MODULE 5) Database

1. What do you understand By Database

Ans: Database is a place to store data for long term. It's not related to only storing but fetching, updating, deleting data, too.

2. What is Normalization?

Ans: Database normalization is the process of structuring a relational database in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity.

3. What is Difference between DBMS and RDBMS?

Ans:

DBMS: It's a type of software that is typically used to manage the data flow such as insertion, updating, deleting and retrieving, so that it maintains the uniformity.

RDBMS: RDBMS stands for Relational Database Management System, as it's name suggest it is used to create or maintain relation using key constraints.

4. What is MF Cod Rule of RDBMS Systems?

Ans: Codd's rule in DBMS also known as Codd's 12 rules/commandments is 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?

Ans: Data redundancy is a process of keeping data in more than one places in the database of the organization.

6. What is DDL Interpreter?

Ans: DDL Interpreter: It interprets the DDL (Data Definition Language) Instructions and stores the record in a data dictionary (in a table containing meta-data) Query Optimizer.

7. What is DML Compiler in SQL?

Ans: A DML (data manipulation language) refers to a computer programming language that allows you to add (insert), delete (delete), and alter (update) data in a database.

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

Ans: Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement.

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

Ans: 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.

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

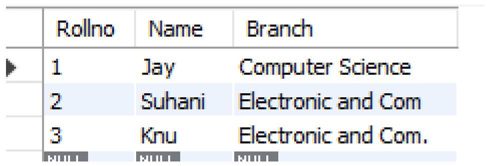
Ans: An SQL trigger allows you to specify SQL actions that should be executed automatically when a specific event occurs in the database

1. Create table named Student and Exam

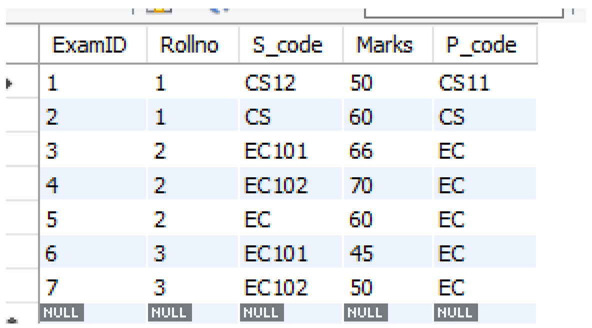
Code:

```
CREATE TABLE Student (
   Rollno INT AUTO_INCREMENT PRIMARY KEY,
   Name VARCHAR(50),
   Branch VARCHAR(50)
);

INSERT INTO Student (Rollno, Name, Branch)
VALUES
   (1, 'Jay', 'Computer Science'),
   (2, 'Suhani', 'Electronic and Com'),
   (3, 'Knu', 'Electronic and Com.');



-- Create the Exam table
CREATE TABLE Exam (
   ExamID INT AUTO_INCREMENT PRIMARY KEY,
   Rollno INT,
   S_code VARCHAR(50),
   Marks INT,
   P_code VARCHAR(50),
   FOREIGN KEY (Rollno) REFERENCES Student(Rollno)
);

-- Insert data into the Exam table
INSERT INTO Exam (Rollno, S_code, Marks, P_code)
VALUES
   (1, 'CS12', 50, 'CS11'),
   (1, 'CS', 60, 'CS'),
   (2, 'EC101', 66, 'EC'),
   (2, 'EC102', 70, 'EC'),
   (2, 'EC', 60, 'EC'),
   (3, 'EC101', 45, 'EC'),
   (3, 'EC102', 50, 'EC');
```

Output:



Rollno	Name	Branch
1	Jay	Computer Science
2	Suhani	Electronic and Com
3	Knu	Electronic and Com.
NULL	NULL	NULL



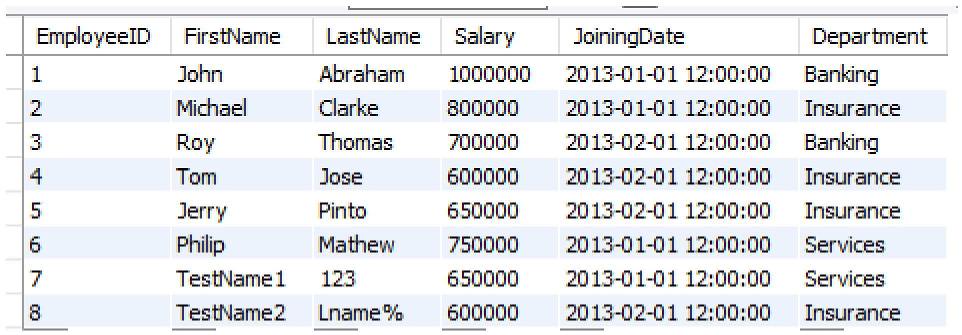
ExamID	Rollno	S_code	Marks	P_code
1	1	CS12	50	CS11
2	1	CS	60	CS
3	2	EC101	66	EC
4	2	EC102	70	EC
5	2	EC	60	EC
6	3	EC101	45	EC
7	3	EC102	50	EC
NULL	NULL	NULL	NULL	NULL

2. Create a table named Employee and Incentives.

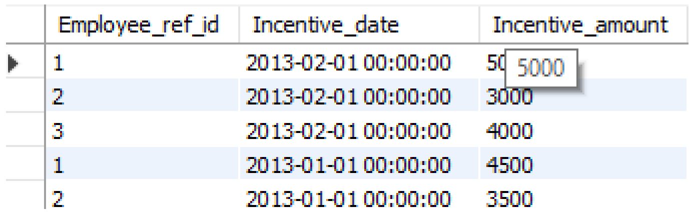
Code:

```
-- Create the Example table
CREATE TABLE Employee (
   EmployeeID INT PRIMARY KEY,
   FirstName VARCHAR(50),
   LastName VARCHAR(50),
   Salary INT,
   JoiningDate DATETIME,
   Department VARCHAR(50)
);

-- Insert data into the Example table
INSERT INTO Employee (EmployeeID, FirstName, LastName, Salary, JoiningDate,
Department)
VALUES
   (1, 'John', 'Abraham', 1000000, '2013-01-01 12:00:00', 'Banking'),
   (2, 'Michael', 'Clarke', 800000, '2013-01-01 12:00:00', 'Insurance'),
   (3, 'Roy', 'Thomas', 700000, '2013-02-01 12:00:00', 'Banking'),
   (4, 'Tom', 'Jose', 600000, '2013-02-01 12:00:00', 'Insurance'),
   (5, 'Jerry', 'Pinto', 650000, '2013-02-01 12:00:00', 'Insurance'),
   (6, 'Philip', 'Mathew', 750000, '2013-01-01 12:00:00', 'Services'),
   (7, 'TestName1', '123', 650000, '2013-01-01 12:00:00', 'Services'),
   (8, 'TestName2', 'Lname%', 600000, '2013-02-01 12:00:00', 'Insurance');

CREATE TABLE Incentive (
   Employee_ref_id INT,
   Incentive_date DATETIME,
   Incentive_amount INT,
   FOREIGN KEY (Employee_ref_id) REFERENCES Employee(EmployeeID)
);

-- Insert data into the Incentive table
INSERT INTO Incentive (Employee_ref_id, Incentive_date, Incentive_amount)
VALUES
   (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);
```

Output:



EmployeeID	FirstName	LastName	Salary	JoiningDate	Department
1	John	Abraham	1000000	2013-01-01 12:00:00	Banking
2	Michael	Clarke	800000	2013-01-01 12:00:00	Insurance
3	Roy	Thomas	700000	2013-02-01 12:00:00	Banking
4	Tom	Jose	600000	2013-02-01 12:00:00	Insurance
5	Jerry	Pinto	650000	2013-02-01 12:00:00	Insurance
6	Philip	Mathew	750000	2013-01-01 12:00:00	Services
7	TestName1	123	650000	2013-01-01 12:00:00	Services
8	TestName2	Lname%	600000	2013-02-01 12:00:00	Insurance



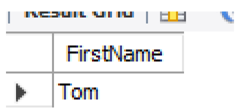
Employee_ref_id	Incentive_date	Incentive_amount
1	2013-02-01 00:00:00	5000
2	2013-02-01 00:00:00	3000
3	2013-02-01 00:00:00	4000
1	2013-01-01 00:00:00	4500
2	2013-01-01 00:00:00	3500

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

Code:

```
SELECT FirstName
FROM Employee
WHERE FirstName = 'Tom' OR LastName = 'Tom';
```

Output:



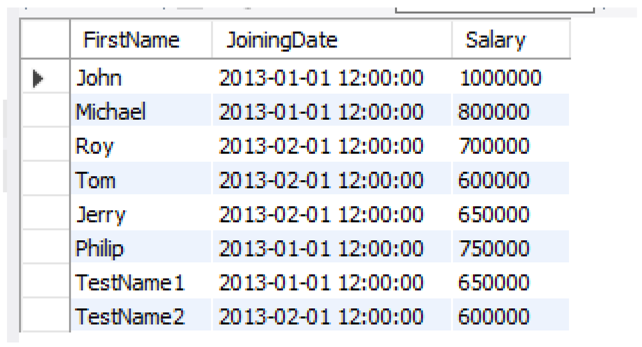
FirstName
Tom

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

Code:

```
SELECT FirstName, JoiningDate, Salary
FROM Employee;
```

Output:



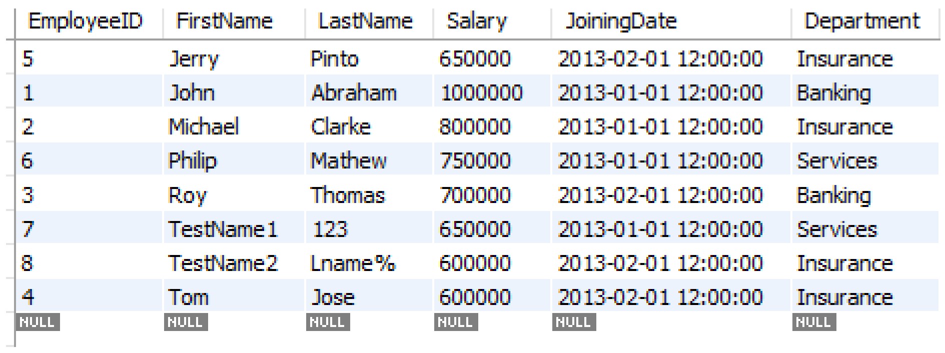
FirstName	JoiningDate	Salary
John	2013-01-01 12:00:00	1000000
Michael	2013-01-01 12:00:00	800000
Roy	2013-02-01 12:00:00	700000
Tom	2013-02-01 12:00:00	600000
Jerry	2013-02-01 12:00:00	650000
Philip	2013-01-01 12:00:00	750000
TestName1	2013-01-01 12:00:00	650000
TestName2	2013-02-01 12:00:00	600000

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

Code:

```
SELECT *
FROM Employee
ORDER BY FirstName ASC, Salary DESC;
```

Output:



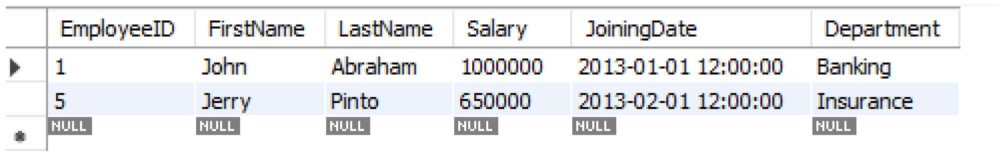
EmployeeID	FirstName	LastName	Salary	JoiningDate	Department
5	Jerry	Pinto	650000	2013-02-01 12:00:00	Insurance
1	John	Abraham	1000000	2013-01-01 12:00:00	Banking
2	Michael	Clarke	800000	2013-01-01 12:00:00	Insurance
6	Philip	Mathew	750000	2013-01-01 12:00:00	Services
3	Roy	Thomas	700000	2013-02-01 12:00:00	Banking
7	TestName1	123	650000	2013-01-01 12:00:00	Services
8	TestName2	Lname%	600000	2013-02-01 12:00:00	Insurance
4	Tom	Jose	600000	2013-02-01 12:00:00	Insurance
NULL	NULL	NULL	NULL	NULL	NULL

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

Code:

```
SELECT *
FROM Employee
WHERE FirstName LIKE '%J%';
```

Output:



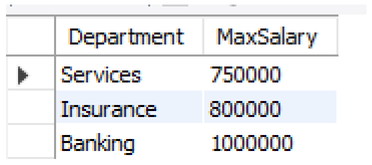
EmployeeID	FirstName	LastName	Salary	JoiningDate	Department
1	John	Abraham	1000000	2013-01-01 12:00:00	Banking
5	Jerry	Pinto	650000	2013-02-01 12:00:00	Insurance
NULL	NULL	NULL	NULL	NULL	NULL

7. Get department wise maximum salary from employee table order by salaryascending?

Code:

```
SELECT Department, MAX(Salary) AS MaxSalary
FROM Employee
GROUP BY Department
ORDER BY MaxSalary ASC;
```

Output:



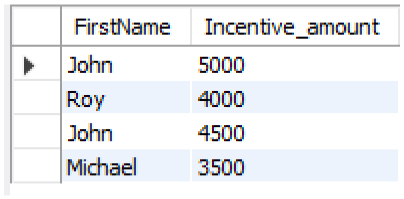
Department	MaxSalary
Services	750000
Insurance	800000
Banking	1000000

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

Code:

```
SELECT e.FirstName, i.Incentive_amount
FROM Employee e
JOIN Incentive i ON e.EmployeeID = i.Employee_ref_id
WHERE i.Incentive_amount > 3000;
```

Output:



FirstName	Incentive_amount
John	5000
Roy	4000
John	4500
Michael	3500

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

Code:

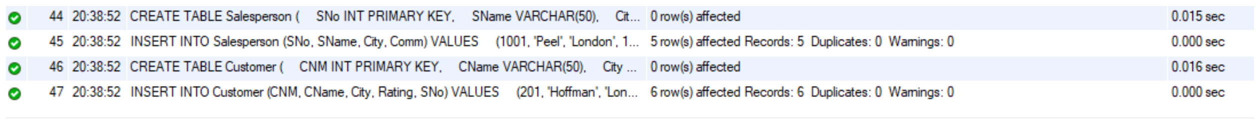
```
DELIMITER //

CREATE TRIGGER after_employee_insert
AFTER INSERT
ON Employee
FOR EACH ROW
BEGIN
   -- Inserting records into viewtable
   INSERT INTO viewtable (EmployeeID, FirstName, LastName, Salary, JoiningDate,
Department)
   VALUES (NEW.EmployeeID, NEW.FirstName, NEW.LastName, NEW.Salary,
NEW.JoiningDate, NEW.Department);
END;
//

DELIMITER ;
```

10. Create table given below: Salesperson and Customer

Code:

```
-- Create Salesperson table
CREATE TABLE Salesperson (
    SNo INT PRIMARY KEY,
    SName VARCHAR(50),
    City VARCHAR(50),
    Comm INT
);

-- Insert data into Salesperson table
INSERT INTO Salesperson (SNo, SName, City, Comm)
VALUES
    (1001, 'Peel', 'London', 12),
    (1002, 'Serres', 'San Jose', 13),
    (1004, 'Motika', 'London', 11),
    (1007, 'Rafkin', 'Barcelona', 15),
    (1003, 'Axelrod', 'New York', 1);

-- Create Customer table
CREATE TABLE Customer (
    CNM INT PRIMARY KEY,
    CName VARCHAR(50),
    City VARCHAR(50),
    Rating INT,
    SNo INT,
    FOREIGN KEY (SNo) REFERENCES Salesperson(SNo)
);

-- Insert data into Customer table
INSERT INTO Customer (CNM, CName, City, Rating, SNo)
VALUES
    (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);
```

Output:



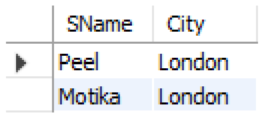
```
44  20:38:52  CREATE TABLE Salesperson (    SNo INT PRIMARY KEY,    SName VARCHAR(50),    Cit...  0 row(s) affected                                        0.015 sec
45  20:38:52  INSERT INTO Salesperson (SNo, SName, City, Comm) VALUES    (1001, 'Peel', 'London', 1...  5 row(s) affected Records: 5  Duplicates: 0  Warnings: 0   0.000 sec
46  20:38:52  CREATE TABLE Customer (    CNM INT PRIMARY KEY,    CName VARCHAR(50),    City ...  0 row(s) affected                                        0.016 sec
47  20:38:52  INSERT INTO Customer (CNM, CName, City, Rating, SNo) VALUES    (201, 'Hoffman', 'Lon...  6 row(s) affected Records: 6  Duplicates: 0  Warnings: 0   0.000 sec
```

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

Code:

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

Output:



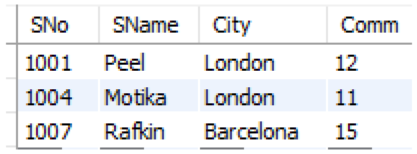
SName	City
Peel	London
Motika	London

12. .All salespeople either in Barcelona or in London

Code:

```
SELECT *
FROM Salesperson
WHERE City IN ('Barcelona', 'London');
```

Output:

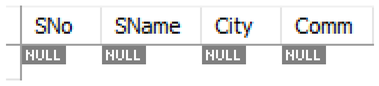


SNo	SName	City	Comm
1001	Peel	London	12
1004	Motika	London	11
1007	Rafkin	Barcelona	15

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

Code:
SELECT *
FROM Salesperson
WHERE Comm > 0.10 AND Comm < 0.12;

Output:



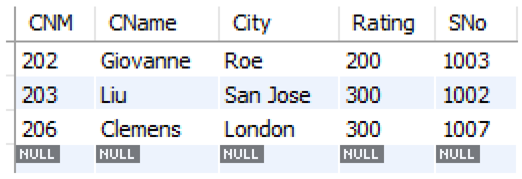
SNo	SName	City	Comm
NULL	NULL	NULL	NULL

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

Code:

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

Output:



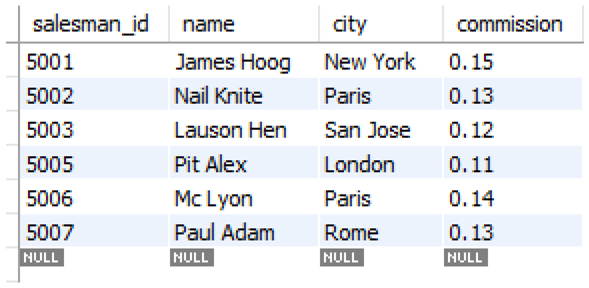
CNM	CName	City	Rating	SNo
202	Giovanne	Roe	200	1003
203	Liu	San Jose	300	1002
206	Clemens	London	300	1007
NULL	NULL	NULL	NULL	NULL

15..Write a SQL statement that displays all the information about all salespeople

Code:

```
-- Create the salespeople table
CREATE TABLE salespeople (
    salesman_id INT PRIMARY KEY,
    name VARCHAR(50),
    city VARCHAR(50),
    commission DECIMAL(5, 2)
);

-- Insert records into the salespeople table
INSERT INTO salespeople (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);
select * from salespeople;
```

Output:



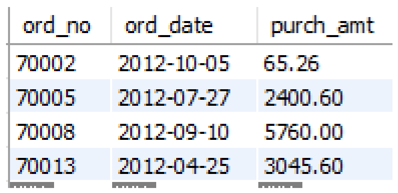
salesman_id	name	city	commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5003	Lauson Hen	San Jose	0.12
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13
NULL	NULL	NULL	NULL

16. 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.

Code:

```
-- Create the orders table
CREATE TABLE orders (
    ord_no INT PRIMARY KEY,
    purch_amt DECIMAL(10, 2),
    ord_date DATE,
    customer_id INT,
    salesman_id INT
);

-- Insert records into the orders table
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);

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

Output:



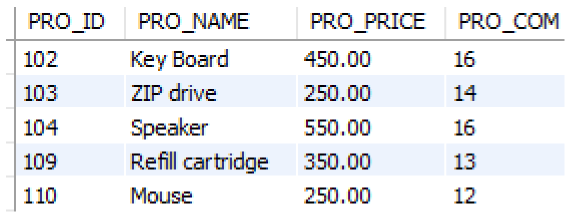
ord_no	ord_date	purch_amt
70002	2012-10-05	65.26
70005	2012-07-27	2400.60
70008	2012-09-10	5760.00
70013	2012-04-25	3045.60

17. 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.

Code:

```
-- Create the item_mast table
CREATE TABLE item_mast (
    PRO_ID INT PRIMARY KEY,
    PRO_NAME VARCHAR(50),
    PRO_PRICE DECIMAL(10, 2),
    PRO_COM INT
);

-- Insert records into the item_mast table
INSERT INTO item_mast (PRO_ID, PRO_NAME, PRO_PRICE, PRO_COM)
VALUES
    (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);

-- Select products within the price range of Rs.200 to Rs.600
SELECT PRO_ID, PRO_NAME, PRO_PRICE, PRO_COM
FROM item_mast
WHERE PRO_PRICE BETWEEN 200.00 AND 600.00;
```

Output:



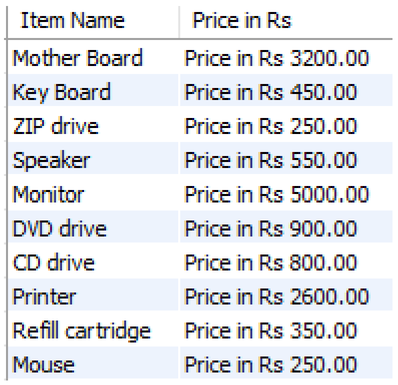
PRO_ID	PRO_NAME	PRO_PRICE	PRO_COM
102	Key Board	450.00	16
103	ZIP drive	250.00	14
104	Speaker	550.00	16
109	Refill cartridge	350.00	13
110	Mouse	250.00	12

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

Code:

```
SELECT pro_name AS 'Item Name', CONCAT('Price in Rs ', pro_price) AS 'Price in Rs'
FROM item_mast;
```

Output:



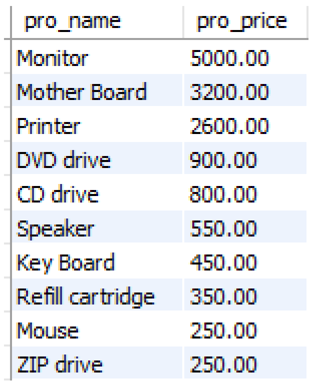
Item Name	Price in Rs
Mother Board	Price in Rs 3200.00
Key Board	Price in Rs 450.00
ZIP drive	Price in Rs 250.00
Speaker	Price in Rs 550.00
Monitor	Price in Rs 5000.00
DVD drive	Price in Rs 900.00
CD drive	Price in Rs 800.00
Printer	Price in Rs 2600.00
Refill cartridge	Price in Rs 350.00
Mouse	Price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

19. SELECT pro_name, pro_price
FROM item_mast
WHERE pro_price >= 250
ORDER BY pro_price DESC, pro_name ASC;

Code:

```
SELECT pro_name, pro_price
FROM item_mast
WHERE pro_price >= 250
ORDER BY pro_price DESC, pro_name ASC;
```

Output:



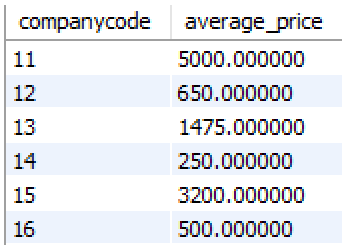
pro_name	pro_price
Monitor	5000.00
Mother Board	3200.00
Printer	2600.00
DVD drive	900.00
CD drive	800.00
Speaker	550.00
Key Board	450.00
Refill cartridge	350.00
Mouse	250.00
ZIP drive	250.00

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

Code:

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

Output:



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