MODULE: 5 (Database)

1. What do you understand By Database.

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. It is a powerful tool for storing and managing information, and it is used by businesses, organizations.

- Storing data : Databases provide a centralized location for storing data, which can be accessed and updated easily.
- Analyzing data : Databases can be used to analyze data, identify trends, and make informed decisions.
- Sharing data : Databases can be used to share data with other users, both within an organization and externally.

Features of databases :

- Tables : Data is organized into tables, which consist of rows and columns.
- Relationships : Tables can be related to each other, allowing for complex data structures.
- Queries : Queries allow you to retrieve and manipulate data from the database.
- Security : Databases have security features to protect data from unauthorized access.
- Relational databases : These databases store data in tables with relationships between them. They are a popular choice for business applications.
- NoSQL databases : These databases are designed for large, unstructured data sets, such as social media data.
- Cloud databases : These databases are hosted on cloud computing platforms, providing scalability and flexibility.

2. What is Normalization?

database normalization entails organizing a database into several tables in order to reduce redundancy. You can design the database to follow any of the types of normalization such as 1NF, 2NF, and 3NF.

The main purpose of database normalization is to avoid complexities, eliminate duplicates, and organize data in a consistent way. In normalization, the data is divided into several tables linked together with relationships.

Database administrators are able to achieve these relationships by using primary keys, foreign keys.

3. What is Difference between DBMS and RDBMS ?

DBMS	**RDBMS**
Data stored is in the file format.	Data stored is in table format.
Individual access of data elements.	Multiple data elements are accessible together.
No connection between data	Data in the form of a table are linked together
No support for distributed database	Support distributed database
Data stored is a small quantity	Data is stored in a large amount
DBMS supports a single user	RDBMS supports multiple users
The software and hardware requirements are low	The software and hardware requirements are higher
Example: XML, Microsoft Access.	Example: Oracle, SQL Server.

4. What is MF Cod Rule of RDBMS Systems?

This is not a standard term in relational database management systems (RDBMS). The question likely refers to the MF Cod Rules, which are related to data normalization. These rules are commonly used to design relational databases to ensure data integrity and reduce redundancy.

Here's a breakdown of the most common "MF Cod Rules" and how they apply to RDBMS:

- 1NF (First Normal Form): Each column (attribute) in a table must contain atomic values (indivisible values). This means no repeating groups or multi-valued attributes within a single column.
- 2NF (Second Normal Form): A table must be in 1NF and each non-key attribute must be fully dependent on the entire primary key. This means eliminating partial dependencies.
- 3NF (Third Normal Form): A table must be in 2NF and every non-key attribute must be dependent only on the primary key, not on other non-key attributes. This means eliminating transitive dependencies.

5. What do you understand By Data Redundancy?

Data redundancy is a situation where the same data is stored in multiple places within a database or system. This can lead to inconsistencies, inefficiencies, and wasted storage space. It is generally considered a bad practice in database design.

6. What is DDL Interpreter?

DDL (Data Definition Language) is a type of SQL command used to define data structures and modify data. It creates, alters, and deletes database objects such as tables, views, indexes, and users. Examples of DDL statements include CREATE, ALTER, DROP and TRUNCATE.

7. What is DML Compiler in SQL?

DML is an abbreviation for Data Manipulation Language. Represents a collection of programming languages explicitly used to make changes to the database, such as: CRUD operations to create, read, update and delete data. Using INSERT, SELECT, UPDATE, and DELETE commands.

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

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

- NOT NULL - Ensures that a column cannot have a NULL value
- UNIQUE - Ensures that all values in a column are different
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- FOREIGN KEY - Prevents actions that would destroy links between tables

- CHECK - Ensures that the values in a column satisfies a specific condition
- DEFAULT - Sets a default value for a column if no value is specified
- CREATE INDEX - Used to create and retrieve data from the database very quickly

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

A SAVEPOINT is a point in a transaction in which you can roll the transaction back to a certain point without rolling back the entire transaction.

Syntax for Savepoint command : SAVEPOINT SAVEPOINT_NAME;

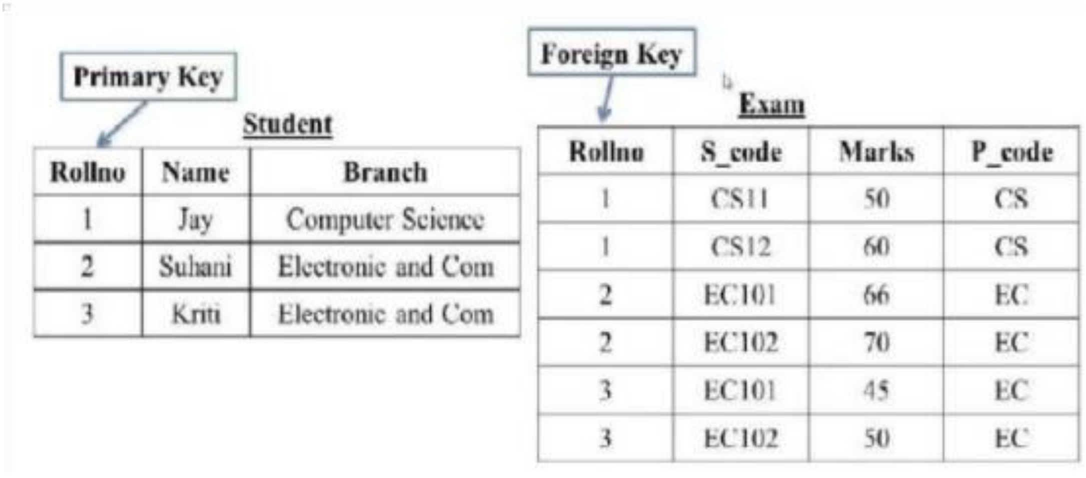
10.What is trigger and how to create a Trigger in SQL?

A trigger is a special kind of stored procedure that automatically executes when a specific event occurs on a table. The event can be an INSERT, UPDATE, or DELETE operation. Triggers are used to enforce business rules, maintain data integrity, and perform auditing.

How to Create a Trigger in SQL :

1. Use the CREATE TRIGGER statement: This statement is used to create a new trigger.
2. Specify the trigger name: The name of the trigger must be unique within the database.
3. Define the trigger type: There are three types of triggers:
    - FOR INSERT: The trigger executes after an INSERT operation.
    - FOR UPDATE: The trigger executes after an UPDATE operation.
    - FOR DELETE: The trigger executes after a DELETE operation.
4. Specify the table and the event: The table on which the trigger should be executed and the event that should trigger it.
5. Write the trigger code: This is the SQL code that will be executed when the trigger is fired. This code can perform any valid SQL operation, such as inserting data into another table, updating existing data, or calling other procedures.

1 Create Table Name : Student and Exam


Primary Key
Foreign Key

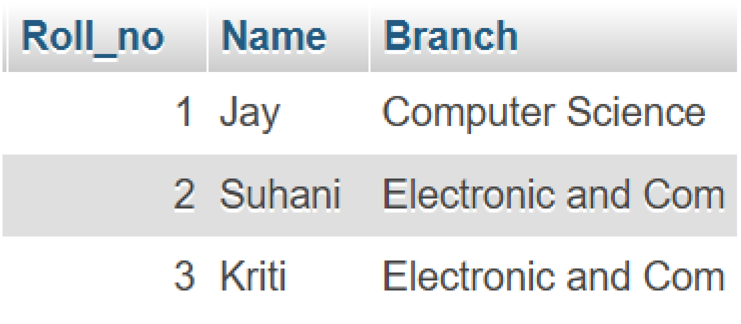
Student

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

Exam

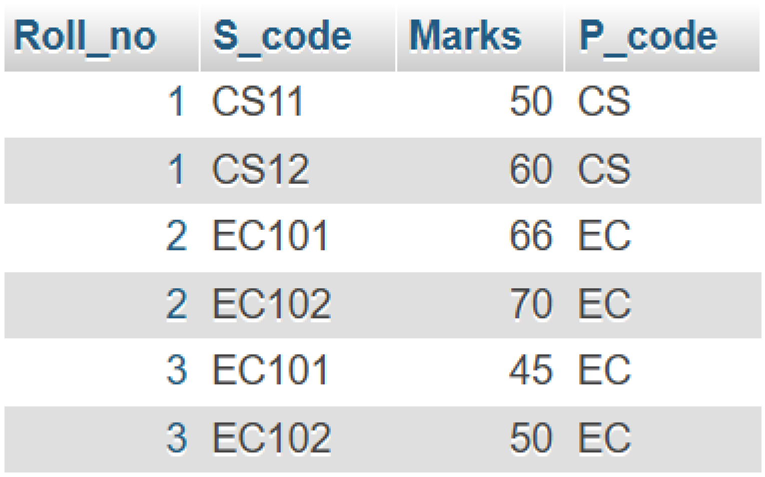
Rollno	S_code	Marks	P_code
1	CS11	50	CS
1	CS12	60	CS
2	EC101	66	EC
2	EC102	70	EC
3	EC101	45	EC
3	EC102	50	EC

Ans.
```
create table Student
(
Roll_no int PRIMARY KEY,
Name varchar(30),
Branch varchar(30)
);

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');
```



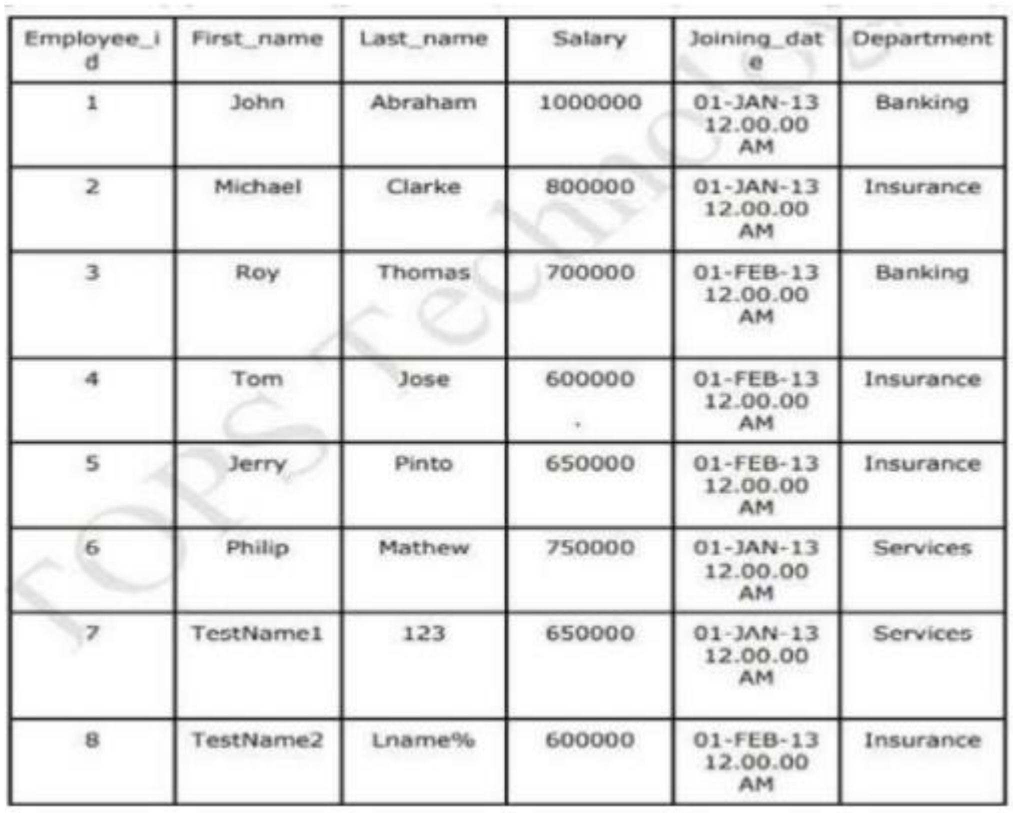
Roll_no	Name	Branch
1	Jay	Computer Science
2	Suhani	Electronic and Com
3	Kriti	Electronic and Com

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

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');
```

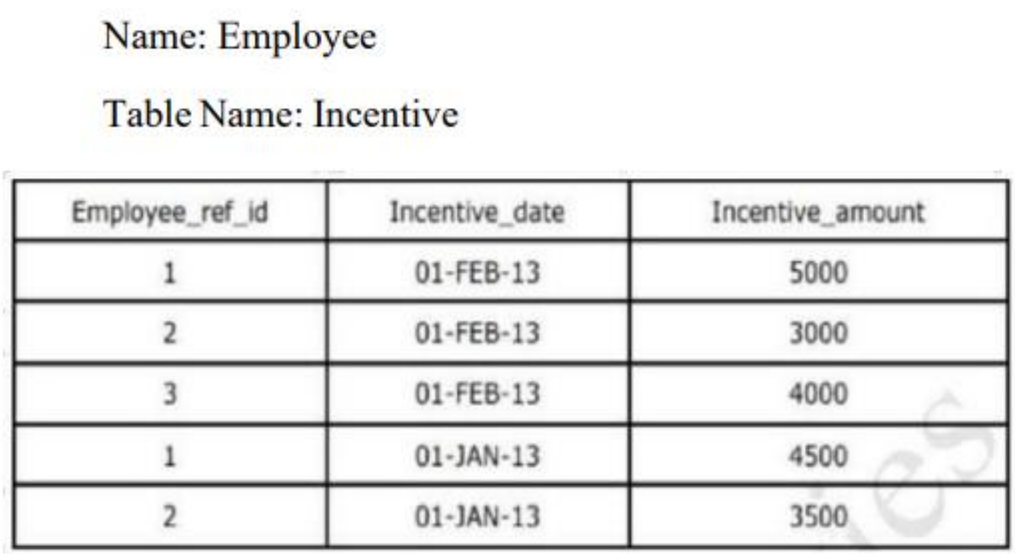


Roll_no	S_code	Marks	P_code
1	CS11	50	CS
1	CS12	60	CS
2	EC101	66	EC
2	EC102	70	EC
3	EC101	45	EC
3	EC102	50	EC

2 Create table given below: Employee and IncentiveTable



Employee_id	First_name	Last_name	Salary	Joining_date	Department
1	John	Abraham	1000000	01-JAN-13 12.00.00 AM	Banking
2	Michael	Clarke	800000	01-JAN-13 12.00.00 AM	Insurance
3	Roy	Thomas	700000	01-FEB-13 12.00.00 AM	Banking
4	Tom	Jose	600000	01-FEB-13 12.00.00 AM	Insurance
5	Jerry	Pinto	650000	01-FEB-13 12.00.00 AM	Insurance
6	Philip	Mathew	750000	01-JAN-13 12.00.00 AM	Services
7	TestName1	123	650000	01-JAN-13 12.00.00 AM	Services
8	TestName2	Lname%	600000	01-FEB-13 12.00.00 AM	Insurance



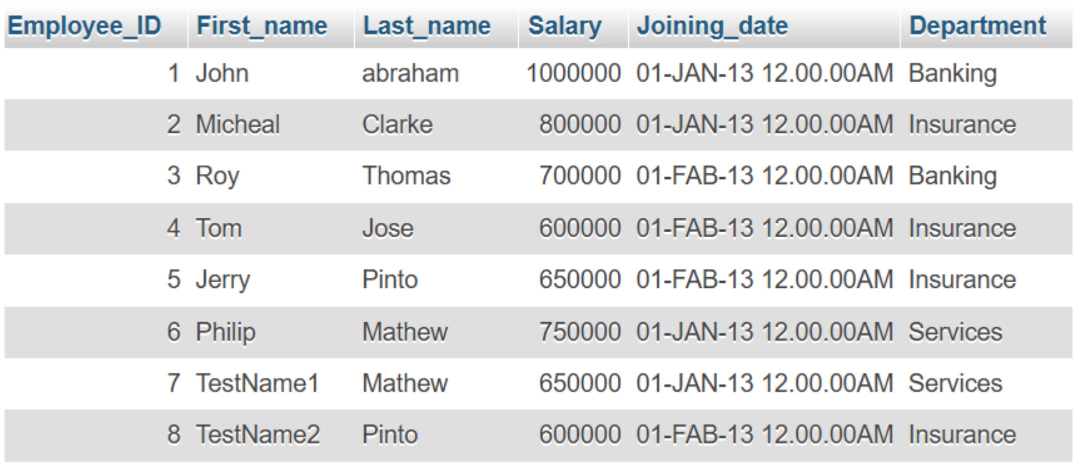
Name: Employee

Table Name: Incentive

Employee_ref_id	Incentive_date	Incentive_amount
1	01-FEB-13	5000
2	01-FEB-13	3000
3	01-FEB-13	4000
1	01-JAN-13	4500
2	01-JAN-13	3500

Ans.

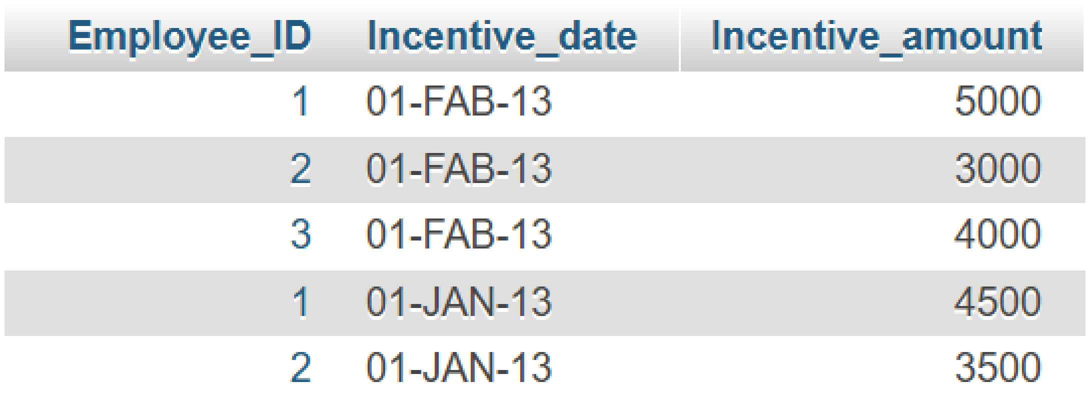
```
CREATE TABLE  Employee
(
Employee_ID int NOT NULL PRIMARY KEY,
First_name varchar(25),
Last_name varchar(25),
Salary int,
Joining_date varchar(25),
Department varchar(25)
);

insert into employee values(1, 'John', 'abraham',1000000,'01-JAN-13
12.00.00AM','Banking');
insert into employee values(2, 'Micheal', 'Clarke',800000,'01-JAN-13
12.00.00AM','Insurance');
insert into employee values(3, 'Roy', 'Thomas',700000,'01-FAB-13
12.00.00AM','Banking');
insert into employee values(4, 'Tom', 'Jose',600000,'01-FAB-13
12.00.00AM','Insurance');
insert into employee values(5, 'Jerry', 'Pinto',650000,'01-FAB-13
12.00.00AM','Insurance');
insert into employee values(6, 'Philip', 'Mathew',750000,'01-JAN-13
12.00.00AM','Services');
insert into employee values(7, 'TestName1', 'Mathew',650000,'01-JAN-13
12.00.00AM','Services');
insert into employee values(8, 'TestName2', 'Pinto',600000,'01-FAB-13
12.00.00AM','Insurance');
```



Employee_ID	First_name	Last_name	Salary	Joining_date	Department
1	John	abraham	1000000	01-JAN-13 12.00.00AM	Banking
2	Micheal	Clarke	800000	01-JAN-13 12.00.00AM	Insurance
3	Roy	Thomas	700000	01-FAB-13 12.00.00AM	Banking
4	Tom	Jose	600000	01-FAB-13 12.00.00AM	Insurance
5	Jerry	Pinto	650000	01-FAB-13 12.00.00AM	Insurance
6	Philip	Mathew	750000	01-JAN-13 12.00.00AM	Services
7	TestName1	Mathew	650000	01-JAN-13 12.00.00AM	Services
8	TestName2	Pinto	600000	01-FAB-13 12.00.00AM	Insurance

```
CREATE TABLE Incentive (
Employee_ID int,
Incentive_date varchar(25),
Incentive_amount int ,
FOREIGN KEY (Employee_ID) REFERENCES employee(Employee_ID)
);
```

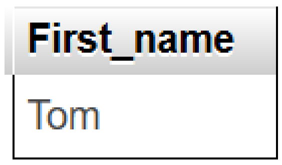
```
insert into incentive values(1,'01-FAB-13',5000);
insert into incentive values(2,'01-FAB-13',3000);
insert into incentive values(3,'01-FAB-13',4000);
insert into incentive values(1,'01-JAN-13',4500);
insert into incentive values(2,'01-JAN-13',3500);
```



Employee_ID	Incentive_date	Incentive_amount
1	01-FAB-13	5000
2	01-FAB-13	3000
3	01-FAB-13	4000
1	01-JAN-13	4500
2	01-JAN-13	3500

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

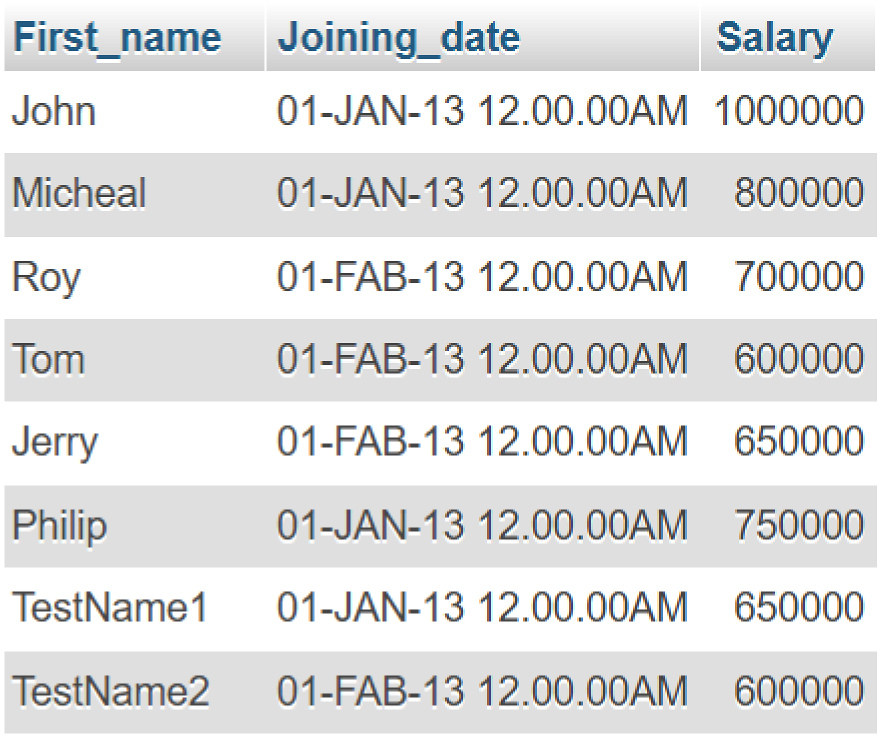
Ans. SELECT First_name FROM employee WHERE First_name='Tom';



First_name
Tom

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

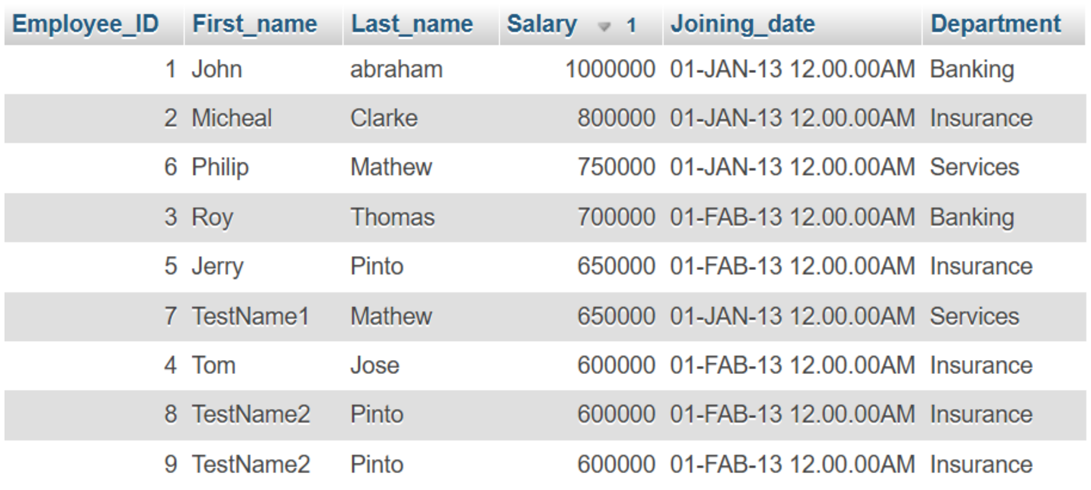
Ans. SELECT First_name ,Joining_date,Salary FROM employee;



First_name	Joining_date	Salary
John	01-JAN-13 12.00.00AM	1000000
Micheal	01-JAN-13 12.00.00AM	800000
Roy	01-FAB-13 12.00.00AM	700000
Tom	01-FAB-13 12.00.00AM	600000
Jerry	01-FAB-13 12.00.00AM	650000
Philip	01-JAN-13 12.00.00AM	750000
TestName1	01-JAN-13 12.00.00AM	650000
TestName2	01-FAB-13 12.00.00AM	600000

5 Get all employee details from the employee table order by First_Name.

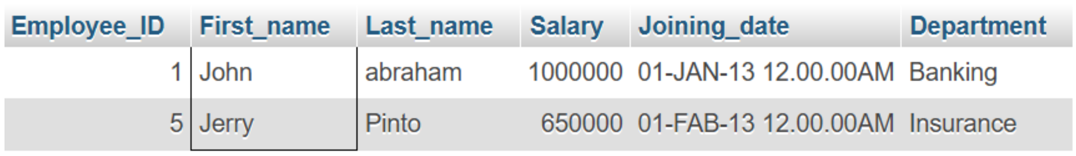
Ans SELECT * FROM employee ORDER BY First_name ,Salary DESC ;



Employee_ID	First_name	Last_name	Salary ▼ 1	Joining_date	Department
1	John	abraham	1000000	01-JAN-13 12.00.00AM	Banking
2	Micheal	Clarke	800000	01-JAN-13 12.00.00AM	Insurance
6	Philip	Mathew	750000	01-JAN-13 12.00.00AM	Services
3	Roy	Thomas	700000	01-FAB-13 12.00.00AM	Banking
5	Jerry	Pinto	650000	01-FAB-13 12.00.00AM	Insurance
7	TestName1	Mathew	650000	01-JAN-13 12.00.00AM	Services
4	Tom	Jose	600000	01-FAB-13 12.00.00AM	Insurance
8	TestName2	Pinto	600000	01-FAB-13 12.00.00AM	Insurance
9	TestName2	Pinto	600000	01-FAB-13 12.00.00AM	Insurance

6 Get employee details from the employee table whose first name contains ‘J’.

Ans. SELECT * FROM employee WHERE First_name LIKE'J%';

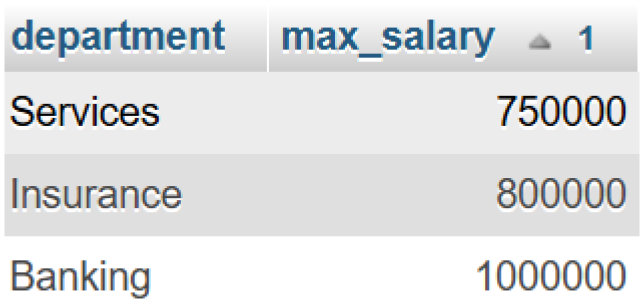


Employee_ID	First_name	Last_name	Salary	Joining_date	Department
1	John	abraham	1000000	01-JAN-13 12.00.00AM	Banking
5	Jerry	Pinto	650000	01-FAB-13 12.00.00AM	Insurance

7 Get the department-wise maximum salary from the employee table order by ascending the salary.

Ans

```
SELECT Department, MAX(Salary) as max_Salary
FROM employee
GROUP BY Department
ORDER BY max_Salary ASC;
```

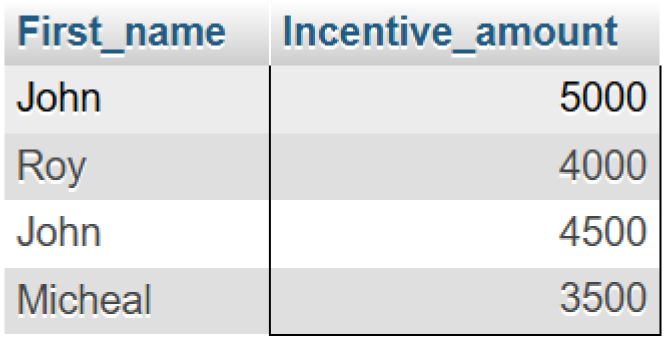


department	max_salary ▲ 1
Services	750000
Insurance	800000
Banking	1000000

9 Select first_name, incentive amount from employee and incentives table for those employees who have incentives and incentive amounts greater than 3000

Ans.

```
SELECT employee.First_name, incentive.Incentive_amount
FROM employee
JOIN incentive ON employee.Employee_ID = incentive.Employee_ID
WHERE incentive.Incentive_amount > 3000;
```

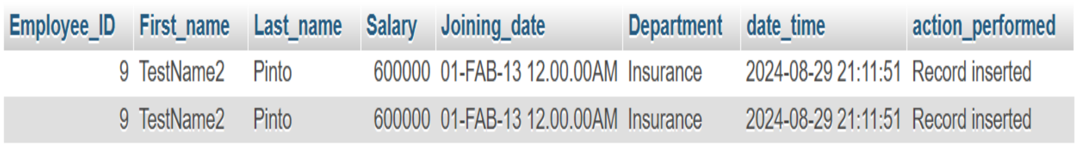


First_name	Incentive_amount
John	5000
Roy	4000
John	4500
Micheal	3500

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

Ans.

```
create table viewtable (
        Employee_ID int,
        First_name varchar(25),
    Last_name varchar(25),
    Salary int,
    Joining_date varchar(25),
    Department varchar(25),
        date_time timestamp,
        action_performed text
);
CREATE TRIGGER trg_employee
AFTER INSERT ON employee
FOR EACH ROW
BEGIN
    INSERT INTO test
(Employee_ID,First_name,Last_name,Salary,Joining_date,Department,action_perf
ormed)
    VALUES (new.Employee_ID,new.First_name,new.Last_name,
new.Salary,new.Joining_date,new.Department,'Record inserted');
END;


insert into employee values(8, 'TestName2', 'Pinto',600000,'01-FAB-13
12.00.00AM','Insurance');
```

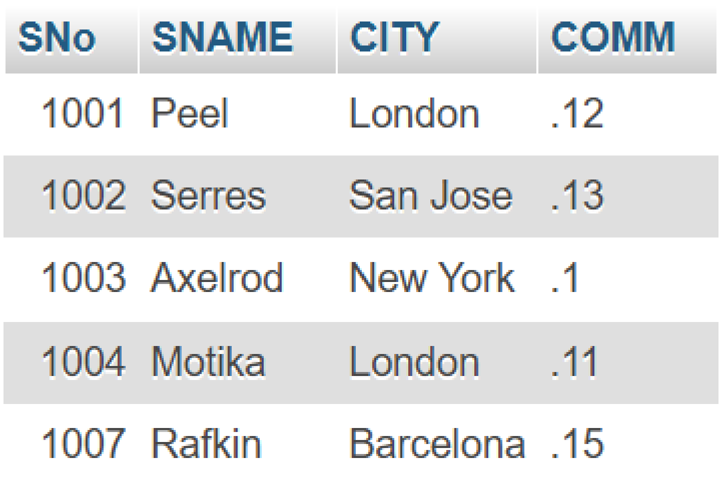


Employee_ID	First_name	Last_name	Salary	Joining_date	Department	date_time	action_performed
9	TestName2	Pinto	600000	01-FAB-13 12.00.00AM	Insurance	2024-08-29 21:11:51	Record inserted
9	TestName2	Pinto	600000	01-FAB-13 12.00.00AM	Insurance	2024-08-29 21:11:51	Record inserted

11 Create table given below: Salesperson and Customer.

Ans.

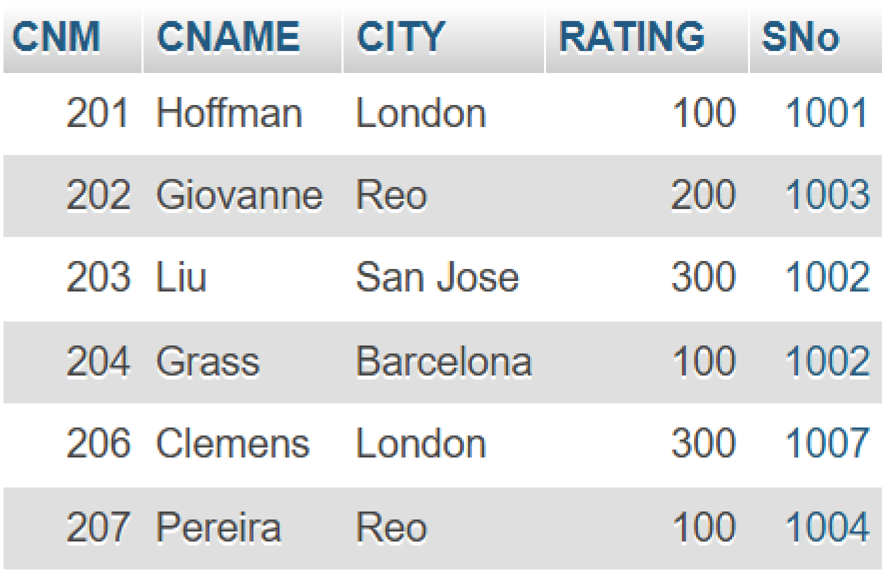
```
create table Salesperson
(
SNo int PRIMARY KEY,
SNAME varchar(25),
CITY varchar(25),
COMM varchar(25)
);

insert into Salesperson values(1001, 'Peel','London','.12');
insert into Salesperson values(1002, 'Serres','San Jose','.13');
insert into Salesperson values(1004, 'Motika','London','.11');
insert into Salesperson values(1007, 'Rafkin','Barcelona','.15');
insert into Salesperson values(1003, 'Axelrod','New York','.1');
```



SNo	SNAME	CITY	COMM
1001	Peel	London	.12
1002	Serres	San Jose	.13
1003	Axelrod	New York	.1
1004	Motika	London	.11
1007	Rafkin	Barcelona	.15

```
CREATE TABLE Customer (
CNM int PRIMARY KEY,
CNAME varchar(25),
CITY varchar(25),
RATING int,
SNo int,
FOREIGN KEY (SNo) REFERENCES salesperson(SNo)
);
```

```
insert into customer values(201, 'Hoffman','London',100,1001);
insert into customer values(202, 'Giovanne','Reo',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','Reo',100,1004);
```



CNM	CNAME	CITY	RATING	SNo
201	Hoffman	London	100	1001
202	Giovanne	Reo	200	1003
203	Liu	San Jose	300	1002
204	Grass	Barcelona	100	1002
206	Clemens	London	300	1007
207	Pereira	Reo	100	1004

13 All orders for more than $1000.

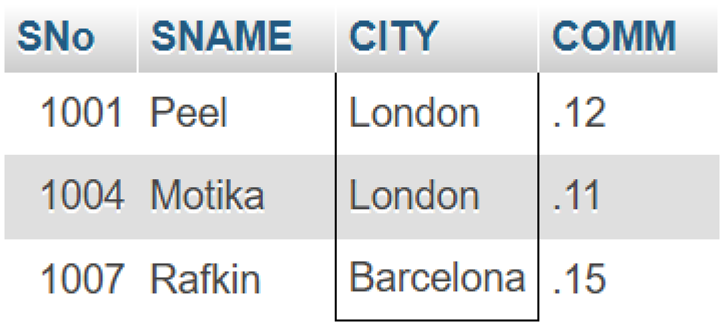
Ans. SELECT * FROM orders WHERE amount > 1000;

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

Ans. SELECT * FROM salesperson WHERE CITY ='London'AND COMM>.12;

15 All salespeople either in Barcelona or in London

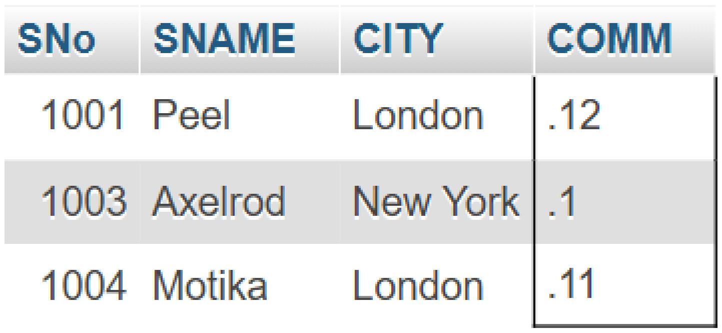
Ans. SELECT * FROM salesperson WHERE CITY='Barcelona' OR CITY='London';



SNo	SNAME	CITY	COMM
1001	Peel	London	.12
1004	Motika	London	.11
1007	Rafkin	Barcelona	.15

16 All salespeople with a commission between 0.10 and 0.12.

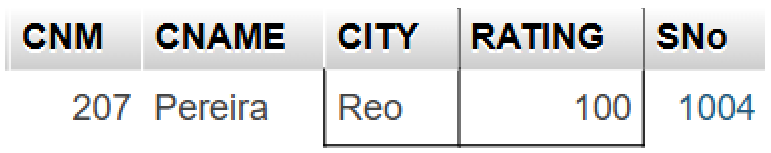
Ans. SELECT * FROM salesperson WHERE COMM BETWEEN .10 AND .12;



SNo	SNAME	CITY	COMM
1001	Peel	London	.12
1003	Axelrod	New York	.1
1004	Motika	London	.11

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

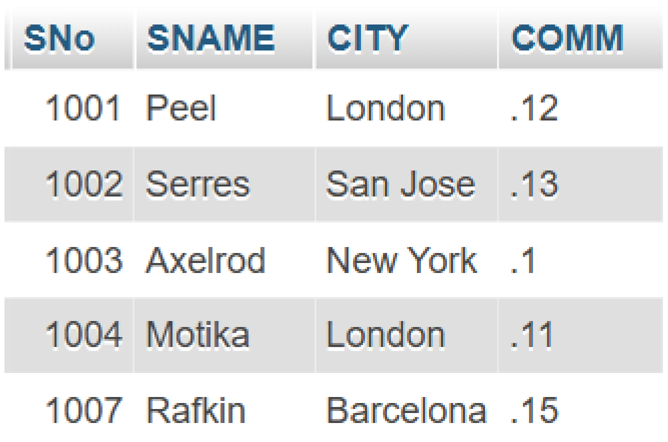
Ans. SELECT * FROM customer WHERE RATING <=100 AND CITY='Reo';



CNM	CNAME	CITY	RATING	SNo
207	Pereira	Reo	100	1004

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

Ans SELECT * FROM salesperson ;



SNo	SNAME	CITY	COMM
1001	Peel	London	.12
1002	Serres	San Jose	.13
1003	Axelrod	New York	.1
1004	Motika	London	.11
1007	Rafkin	Barcelona	.15

19

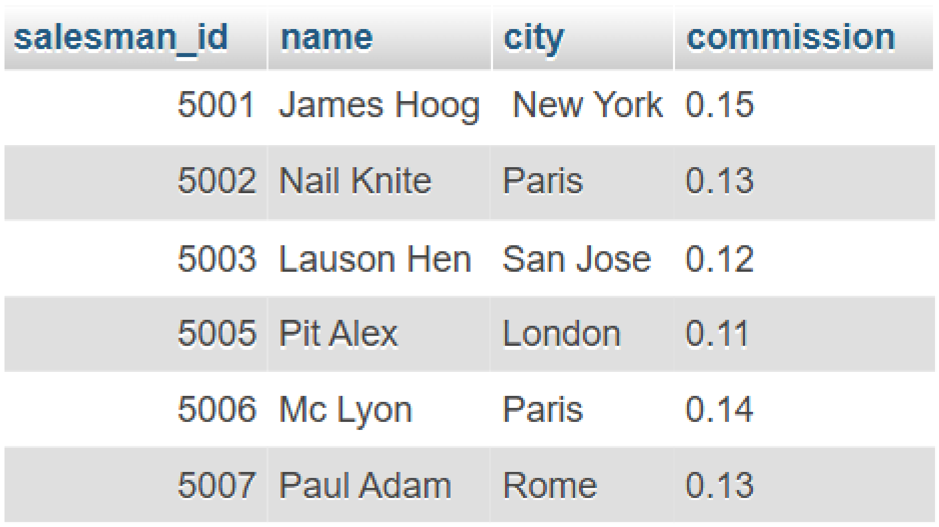
```
salesman_id |    name     |   city    | commission
------------+-------------+-----------+------------
5001 | James Hoog | New York |       0.15
5002 | Nail Knite | Paris    |       0.13
5005 | Pit Alex   | London   |       0.11
5006 | Mc Lyon    | Paris    |       0.14
5007 | Paul Adam  | Rome 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

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

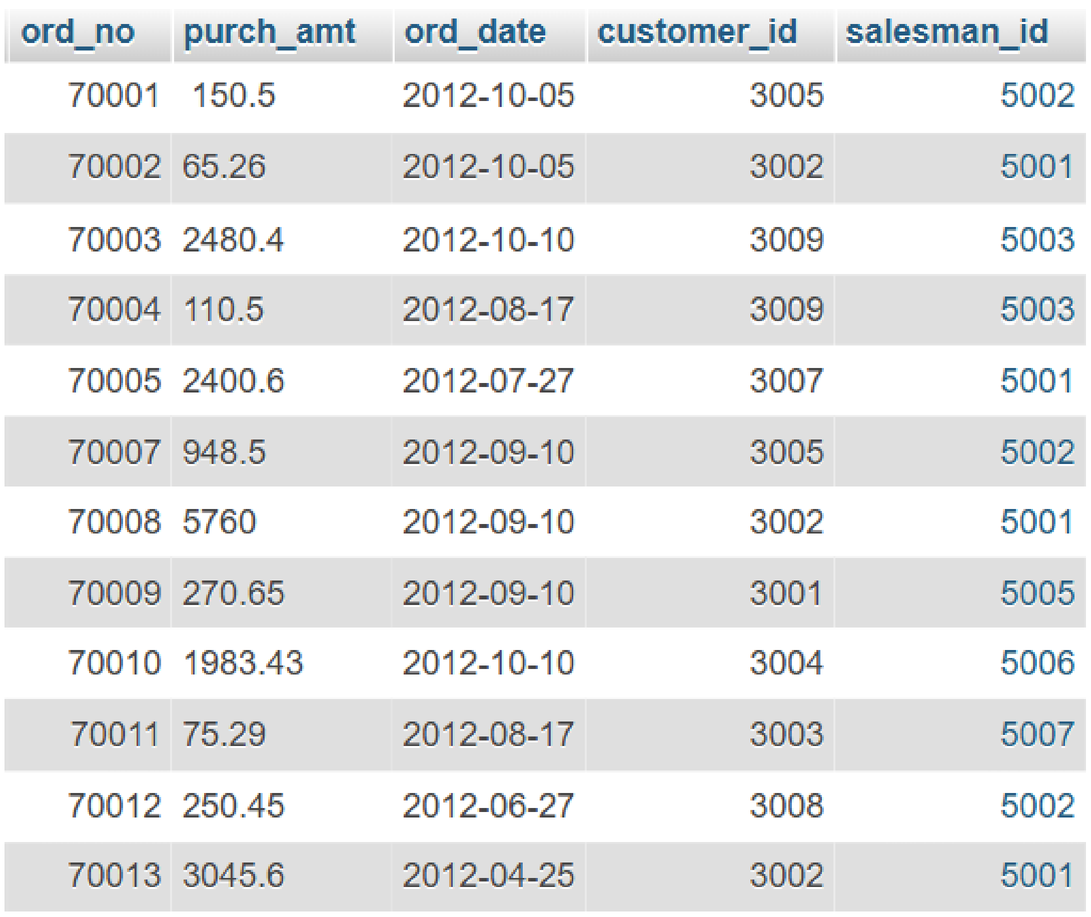
Ans

```
CREATE TABLE Salesman
(
salesman_id int PRIMARY KEY,
name varchar(25),
city varchar(25),
commission varchar(25)
);

insert into salesman values(5001, 'James Hoog',' New York','0.15');
insert into salesman values(5002, 'Nail Knite','Paris','0.13');
insert into salesman values(5005, 'Pit Alex','London','0.11');
insert into salesman values(5006, 'Mc Lyon ','Paris','0.14');
insert into salesman values(5007, 'Paul Adam','Rome','0.13');
insert into salesman 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
5003	Lauson Hen	San Jose	0.12
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13

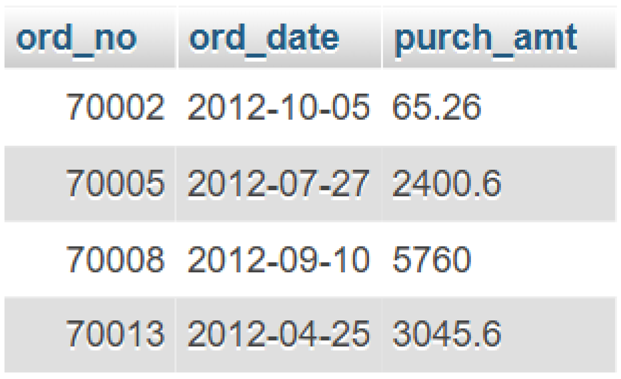
```
CREATE TABLE orders (
ord_no int PRIMARY KEY,
purch_amt varchar(25),
ord_date  date,
customer_id int,
salesman_id int,
FOREIGN KEY (salesman_id) REFERENCES salesman(salesman_id)
);

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

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.

Ans. SELECT ord_no, ord_date, purch_amt FROM orders WHERE salesman_id = 5001;

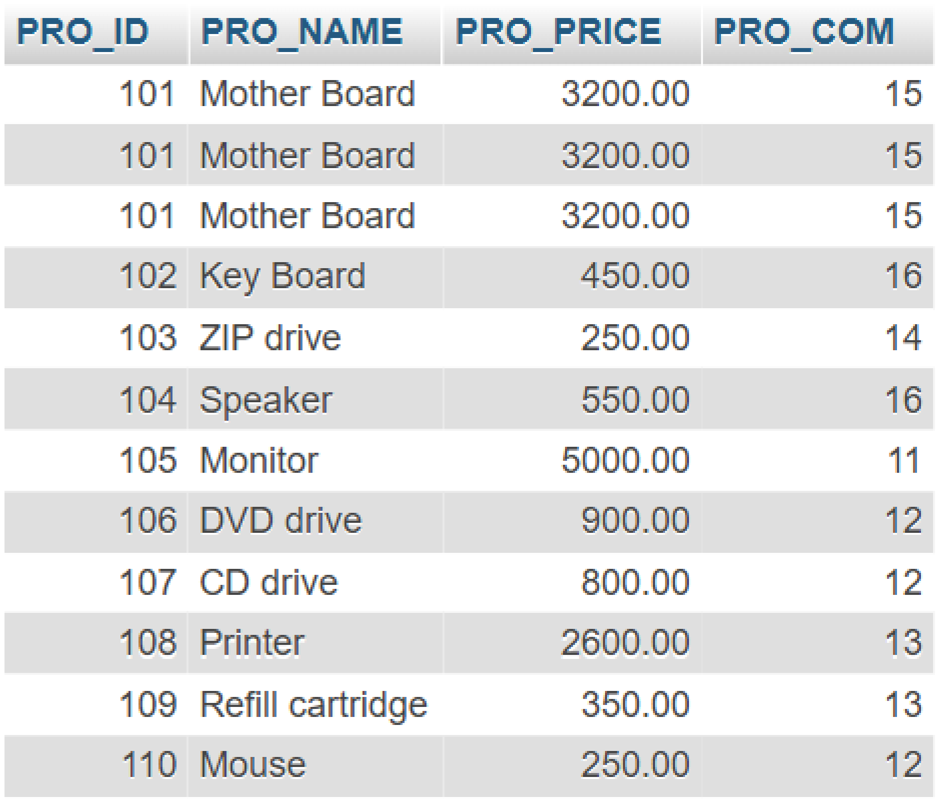


ord_no	ord_date	purch_amt
70002	2012-10-05	65.26
70005	2012-07-27	2400.6
70008	2012-09-10	5760
70013	2012-04-25	3045.6

20

```
CREATE TABLE item_mast (
  PRO_ID int,
  PRO_NAME varchar(25),
  PRO_PRICE DECIMAL(10,2),
  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 cartridge', 350.00, 13);
INSERT INTO item_mast VALUES (110, 'Mouse', 250.00, 12);
```

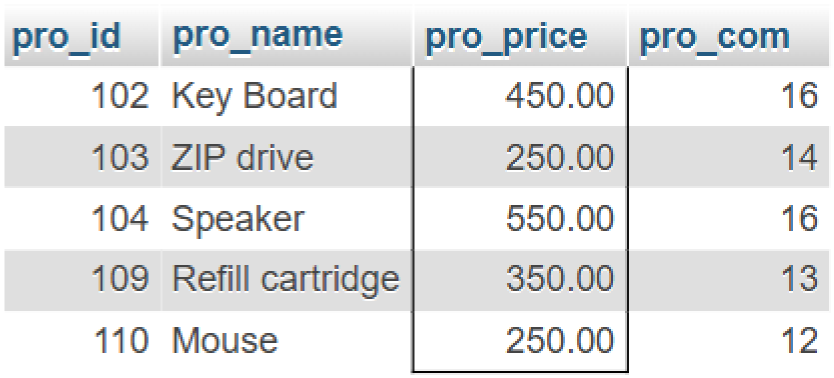


PRO_ID	PRO_NAME	PRO_PRICE	PRO_COM
101	Mother Board	3200.00	15
101	Mother Board	3200.00	15
101	Mother Board	3200.00	15
102	Key Board	450.00	16
103	ZIP drive	250.00	14
104	Speaker	550.00	16
105	Monitor	5000.00	11
106	DVD drive	900.00	12
107	CD drive	800.00	12
108	Printer	2600.00	13
109	Refill cartridge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

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.

Ans.

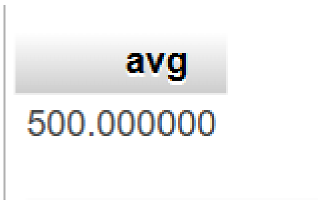
```
SELECT pro_id, pro_name, pro_price, pro_com FROM item_mast WHERE
pro_price BETWEEN 200 AND 600;
```



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 cartridge	350.00	13
110	Mouse	250.00	12

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

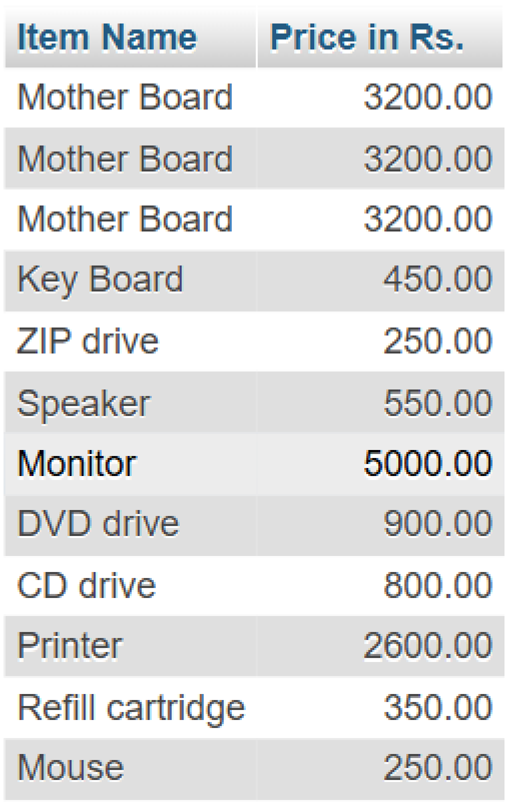
Ans. SELECT AVG(PRO_PRICE)AS avg FROM item_mast WHERE PRO_COM = 16;



avg
500.000000

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

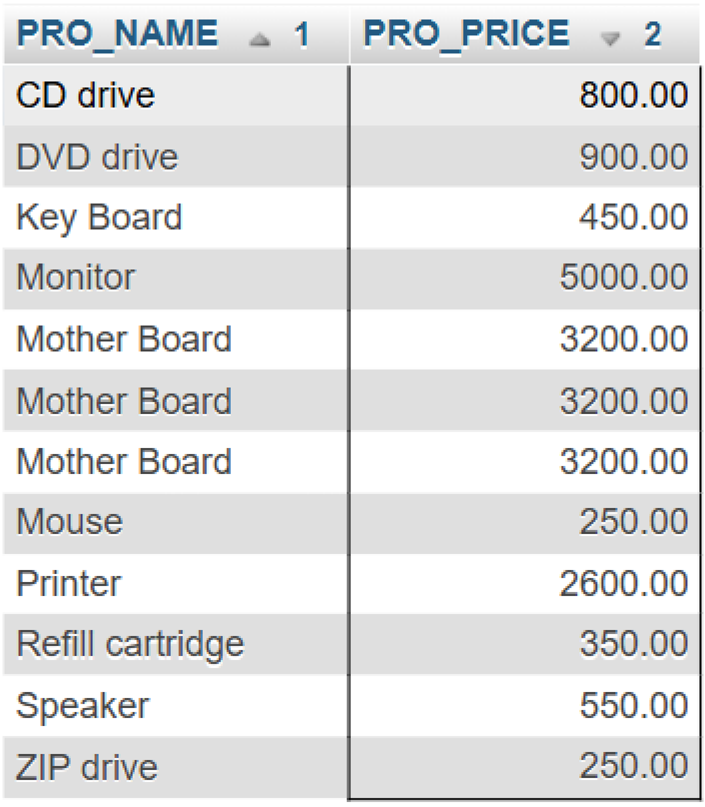
Ans. SELECT PRO_NAME AS 'Item Name', PRO_PRICE AS 'Price in Rs.' FROM item_mast;



Item Name	Price in Rs.
Mother Board	3200.00
Mother Board	3200.00
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 cartridge	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.

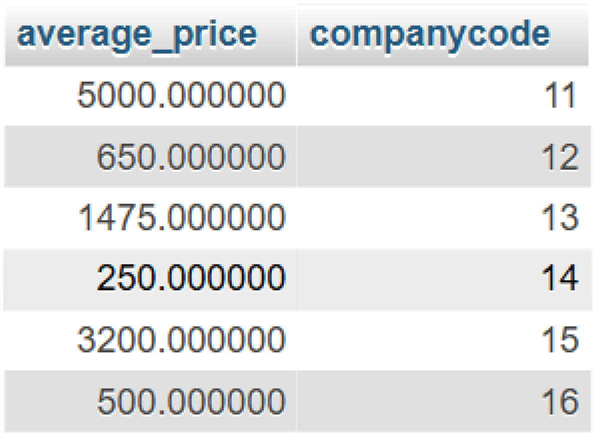
Ans. SELECT PRO_NAME, PRO_PRICE FROM item_mast WHERE PRO_PRICE >= 250 ORDER BY PRO_NAME ASC, PRO_PRICE DESC;



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

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

SELECT AVG(PRO_PRICE) AS average_price, PRO_COM AS companycode FROM item_mast GROUP BY PRO_COM;



average_price	companycode
5000.000000	11
650.000000	12
1475.000000	13
250.000000	14
3200.000000	15
500.000000	16