MODULE – 5

(DBMS Assignment)

● Topics Covered Basics of Database

1. What do you understand By Database

Answer: A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS).

2. What is Normalization?

Answer: Normalization is the process of organizing data in a database. It includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.

3. What is Difference between DBMS and RDBMS?

Answer: The main differences are: RDBMS stores data in the form of tables, whereas DBMS stores data in the form of files. Single users are supported by DBMS, whereas multiple users are supported by RDBMS. Client-server architecture is not supported by DBMS, although it is supported by RDBMS.

4.What is MF Cod Rule of RDBMS Systems?

Answer:

Codd's rules are proposed by a computer scientist named Dr. Edgar F. Codd and he also invent the relational model for database management. These rules are made to ensure data integrity, consistency, and usability. This set of rules basically signifies the characteristics and requirements of a relational database management system (RDBMS). In this article, we will learn about various Codd's rules.

Codd's Rules in DBMS

Rule 1: The Information Rule

All information, whether it is user information or metadata, that is stored in a database must be entered as a value in a cell of a table. It is said that everything within the database is organized in a table layout.

Rule 2: The Guaranteed Access Rule

Each data element is guaranteed to be accessible logically with a combination of the table name, primary key (row value), and attribute name (column value).

Rule 3: Systematic Treatment of NULL Values

Every Null value in a database must be given a systematic and uniform treatment.

Rule 4: Active Online Catalog Rule

The database catalog, which contains metadata about the database, must be stored and accessed using the same relational database management system.

Rule 5: The Comprehensive Data Sublanguage Rule

A crucial component of any efficient database system is its ability to offer an easily understandable data manipulation language (DML) that facilitates defining, querying, and modifying information within the database.

Rule 6: The View Updating Rule

All views that are theoretically updatable must also be updatable by the system.

Rule 7: High-level Insert, Update, and Delete

A successful database system must possess the feature of facilitating high-level insertions, updates, and deletions that can grant users the ability to conduct these operations with ease through a single query.

Rule 8: Physical Data Independence

Application programs and activities should remain unaffected when changes are made to the physical storage structures or methods.

Rule 9: Logical Data Independence

Application programs and activities should remain unaffected when changes are made to the logical structure of the data, such as adding or modifying tables.

Rule 10: Integrity Independence

Integrity constraints should be specified separately from application programs and stored in the catalog. They should be automatically enforced by the database system.

Rule 11: Distribution Independence

The distribution of data across multiple locations should be invisible to users, and the database system should handle the distribution transparently.

Rule 12: Non-Subversion Rule

If the interface of the system is providing access to low-level records, then the interface must not be able to damage the system and bypass security and integrity constraints.

5. What do you understand ByData Redundancy?

Answer: Data redundancy occurs when the same piece of data exists in multiple places, whereas data inconsistency is when the same data exists in different formats in multiple tables.

6. What is DDL Interpreter?

Answer: DDL Interpreter interprets the DDL statements and records the generated statements in the table containing metadata.

7. What is DML Compiler in SQL?

Answer:

DML Commands in SQL

DML is an abbreviation of Data Manipulation Language.

The DML commands in Structured Query Language change the data present in the SQL database. We can easily access, store, modify, update and delete the existing records from the database using DML commands.

Following are the four main DML commands in SQL:

1. SELECT Command
2. INSERT Command
3. UPDATE Command
4. DELETE Command

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

Answer:

SQL Constraints

In a database table, we can add rules to a column known as constraints. These rules control the data that can be stored in a column.

- **NOT NULL**
- **UNIQUE**
- **PRIMARY KEY**
- **FOREIGN KEY**
- **CHECK**
- **DEFAULT**
- **CREATE INDEX**

NOT NULL Constraint

The NOT NULL constraint in a column means that the column cannot store NULL values. For example,

```
CREATE TABLE Colleges (
  college_id INT NOT NULL,
  college_code VARCHAR(20) NOT NULL,
  college_name VARCHAR(50)
);
```

Here, college_id the college_id and the college_code columns of the Colleges table won't allow NULL values.

UNIQUE Constraint

The UNIQUE constraint in a column means that the column must have unique value. For example,

```
CREATE TABLE Colleges (
  college_id INT NOT NULL UNIQUE,
  college_code VARCHAR(20) UNIQUE,
  college_name VARCHAR(50)
);
```

Here, the value of the college_code column must be unique. Similarly, the value of college_id must be unique as well as it cannot store NULL values.

PRIMARY KEY Constraint

The PRIMARY KEY constraint is simply a combination of NOT NULL and UNIQUE constraints. It means that the column value is used to uniquely identify the row. For example,

```
CREATE TABLE Colleges (
  college_id INT PRIMARY KEY,
  college_code VARCHAR(20) NOT NULL,
  college_name VARCHAR(50)
);
```

Here, the value of the college_id column is a unique identifier for a row. Similarly, it cannot store NULL value and must be UNIQUE.

FOREIGN KEY Constraint

The FOREIGN KEY (REFERENCES in some databases) constraint in a column is used to reference a record that exists in another table. For example,

```
CREATE TABLE Orders (
  order_id INT PRIMARY KEY,
  customer_id int REFERENCES Customers(id)
);
```

Here, the value of the college_code column references the row in another table named Customers.

It means that the value of customer_id in the Orders table must be a value from the id column of the Customers table.

CHECK Constraint

The CHECK constraint checks the condition before allowing values in a table. For example,

```
CREATE TABLE Orders (
  order_id INT PRIMARY KEY,
  amount int CHECK (amount >= 100)
);
```

Here, the value of the amount column must be **greater than or equal to 100**. If not, the SQL statement results in an error.

DEFAULT Constraint

The DEFAULT constraint is used to set the default value if we try to store NULL in a column. For example,

```
CREATE TABLE College (
  college_id INT PRIMARY KEY,
  college_code VARCHAR(20),
  college_country VARCHAR(20) DEFAULT 'US'
);
```

Here, the default value of the college_country column is **US**.

If we try to store the NULL value in the college_country column, its value will be **US**.

CREATE INDEX Constraint

If a column has CREATE INDEX constraint, it's faster to retrieve data if we use that column for data retrieval. For example,

```
-- create table
CREATE TABLE Colleges (
  college_id INT PRIMARY KEY,
  college_code VARCHAR(20) NOT NULL,
  college_name VARCHAR(50)
);

-- create index
CREATE INDEX college_index
ON Colleges(college_code);
```

Here, the SQL command creates an index named college_index on the Colleges table using college_id column.

Note: We cannot see the speed difference with less records in a table. However, we can easily notice the speed difference between using indexes and not using indexes.

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

Answer:

Savepoint in SQL

- Savepoint is a command in SQL that is used with the rollback command.
- It is a command in Transaction Control Language that is used to mark the transaction in a table.
- Consider you are making a very long table, and you want to roll back only to a certain position in a table then; this can be achieved using the savepoint.
- If you made a transaction in a table, you could mark the transaction as a certain name, and later on, if you want to roll back to that point, you can do it easily by using the transaction's name.
- Savepoint is helpful when we want to roll back only a small part of a table and not the whole table. In simple words, we can say savepoint is a bookmark in SQL.

Let us see the practical examples to understand this concept more clearly. We will use the MySQL database for writing all the queries**.**

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

Answer:

Triggers in SQL Server

A trigger is a set of SQL statements that reside in system memory with unique names. It is a specialized category of stored procedure that is called automatically when a database server event occurs. Each trigger is always associated with a table.

A **trigger is called a special procedure** because it cannot be called directly like a stored procedure. The key distinction between the trigger and procedure is that a trigger is called automatically when a data modification event occurs against a table. A stored procedure, on the other hand, must be invoked directly.

The following are the main characteristics that distinguish triggers from stored procedures:

- We cannot manually execute/invoked triggers.
- Triggers have no chance of receiving parameters.
- A transaction cannot be committed or rolled back inside a trigger.

SQL Queries

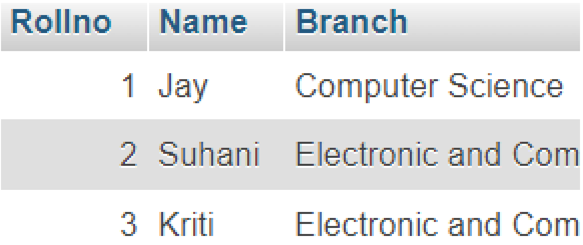
1. Create Table Name : Student and Exam

Create Table:

```
CREATE TABLE student
(Rollno int PRIMARY KEY, Name varchar(50),Branch varchar(50));
```

Insert Data:

```
INSERT INTO student values

(1,"Jay","ComputerScience"),

(2,"Suhani","Electronicand Com"),

(3,"Kriti","Electronic and Com");
```



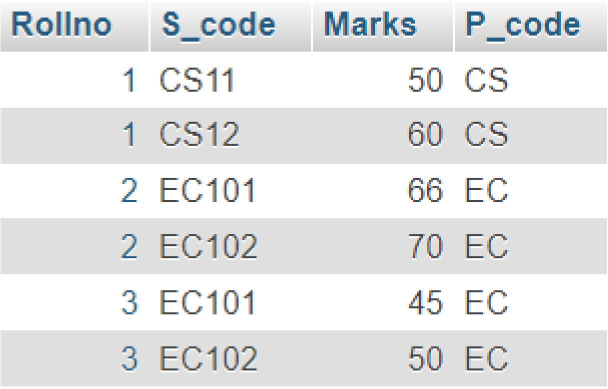
Rollno	Name	Branch
1	Jay	Computer Science
2	Suhani	Electronic and Com
3	Kriti	Electronic and Com

Create Table:

```
CREATE TABLE Exam

(Rollno int,S_code varchar(50), Marks int, P_code varchar(50),

 FOREIGN KEY (Rollno) REFERENCES student (Rollno));
```

Insert data:

```
insert into Exam values

 (1, "CS11", 50, "CS"),

 (1, "CS12", 60, "CS"),

 (2, "EC101", 66, "EC"),

(2, "EC102", 70, "EC"),

(3, "EC101", 45, "EC"),

(3, "EC102", 50, "EC");
```



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

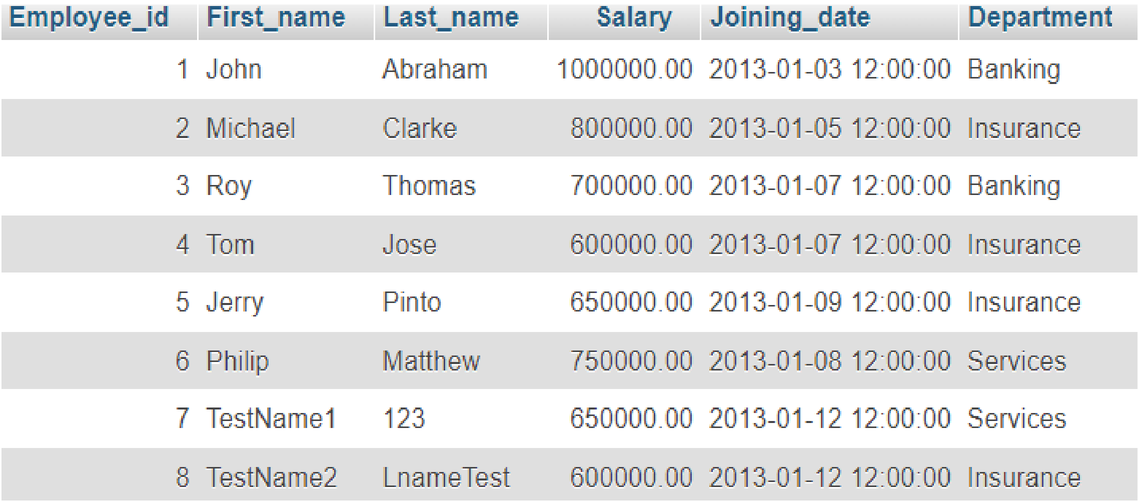
(Q-2)1.Create table given below: Employee and IncentiveTable

Create table:

```
create table Employee (Employee_id int, First_name

varchar(50), Last_name varchar(50), Salary int,

Joining_date DateTime, Department varchar(50));
```

Insert Data:

```
INSERT INTO Employee VALUES

(1, 'John', 'Abraham', 1000000, '2013-01-012:00:00 AM', 'Banking'),

(2, 'Michael', 'Clarke', 800000, '2013-01-05 12:00:00 AM', 'Insurance'),

(3, 'Roy', 'Thomas', 700000, '2013-01-07 12:00:00 AM', 'Banking'),

(4, 'Tom', 'Jose', 600000, '2013-01-07 12:00:00 AM', 'Insurance'),

(5, 'Jerry', 'Pinto', 650000, '2013-01-09 12:00:00 AM', 'Insurance'),

(6, 'Philip', 'Matthew', 750000, '2013-01-08 12:00:00 AM', 'Services'),

 (7, 'TestName1', '123', 650000, '2013-01-12 12:00:00 AM', 'Services'),

(8, 'TestName2', 'LnameTest', 600000, '2013-01-12 12:00:00 AM', 'Insurance');
```



Employee_id	First_name	Last_name	Salary	Joining_date	Department
1	John	Abraham	1000000.00	2013-01-03 12:00:00	Banking
2	Michael	Clarke	800000.00	2013-01-05 12:00:00	Insurance
3	Roy	Thomas	700000.00	2013-01-07 12:00:00	Banking
4	Tom	Jose	600000.00	2013-01-07 12:00:00	Insurance
5	Jerry	Pinto	650000.00	2013-01-09 12:00:00	Insurance
6	Philip	Matthew	750000.00	2013-01-08 12:00:00	Services
7	TestName1	123	650000.00	2013-01-12 12:00:00	Services
8	TestName2	LnameTest	600000.00	2013-01-12 12:00:00	Insurance

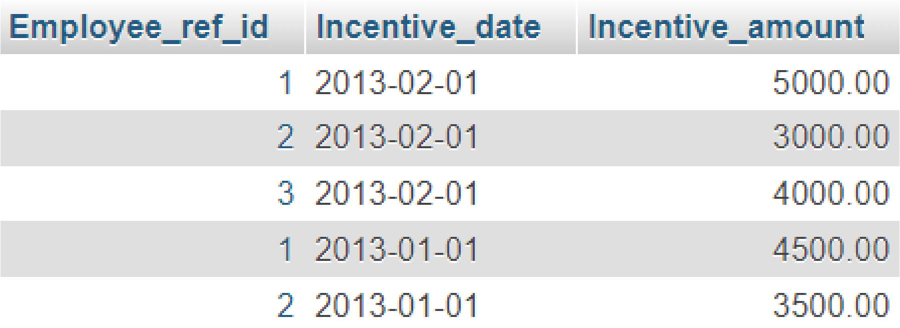
2.Table Name : Incentive

Create table:

```
CREATE TABLE Incentive

( Employee_ref_id INT, Incentive_date DATE, Incentive_amount

 DECIMAL(10, 2), FOREIGN KEY (Employee_ref_id) REFERENCES

 Employee(Employee_id) );
```

Insert Data:

```
INSERT INTO Incentive 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);
```

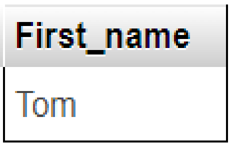


Employee_ref_id	Incentive_date	Incentive_amount
1	2013-02-01	5000.00
2	2013-02-01	3000.00
3	2013-02-01	4000.00
1	2013-01-01	4500.00
2	2013-01-01	3500.00

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

Select Query:

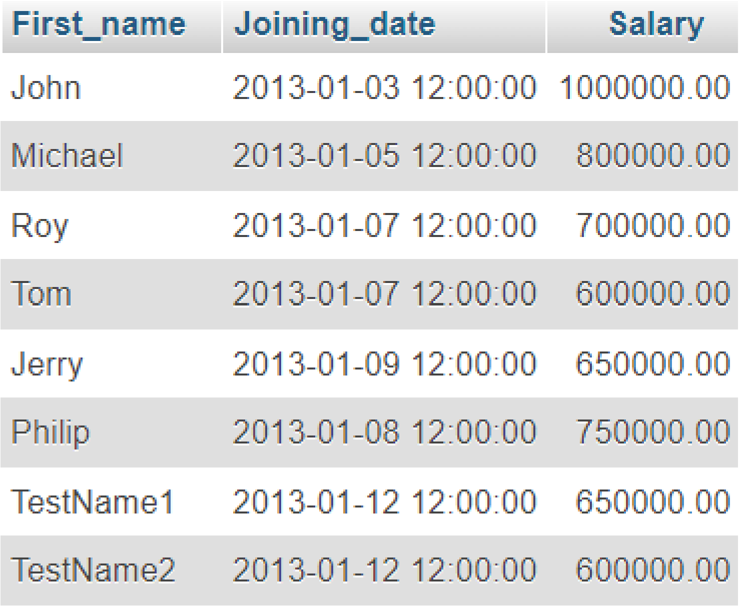
```
SELECT First_name from employee WHERE First_name = 'Tom';
```



First_name
Tom

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

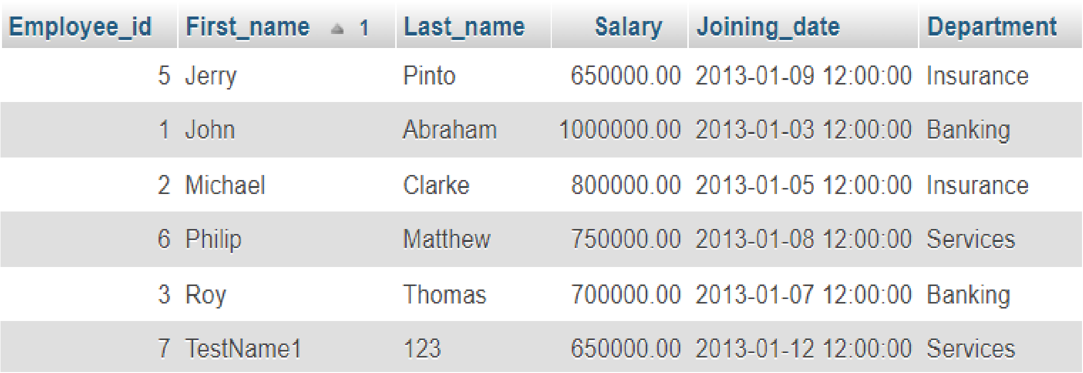
```
select First_name, Joining_date, Salary from employee;
```



First_name	Joining_date	Salary
John	2013-01-03 12:00:00	1000000.00
Michael	2013-01-05 12:00:00	800000.00
Roy	2013-01-07 12:00:00	700000.00
Tom	2013-01-07 12:00:00	600000.00
Jerry	2013-01-09 12:00:00	650000.00
Philip	2013-01-08 12:00:00	750000.00
TestName1	2013-01-12 12:00:00	650000.00
TestName2	2013-01-12 12:00:00	600000.00

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

```
SELECT * FROM Employee ORDER BY First_name;
```

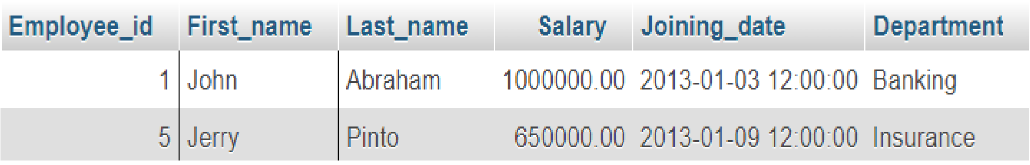


Employee_id	First_name ▲ 1	Last_name	Salary	Joining_date	Department
5	Jerry	Pinto	650000.00	2013-01-09 12:00:00	Insurance
1	John	Abraham	1000000.00	2013-01-03 12:00:00	Banking
2	Michael	Clarke	800000.00	2013-01-05 12:00:00	Insurance
6	Philip	Matthew	750000.00	2013-01-08 12:00:00	Services
3	Roy	Thomas	700000.00	2013-01-07 12:00:00	Banking
7	TestName1	123	650000.00	2013-01-12 12:00:00	Services

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

Select Query:

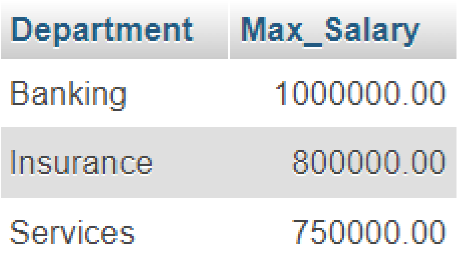
```
SELECT * FROM employee WHERE First_name LIKE 'J%';
```



Employee_id	First_name	Last_name	Salary	Joining_date	Department
1	John	Abraham	1000000.00	2013-01-03 12:00:00	Banking
5	Jerry	Pinto	650000.00	2013-01-09 12:00:00	Insurance

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

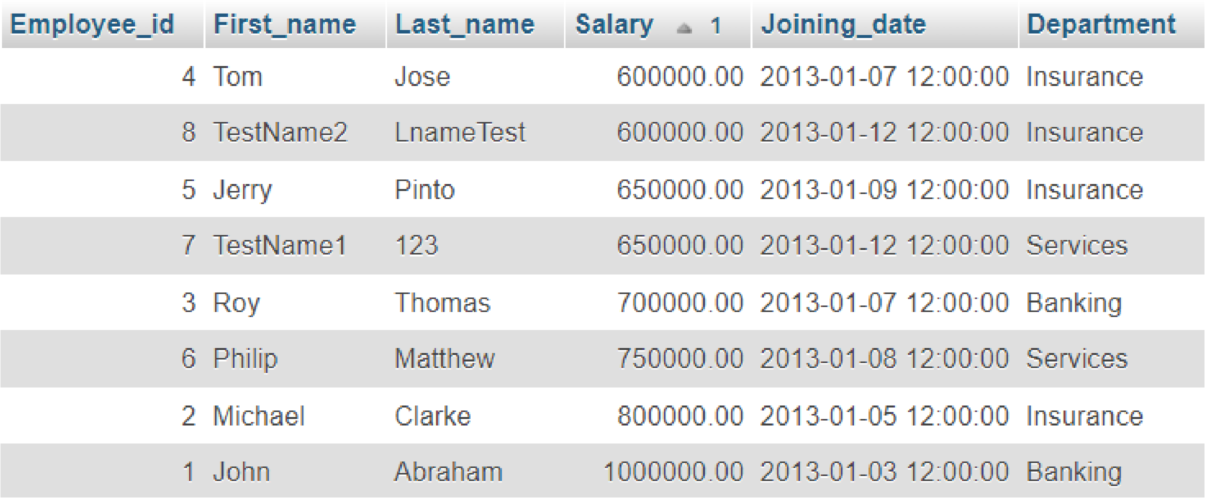
```
SELECT Department, MAX(Salary) AS Max_Salary FROM Employee GROUP BY Department;
```



Department	Max_Salary
Banking	1000000.00
Insurance	800000.00
Services	750000.00

8. salaryascending?

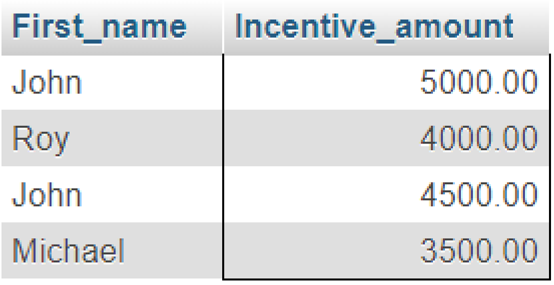
```
SELECT * FROM Employee ORDER BY Salary ASC;
```



Employee_id	First_name	Last_name	Salary ▲ 1	Joining_date	Department
4	Tom	Jose	600000.00	2013-01-07 12:00:00	Insurance
8	TestName2	LnameTest	600000.00	2013-01-12 12:00:00	Insurance
5	Jerry	Pinto	650000.00	2013-01-09 12:00:00	Insurance
7	TestName1	123	650000.00	2013-01-12 12:00:00	Services
3	Roy	Thomas	700000.00	2013-01-07 12:00:00	Banking
6	Philip	Matthew	750000.00	2013-01-08 12:00:00	Services
2	Michael	Clarke	800000.00	2013-01-05 12:00:00	Insurance
1	John	Abraham	1000000.00	2013-01-03 12:00:00	Banking

9. Select first_name, incentive amount from employee and incentives table forthose employees who have incentives and incentive amount greater than3000

```
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	5000.00
Roy	4000.00
John	4500.00
Michael	3500.00

10. Create After Insert trigger on Employee table which insert records inviewtable

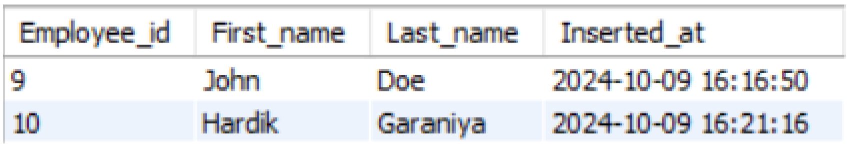
Create Table:

```
CREATE TABLE ViewTable (Employee_id INT,

 First_name VARCHAR(100),Last_name VARCHAR(100),

Inserted_at DATETIME);
```

Create Trigger:

```
DELIMITER $$


CREATE TRIGGER after_employee_insert

AFTER INSERT ON Employee

FOR EACH ROW

BEGIN

   INSERT INTO ViewTable (Employee_id, First_name, Last_name, Inserted_at)

   VALUES (NEW.Employee_id, NEW.First_name, NEW.Last_name, NOW());

END$$


DELIMITER ;
```

```
INSERT INTO Employee VALUES

 (9, John, Deo, 50000.00, '2024-10-09', 'HR'),

(10, 'Hardik', 'Garaniya', 50000.00, '2024-10-09', 'HR');
```



Employee_id	First_name	Last_name	Inserted_at
9	John	Doe	2024-10-09 16:16:50
10	Hardik	Garaniya	2024-10-09 16:21:16

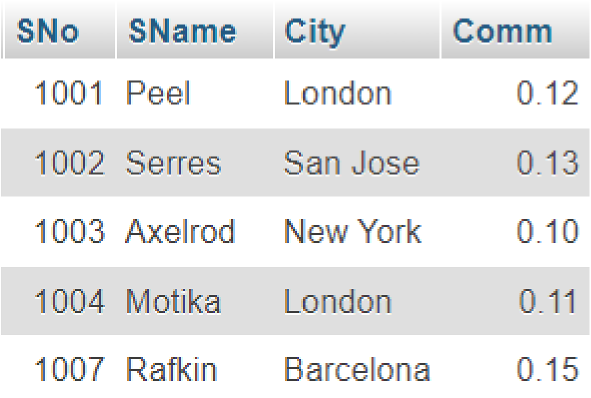
11.Create table given below: Salesperson and Customer

Create Table:

```
CREATE TABLE Salesperson ( SNo INT PRIMARY KEY, SName
 VARCHAR(100), City VARCHAR(100), Comm DECIMAL(3, 2) );
```

Insert Data:

```
INSERT INTO Salesperson VALUES
(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', 0.10);
```



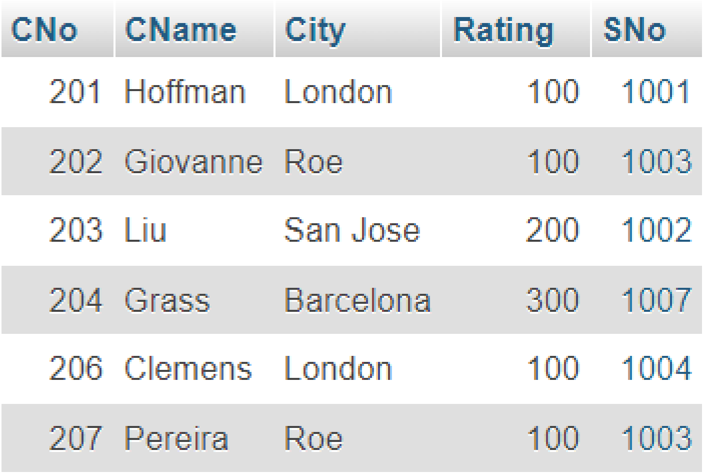
SNo	SName	City	Comm
1001	Peel	London	0.12
1002	Serres	San Jose	0.13
1003	Axelrod	New York	0.10
1004	Motika	London	0.11
1007	Rafkin	Barcelona	0.15

Create Table:

```
CREATE TABLE Customer ( CNo INT PRIMARY KEY, CName

VARCHAR(100), City VARCHAR(100), Rating INT, SNo INT,

 FOREIGN KEY (SNo)REFERENCES Salesperson(SNo) );
```

Insert Data:

```
INSERT INTO Customer VALUES

 (201, 'Hoffman', 'London', 100, 1001),

 (202, 'Giovanne', 'Roe', 100, 1003),

 (203, 'Liu', 'San Jose', 200, 1002),

 (204, 'Grass', 'Barcelona', 300, 1007),

 (206, 'Clemens', 'London', 100, 1004),

 (207, 'Pereira', 'Roe', 100, 1003);
```



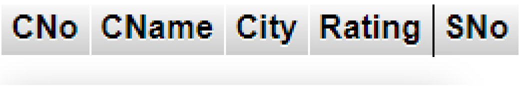
CNo	CName	City	Rating	SNo
201	Hoffman	London	100	1001
202	Giovanne	Roe	100	1003
203	Liu	San Jose	200	1002
204	Grass	Barcelona	300	1007
206	Clemens	London	100	1004
207	Pereira	Roe	100	1003

12.Retrieve the below data from above table

13.All orders for more than $1000

Select Query:

```
Select * from customer where Rating > 1000;
```

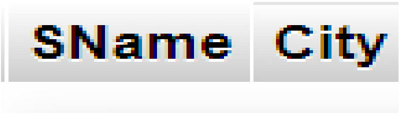


CNo	CName	City	Rating	SNo

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

Select Query:

```
SELECT SName, City FROM Salesperson

WHERE City = 'London' AND Comm > 0.12;
```

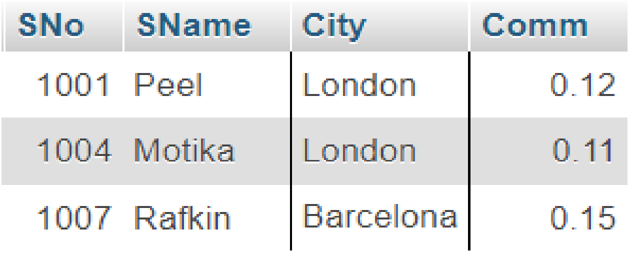


SName	City

15. All salespeople either in Barcelona or in London

Select Query:

```
SELECT * FROM Customer WHERE City = 'Barcelona' OR City = 'London';
```

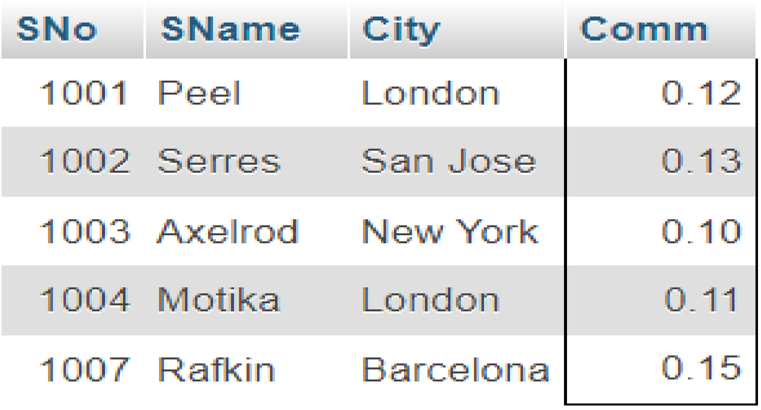


SNo	SName	City	Comm
1001	Peel	London	0.12
1004	Motika	London	0.11
1007	Rafkin	Barcelona	0.15

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

Select Query:

```
SELECT * FROM Salesperson WHERE Comm > 0.10 OR

 Comm < 0.12;
```

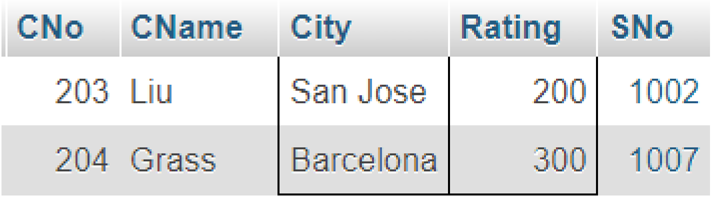


SNo	SName	City	Comm
1001	Peel	London	0.12
1002	Serres	San Jose	0.13
1003	Axelrod	New York	0.10
1004	Motika	London	0.11
1007	Rafkin	Barcelona	0.15

17.All customers excluding those with rating <= 100 unless they are locatedinRome

Select Query:

```
Select * From Customer Where (Rating > 100) OR

(City = 'Rome');
```



CNo	CName	City	Rating	SNo
203	Liu	San Jose	200	1002
204	Grass	Barcelona	300	1007

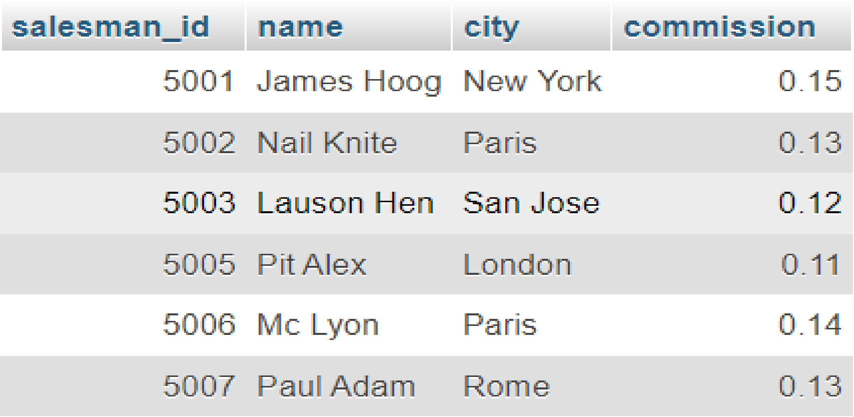
18. Write a SQL statement that displays all the information about all salespeople

Create Table:

```
CREATE TABLE Salespeople ( salesman_id INT PRIMARY KEY,

 name VARCHAR(100), city VARCHAR(100), commission

 DECIMAL(3, 2) );
```

Insert Data:

```
INSERT INTO Salespeople 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);
```



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

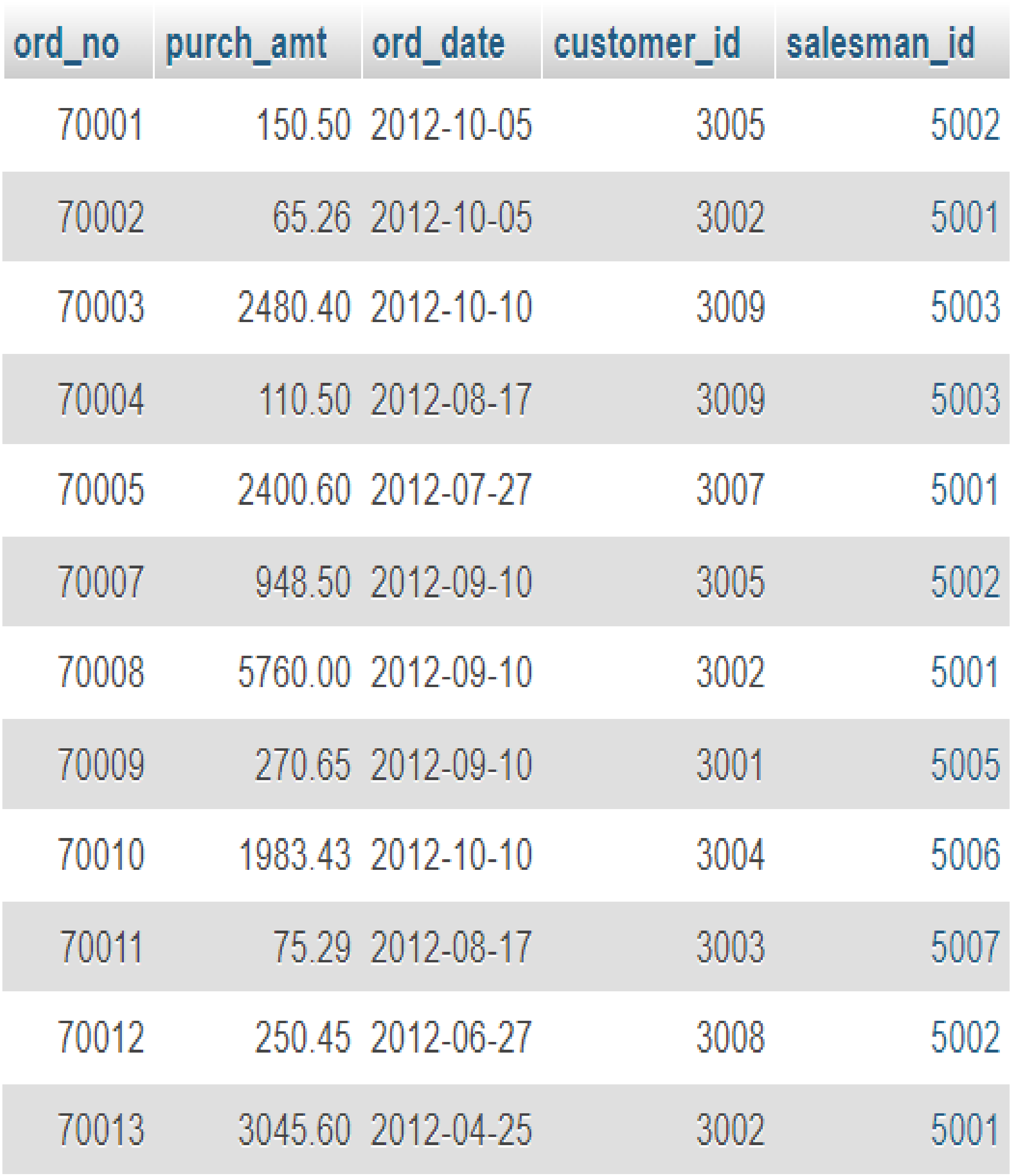
19. From the following table, write a SQL query to find orders that are delivered by a salesperson with ID. 5001. Return ord_no, ord_date, purch_amt.

Create Table:

```
CREATE TABLE Orders ( ord_no INT PRIMARY KEY, purch_amt
 DECIMAL(10, 2), ord_date DATE, customer_id INT,
salesman_id INT, FOREIGN KEY (salesman_id) REFERENCES
Salespeople(salesman_id) );
```

Insert Data:

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



ord_no	purch_amt	ord_date	customer_id	salesman_id
70001	150.50	2012-10-05	3005	5002
70002	65.26	2012-10-05	3002	5001
70003	2480.40	2012-10-10	3009	5003
70004	110.50	2012-08-17	3009	5003
70005	2400.60	2012-07-27	3007	5001
70007	948.50	2012-09-10	3005	5002
70008	5760.00	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	3045.60	2012-04-25	3002	5001

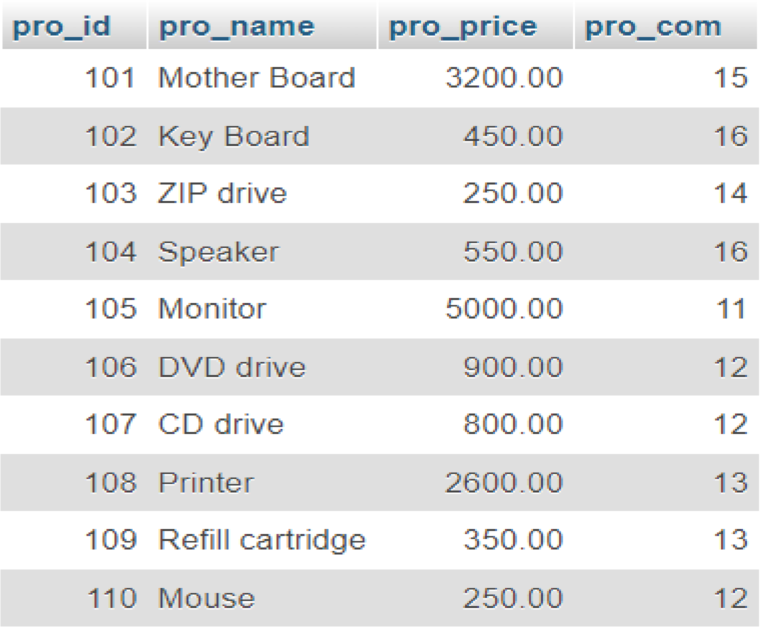
20. From the following table, write a SQL query to select a range of products whose price is in the range Rs.200 to Rs.600. Begin and end values are included. Return pro_id, pro_name, pro_price, and pro_com.

Create Table:

```
CREATE TABLE item_mast ( pro_id INT PRIMARY KEY, pro_name VARCHAR(100),

 pro_price DECIMAL(10, 2), pro_com INT );
```

Insert Data:

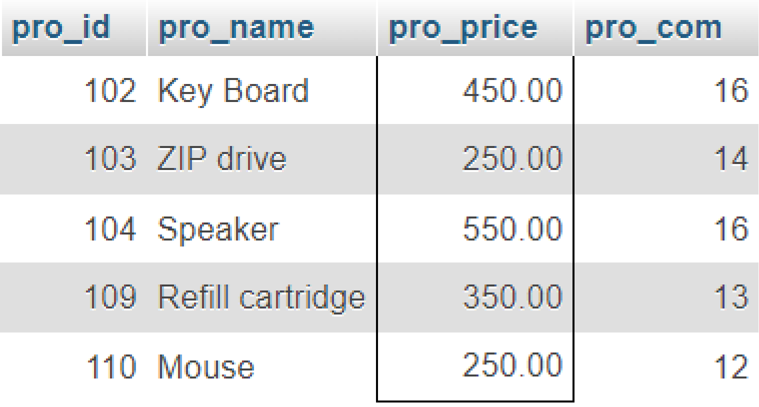
```
INSERT INTO item_mast VALUES
 (101, 'Mother Board', 3200.00, 15),
 (102, 'Key Board', 450.00, 16), (103, 'ZIP drive', 250.00, 14),
 (104, 'Speaker', 550.00, 16), (105, 'Monitor', 5000.00, 11),
 (106, 'DVD drive', 900.00, 12),
 (107, 'CD drive', 800.00, 12),
 (108, 'Printer', 2600.00, 13),
 (109,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



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

Select Query:

```
SELECT * FROM item_mast WHERE pro_price BETWEEN 200 AND 600;
```

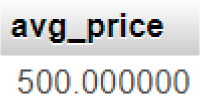


pro_id	pro_name	pro_price	pro_com
102	Key Board	450.00	16
103	ZIP drive	250.00	14
104	Speaker	550.00	16
109	Refill cartridge	350.00	13
110	Mouse	250.00	12

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

Select Query:

```
SELECT AVG(pro_price) AS avg_price FROM item_mast WHERE pro_com = 16;
```

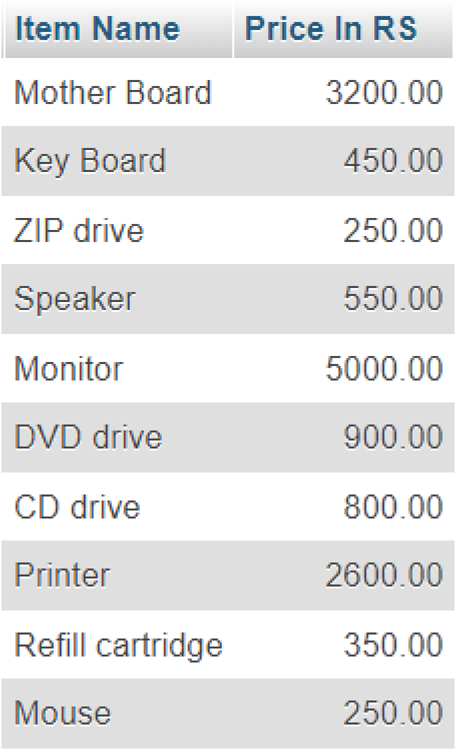


avg_price
500.000000

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

Select Query:

```
SELECT pro_name AS 'Item Name', pro_price AS 'Price In RS' FROM item_mast;
```

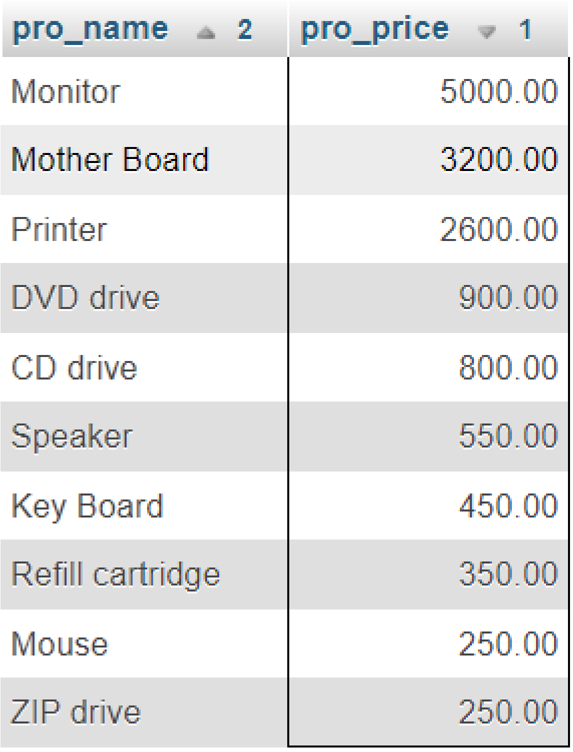


Item Name	Price In RS
Mother Board	3200.00
Key Board	450.00
ZIP drive	250.00
Speaker	550.00
Monitor	5000.00
DVD drive	900.00
CD drive	800.00
Printer	2600.00
Refill cartridge	350.00
Mouse	250.00

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

Select Query:

```
SELECT pro_name, pro_price FROM item_mast WHERE pro_price >= 250

ORDER BY pro_price DESC, pro_name ASC;
```

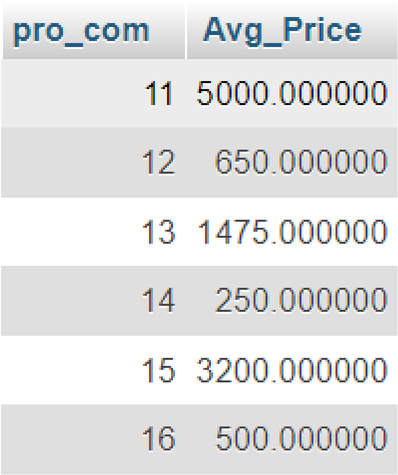


pro_name ▲ 2	pro_price ▼ 1
Monitor	5000.00
Mother Board	3200.00
Printer	2600.00
DVD drive	900.00
CD drive	800.00
Speaker	550.00
Key Board	450.00
Refill cartridge	350.00
Mouse	250.00
ZIP drive	250.00

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.

Select Query:

```
SELECT pro_com,  AVG(pro_price)  AS  Avg_Price  FROM  item_mast

 GROUP BY pro_com;
```



pro_com	Avg_Price
11	5000.000000
12	650.000000
13	1475.000000
14	250.000000
15	3200.000000
16	500.000000