2013-02-01 12:00:00	Insurance
7	TestName1	123	650000	2013-01-01 12:00:00	Service
3	Roy	Thomas	700000	2013-02-01 12:00:00	Banking
6	Philip	Mathew	750000	2013-01-01 12:00:00	Service
2	Michael	Clarke	800000	2013-01-01 12:00:00	Insurance
1	John	Abraham	1000000	2013-01-01 12:00:00	Banking

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

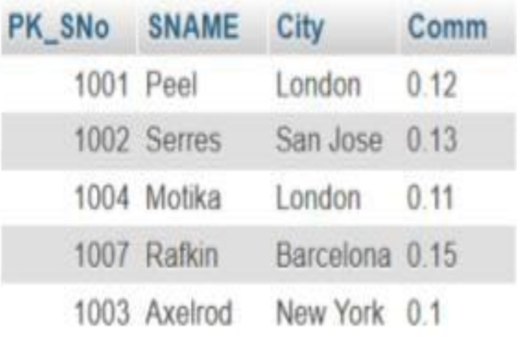


```sql
SELECT e.First_name, i.Incentive_amount
FROM Employee e
JOIN Incentive i ON e.Employee_id = i.Employee_ref_id
WHERE i.Incentive_amount > 3000;
```



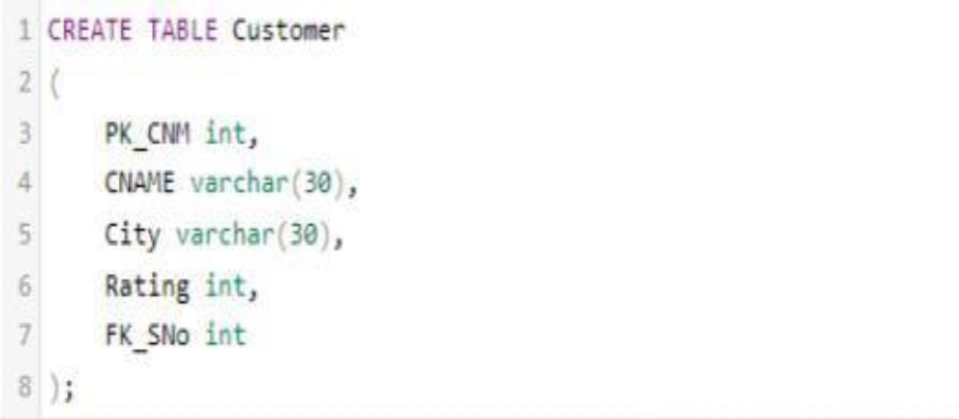
First_name	Incentive_amount
John	5000
Roy	4000
John	4500
Michael	3500

10 **Create After Insert trigger on Employee table which insert records in viewable**

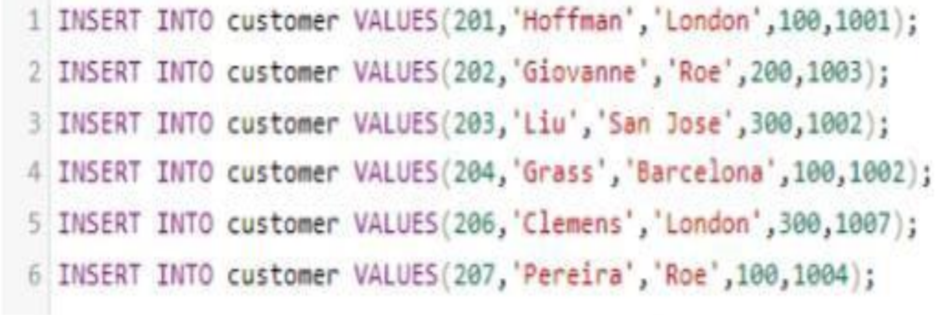


PK_SNo	SNAME	City	Comm
1001	Peel	London	0.12
1002	Serres	San Jose	0.13
1004	Motika	London	0.11
1007	Rafkin	Barcelona	0.15
1003	Axelrod	New York	0.1

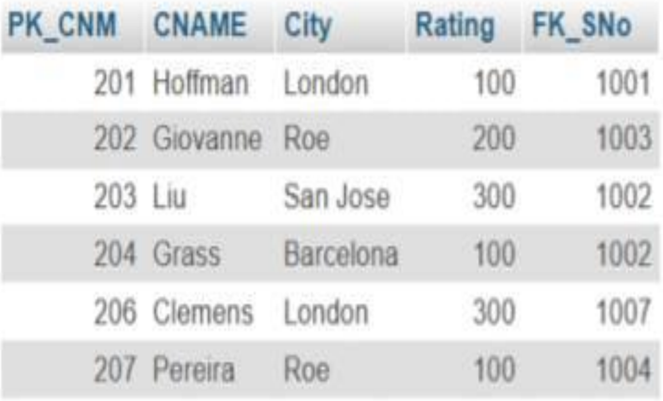
Create Table 2: Customer



```
CREATE TABLE Customer
(
    PK_CNM int,
    CNAME varchar(30),
    City varchar(30),
    Rating int,
    FK_SNo int
);
```



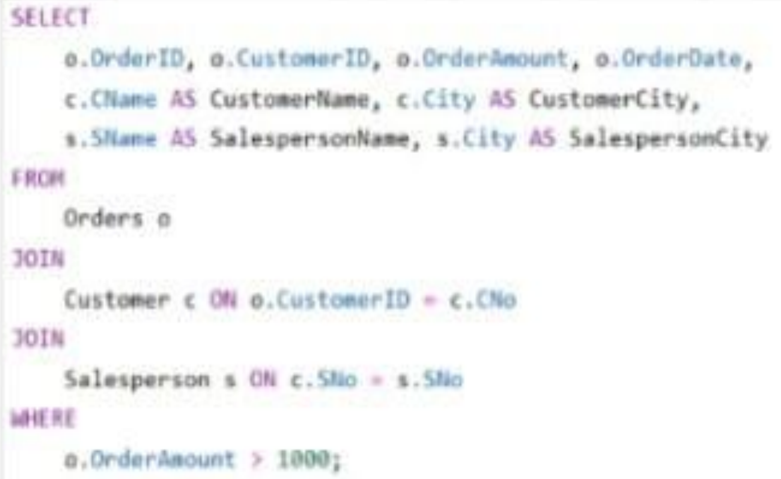
```
INSERT INTO customer VALUES(201,'Hoffman','London',100,1001);
INSERT INTO customer VALUES(202,'Giovanne','Roe',200,1003);
INSERT INTO customer VALUES(203,'Liu','San Jose',300,1002);
INSERT INTO customer VALUES(204,'Grass','Barcelona',100,1002);
INSERT INTO customer VALUES(206,'Clemens','London',300,1007);
INSERT INTO customer VALUES(207,'Pereira','Roe',100,1004);
```



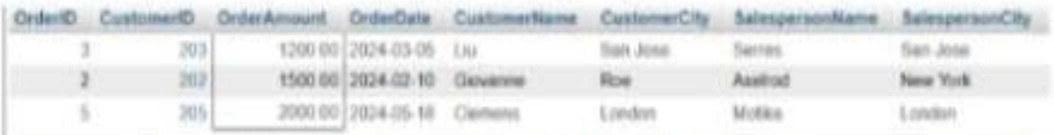
PK_CNM	CNAME	City	Rating	FK_SNo
201	Hoffman	London	100	1001
202	Giovanne	Roe	200	1003
203	Liu	San Jose	300	1002
204	Grass	Barcelona	100	1002
206	Clemens	London	300	1007
207	Pereira	Roe	100	1004

12 **Retrieve the below data from above table**

13 **All orders for more than $1000.**

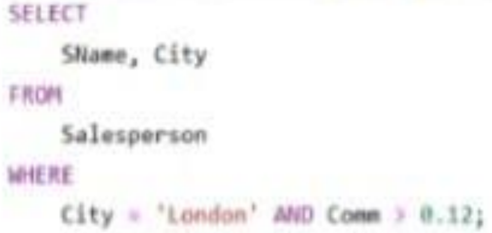


```
SELECT
    o.OrderID, o.CustomerID, o.OrderAmount, o.OrderDate,
    c.CName AS CustomerName, c.City AS CustomerCity,
    s.SName AS SalespersonName, s.City AS SalespersonCity
FROM
    Orders o
JOIN
    Customer c ON o.CustomerID = c.CNo
JOIN
    Salesperson s ON c.SNo = s.SNo
WHERE
    o.OrderAmount > 1000;
```

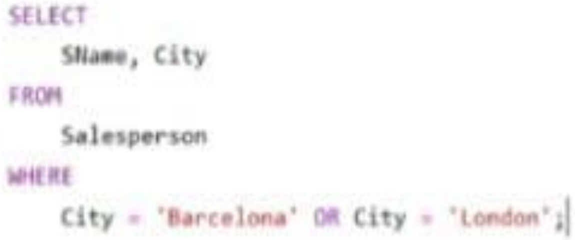


OrderID	CustomerID	OrderAmount	OrderDate	CustomerName	CustomerCity	SalespersonName	SalespersonCity
3	203	1200.00	2024-03-05	Liu	San Jose	Serres	San Jose
2	202	1500.00	2024-02-10	Giovanne	Roe	Axelrod	New York
5	205	2000.00	2024-05-18	Clemens	London	Motika	London

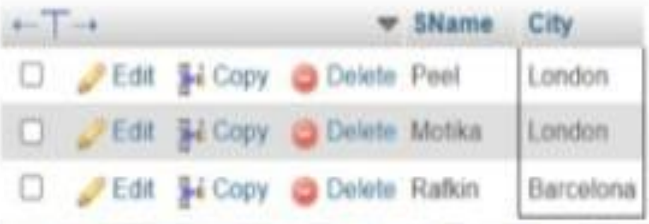
14 **Names and cities of all salespeople in London with commission above 0.12**



```
SELECT
    SName, City
FROM
    Salesperson
WHERE
    City = 'London' AND Comm > 0.12;
```

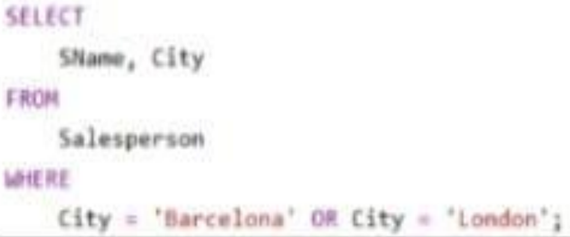


```
SELECT
    SName, City
FROM
    Salesperson
WHERE
    City = 'Barcelona' OR City = 'London';
```

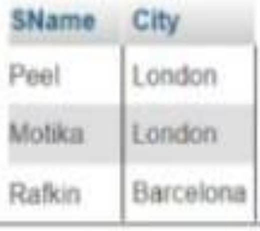


				SName	City
□	Edit	Copy	Delete	Peel	London
□	Edit	Copy	Delete	Motika	London
□	Edit	Copy	Delete	Rafkin	Barcelona

15 **All salespeople either in Barcelona or in London**

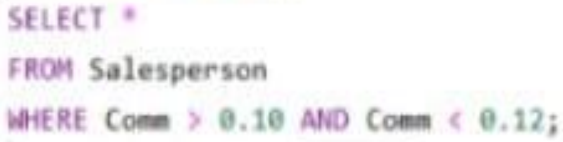


```
SELECT
    SName, City
FROM
    Salesperson
WHERE
    City = 'Barcelona' OR City = 'London';
```

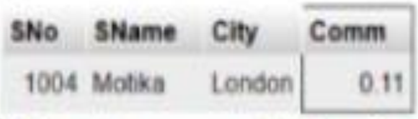


SName	City
Peel	London
Motika	London
Rafkin	Barcelona

16 **All salespeople with commission between 0.10 and 0.12. (Boundary values should be excluded).**

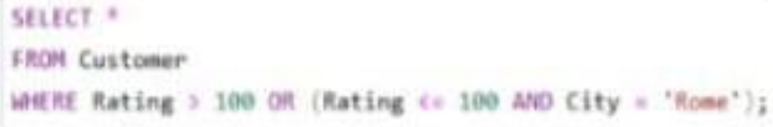


```sql
SELECT *
FROM Salesperson
WHERE Comm > 0.10 AND Comm < 0.12;
```

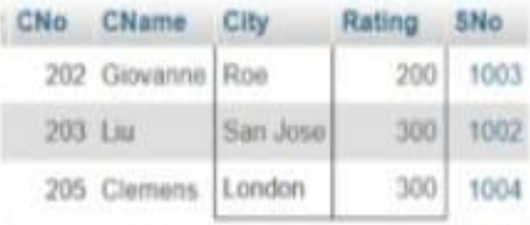


SNo	SName	City	Comm
1004	Motika	London	0.11

17 **All customers excluding those with rating <= 100 unless they are located in Rome**

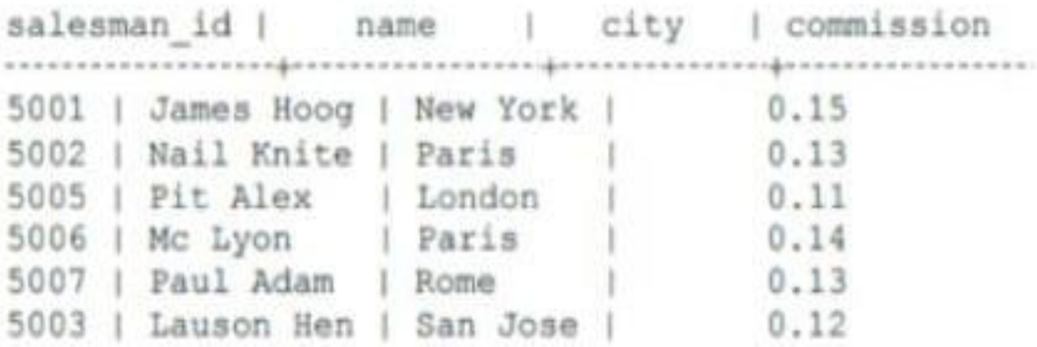


```sql
SELECT *
FROM Customer
WHERE Rating > 100 OR (Rating <= 100 AND City = 'Rome');
```



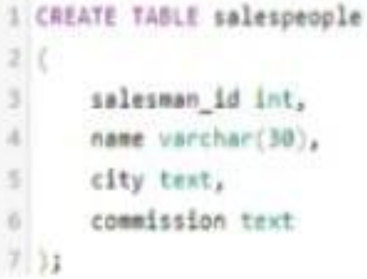
CNo	CName	City	Rating	SNo
202	Giovanne	Roe	200	1003
203	Liu	San Jose	300	1002
205	Clemens	London	300	1004

18 **Write a SQL statement that displays all the information about all salespeople**

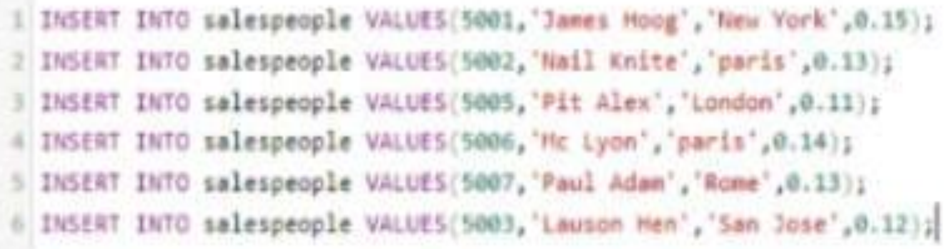


```
salesman_id |    name    |   city    | commission
------------+------------+-----------+------------
5001 | James Hoog | New York |        0.15
5002 | Nail Knite | Paris    |        0.13
5005 | Pit Alex   | London   |        0.11
5006 | Mc Lyon    | Paris    |        0.14
5007 | Paul Adam  | Rome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

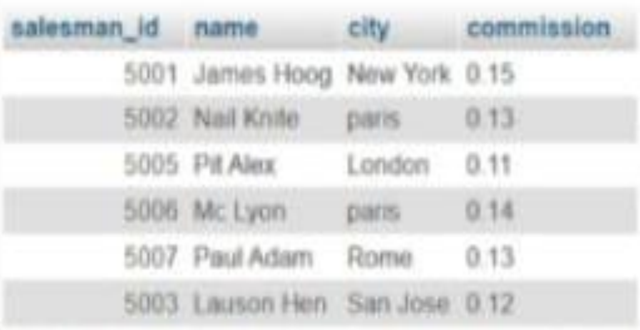
Create Table Salespeople



```sql
CREATE TABLE salespeople
(
    salesman_id int,
    name varchar(30),
    city text,
    commission text
);
```



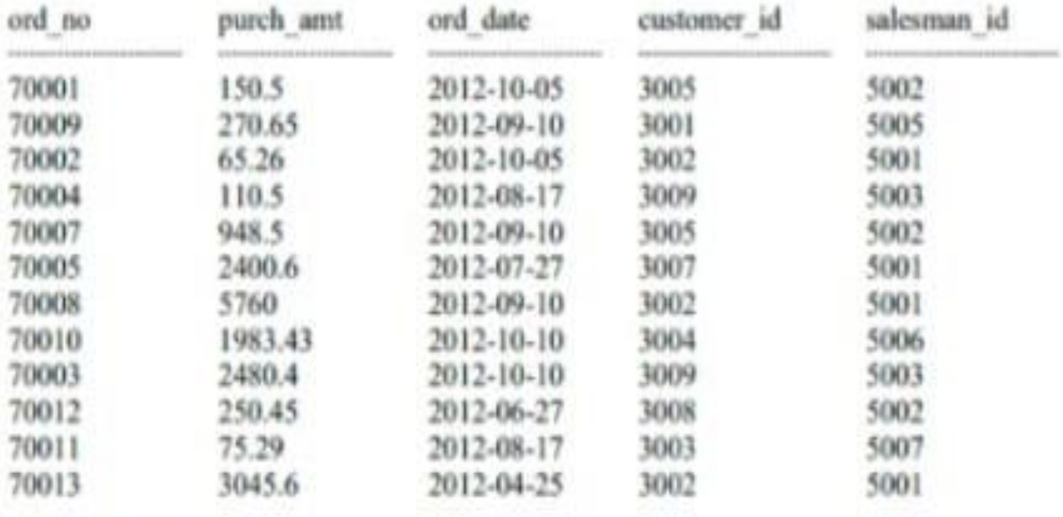
```sql
INSERT INTO salespeople VALUES(5001,'James Hoog','New York',0.15);
INSERT INTO salespeople VALUES(5002,'Nail Knite','paris',0.13);
INSERT INTO salespeople VALUES(5005,'Pit Alex','London',0.11);
INSERT INTO salespeople VALUES(5006,'Mc Lyon','paris',0.14);
INSERT INTO salespeople VALUES(5007,'Paul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



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

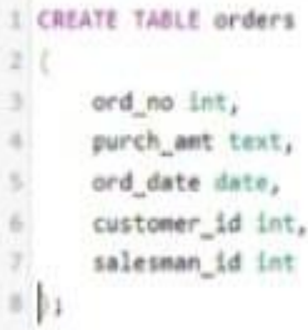
19 **From the following table, write a SQL query to find orders that are delivered by a salesperson with ID. 5001. Return ord_no, ord_date, purch_amt.**

Sample table: orders

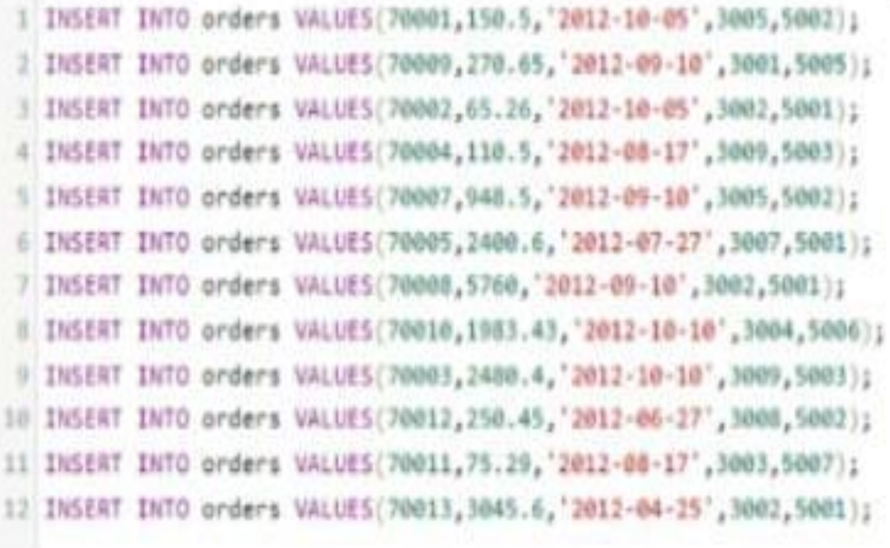


ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	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

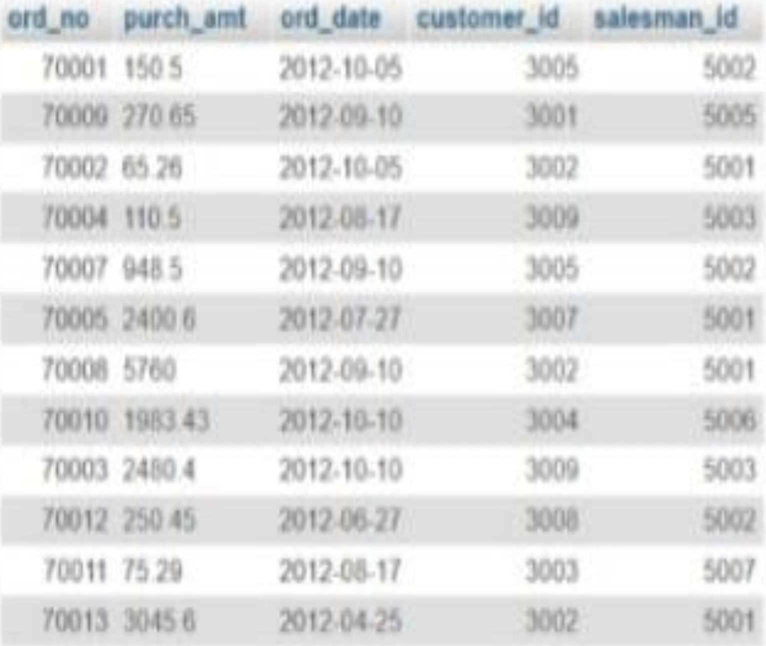
Create Table Orders



```sql
CREATE TABLE orders
(
    ord_no int,
    purch_amt text,
    ord_date date,
    customer_id int,
    salesman_id int
);
```

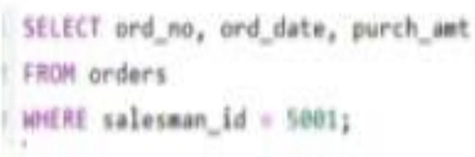


```sql
INSERT INTO orders VALUES(70001,150.5,'2012-10-05',3005,5002);
INSERT INTO orders VALUES(70009,270.65,'2012-09-10',3001,5005);
INSERT INTO orders VALUES(70002,65.26,'2012-10-05',3002,5001);
INSERT INTO orders VALUES(70004,110.5,'2012-08-17',3009,5003);
INSERT INTO orders VALUES(70007,948.5,'2012-09-10',3005,5002);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

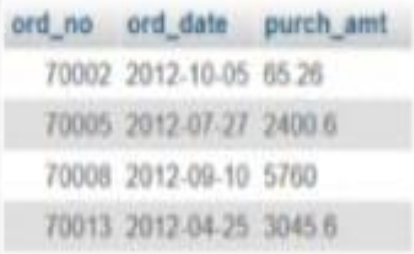


ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.5	2012-10-05	3005	5002
70009	270.65	2012-09-10	3001	5005
70002	65.26	2012-10-05	3002	5001
70004	110.5	2012-08-17	3009	5003
70007	948.5	2012-09-10	3005	5002
70005	2400.6	2012-07-27	3007	5001
70008	5760	2012-09-10	3002	5001
70010	1983.43	2012-10-10	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

Query:

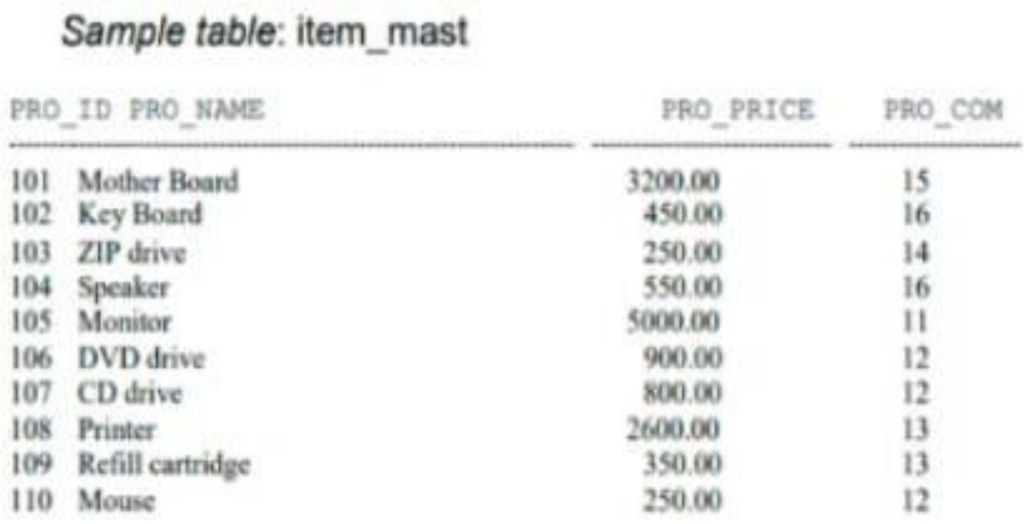


```sql
SELECT ord_no, ord_date, purch_amt
FROM orders
WHERE salesman_id = 5001;
```



ord_no	ord_date	purch_amt
70002	2012-10-05	65.26
70005	2012-07-27	2400.6
70008	2012-09-10	5760
70013	2012-04-25	3045.6

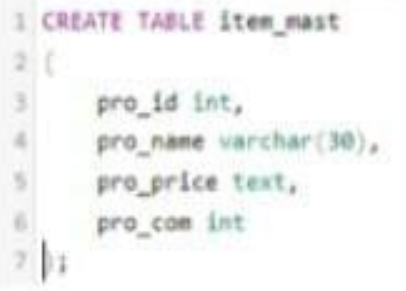
20 **From the following table, write a SQL query to select a range of products whose price is in the range Rs.200 to Rs.600. Begin and end values are included. Return pro_id, pro_name, pro_price, and pro_com.**



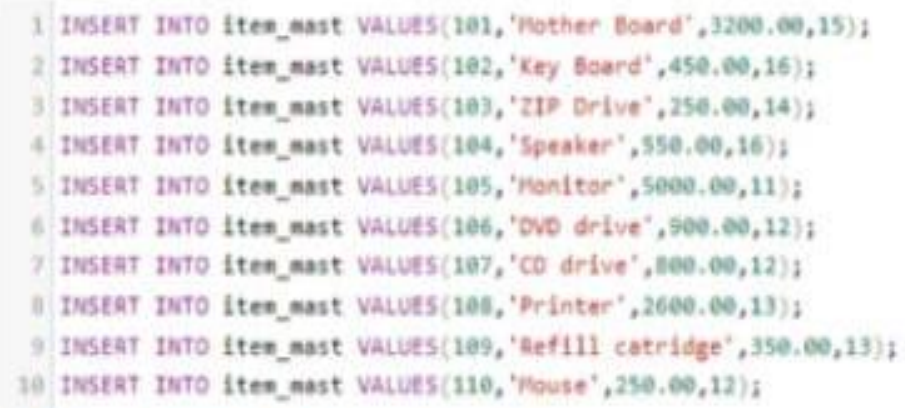
Sample table: item_mast

PRO_ID	PRO_NAME	PRO_PRICE	PRO_COM
101	Mother Board	3200.00	15
102	Key Board	450.00	16
103	ZIP drive	250.00	14
104	Speaker	550.00	16
105	Monitor	5000.00	11
106	DVD drive	900.00	12
107	CD drive	800.00	12
108	Printer	2600.00	13
109	Refill cartridge	350.00	13
110	Mouse	250.00	12

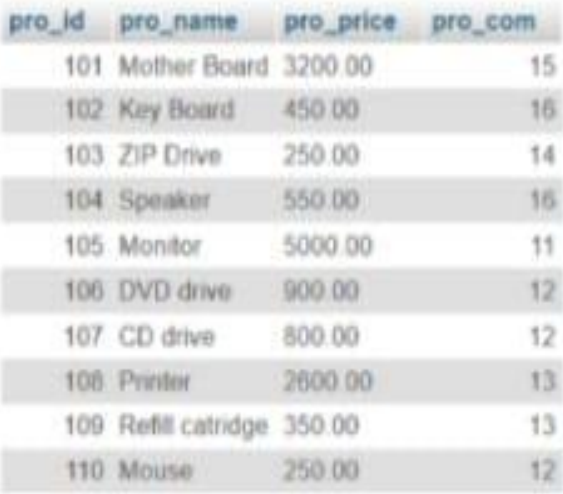
Create Table Item_mast



```
CREATE TABLE item_mast
(
    pro_id int,
    pro_name varchar(30),
    pro_price text,
    pro_com int
);
```

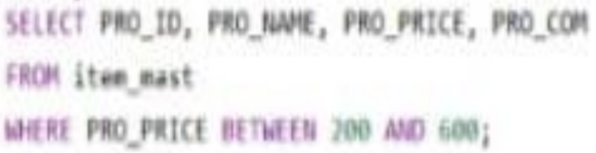


```
INSERT INTO item_mast VALUES(101,'Mother Board',3200.00,15);
INSERT INTO item_mast VALUES(102,'Key Board',450.00,16);
INSERT INTO item_mast VALUES(103,'ZIP Drive',250.00,14);
INSERT INTO item_mast VALUES(104,'Speaker',550.00,16);
INSERT INTO item_mast VALUES(105,'Monitor',5000.00,11);
INSERT INTO item_mast VALUES(106,'DVD drive',900.00,12);
INSERT INTO item_mast VALUES(107,'CD drive',800.00,12);
INSERT INTO item_mast VALUES(108,'Printer',2600.00,13);
INSERT INTO item_mast VALUES(109,'Refill catridge',350.00,13);
INSERT INTO item_mast VALUES(110,'Mouse',250.00,12);
```

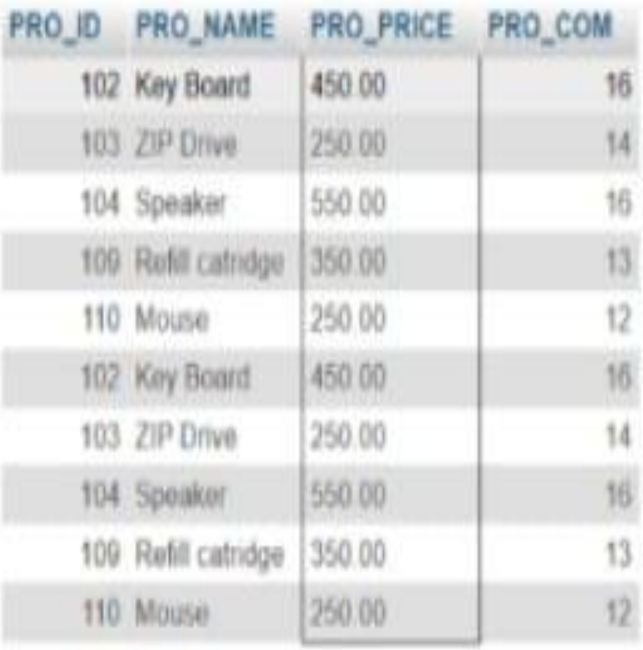


pro_id	pro_name	pro_price	pro_com
101	Mother Board	3200.00	15
102	Key Board	450.00	16
103	ZIP Drive	250.00	14
104	Speaker	550.00	16
105	Monitor	5000.00	11
106	DVD drive	900.00	12
107	CD drive	800.00	12
108	Printer	2600.00	13
109	Refill catridge	350.00	13
110	Mouse	250.00	12

Query:



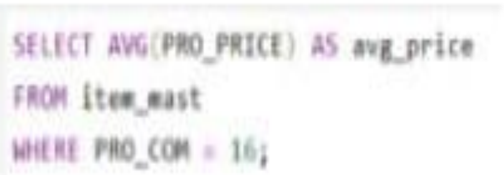
```
SELECT PRO_ID, PRO_NAME, PRO_PRICE, PRO_COM
FROM item_mast
WHERE PRO_PRICE BETWEEN 200 AND 600;
```



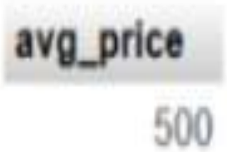
PRO_ID	PRO_NAME	PRO_PRICE	PRO_COM
102	Key Board	450.00	16
103	ZIP Drive	250.00	14
104	Speaker	550.00	16
109	Refill catridge	350.00	13
110	Mouse	250.00	12
102	Key Board	450.00	16
103	ZIP Drive	250.00	14
104	Speaker	550.00	16
109	Refill catridge	350.00	13
110	Mouse	250.00	12

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

Query:



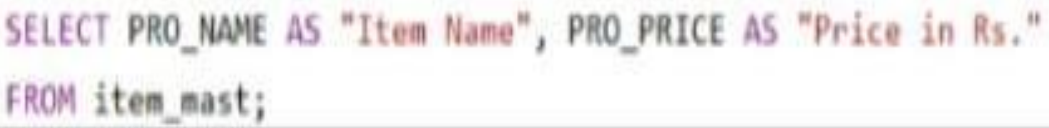
```
SELECT AVG(PRO_PRICE) AS avg_price
FROM item_mast
WHERE PRO_COM = 16;
```



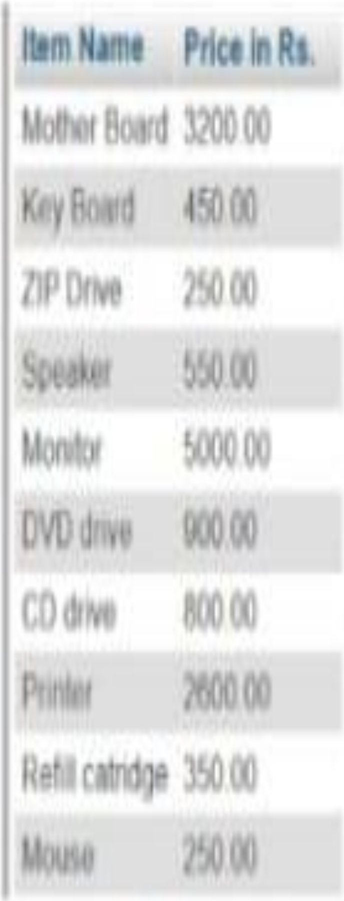
avg_price
500

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

Query:



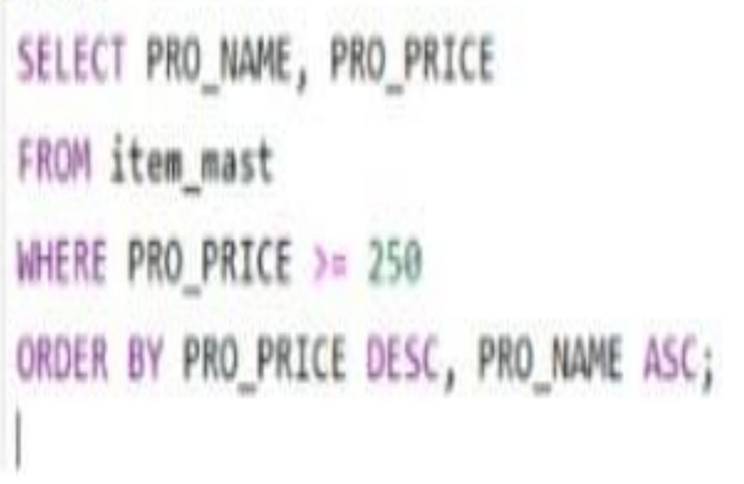
```
SELECT PRO_NAME AS "Item Name", PRO_PRICE AS "Price in Rs."
FROM item_mast;
```



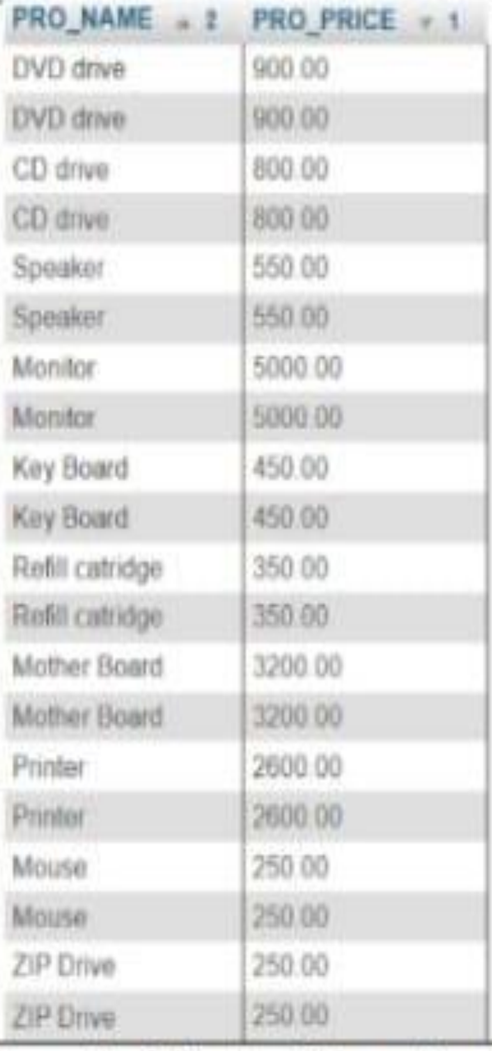
Item Name	Price in Rs.
Mother Board	3200.00
Key Board	450.00
ZIP Drive	250.00
Speaker	550.00
Monitor	5000.00
DVD drive	900.00
CD drive	800.00
Printer	2600.00
Refill catridge	350.00
Mouse	250.00

23 **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.**

Query:



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



PRO_NAME ▲ 2	PRO_PRICE ▼ 1
DVD drive	900.00
DVD drive	900.00
CD drive	800.00
CD drive	800.00
Speaker	550.00
Speaker	550.00
Monitor	5000.00
Monitor	5000.00
Key Board	450.00
Key Board	450.00
Refill catridge	350.00
Refill catridge	350.00
Mother Board	3200.00
Mother Board	3200.00
Printer	2600.00
Printer	2600.00
Mouse	250.00
Mouse	250.00
ZIP Drive	250.00
ZIP Drive	250.00

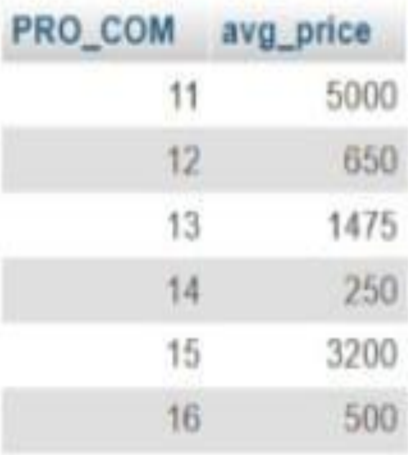
24 **From the following table, write a SQL query to calculate average price of the items for each company. Return average price and company code.**

Query:

```
SELECT PRO_COM, AVG(PRO_PRICE) AS avg_price

FROM item_mast

GROUP BY PRO_COM;
```



PRO_COM	avg_price
11	5000
12	650
13	1475
14	250
15	3200
16	500