MODULE -5 (DATABASE)

Basic of Database

1. What do you understand By Database

A database is a structured collection of data that is organized in a way that makes it easy to manage, retrieve, and update.

1. What is Normalization?

Normalization is a database design technique used to organize and structure data in a relational database efficiently. The process involves breaking down large tables into smaller, related tables and defining relationships between them.

1. What is Difference between DBMS and RDBMS?

|  |  |
| --- | --- |
| DBMS  [DATABASE MANAGEMENT SYSTEM] | RDBMS  [RELATIONAL DATABASE MANAGEMENT SYSTEM] |
| A DBMS is a software system that provides an interface for interacting with databases. | An RDBMS is a specific type of DBMS that is based on the relational model of data |
| The data can be organized in various ways | An RDBMS strictly adheres to the relational model. |
| In DBMS we can do creation, retrieval, updating, and management of data in a database | It organizes data into tables with rows and columns |

1. What is EF Cod Rule of RDBMS Systems?

Dr Edgar F Codd, after his extensive research on the Relational Model of database systems, came up with twelve rules of his own, which according to him, a database must obey in order to be regarded as a true relational database.

Rules are as follow:

**Rule 1**: Information Rule

The data stored in a database, may it be user data or metadata, must be a value of some table cell. Everything in a database must be stored in a table format

**Rule 2**: Guaranteed Access Rule

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

No other means, such as pointers, can be used to access data.

**Rule 3**: Systematic Treatment of NULL Values

The NULL values in a database must be given a systematic and uniform treatment. This is a very important rule because a NULL can be interpreted as one the following - data is missing, data is not known, or data is not applicable.

**Rule 4**: Active Online Catalog

The structure description of the entire database must be stored in an online catalog. known as data dictionary, which can be accessed by authorized users. Users can use the same query language to access the catalog which they use to access the database itself.

**Rule 5**: Comprehensive Data Sub-Language Rule

A database can only be accessed using a language having linear syntax that supports data definition, data manipulation, and transaction management operations. This language can be used directly or by means of some application. If the database allows access to data without any help of this language, then it is considered as a violation.

**Rule 6**: View Updating Rule

All the views of a database, which can theoretically be updated, must also be updatable by the system.

**Rule 7**: High-Level Insert, Update, and Delete Rule

A database must support high-level insertion, updation and deletion. This must not be limited to a single row, that is, it must also support union, intersection and minus operations to yield sets of data records

**Rule 8**: Physical Data Independence

The data stored in a database must be independent of the applications that access the database. Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.

**Rule 9**: Logical Data Independence

The logical data in a database must be independent of its user's view (application). Am change in logical data must not affect the applications using it. For example, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application. This is one of the most difficult rule to apply.

**Rule 10**: Integrity Independence

A database must be independent of the application that uses it. All its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and it interface.

**Rule 11**: Distribution Independence

The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only. This rule has been regarded as the foundation of the distributed database systems.

**Rule 12**: Non-Subversion Rule

If a system has an interface that provides access to low-levels records, then the interface must not be able to subvert the system and bypass security and integrity constraints.

1. What do you understand By Data Redundancy?

Data redundancy refers to the unnecessary repetition or duplication of data within a database. When the same data is stored in multiple places, it can lead to various issues and inefficiencies. Redundancy is considered undesirable in database design because it can result in problems such as increased storage requirements, inconsistencies, and difficulties in maintaining data integrity.

1. What is DDL Interpreter?

DDL interpreter includes statements for creating, altering, and deleting database objects such as tables, indexes, and views. The DDL interpreter handles errors that may occur during the execution of DDL statements. DDL (Data Definition Language) is a subset of SQL (Structured Query Language) that deals with the definition and management of database structures.

1. What is DML Compiler in SQL?

The SQL commands that deal with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements. It is the component of the SQL statement that controls access to data and to the database. Basically, DCL statements are grouped with DML statements.

Example: INSERT, UPDATE, DELETE

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

In SQL, key constraints define the rules for how data integrity is maintained within a relational database.

There are several types of key constraints, including PRIMARY KEY, UNIQUE, and FOREIGN KEY

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

A save point in SQL is a point within a transaction to which you can later roll back. It allows you to set intermediate points within a transaction, and if an error occurs, you can roll back to a specific save point instead of rolling back the entire transaction.

Syntax for creating save point: SAVEPOINT your\_savepoint\_name;

Using save point with Roleback:

ROLLBACK TO your\_savepoint\_name;

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

Ans.) A trigger in SQL is a stored procedure that automatically executes in response to a specific event, such as an INSERT, UPDATE, DELETE, or TRUNCATE statement, on a specified table or view.

Syntax:-

CREATE TRIGGER [schema\_name.]trigger\_name

ON table\_name

AFTER {[INSERT],[UPDATE],[DELETE]}

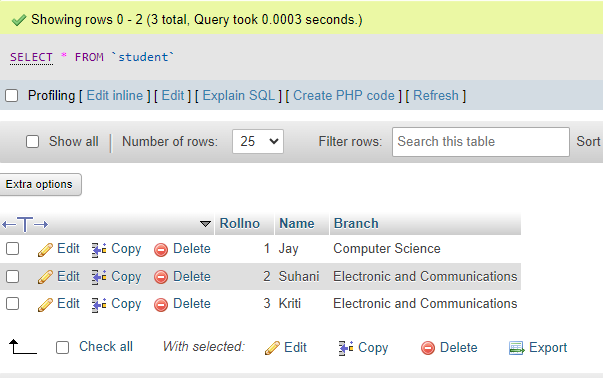
[NOT FOR REPLICATION]

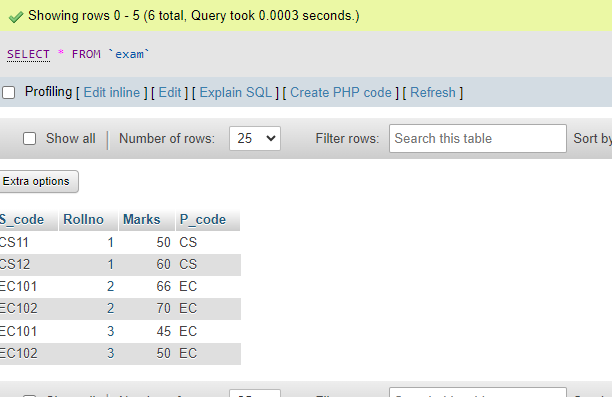
AS

{sql\_statements}

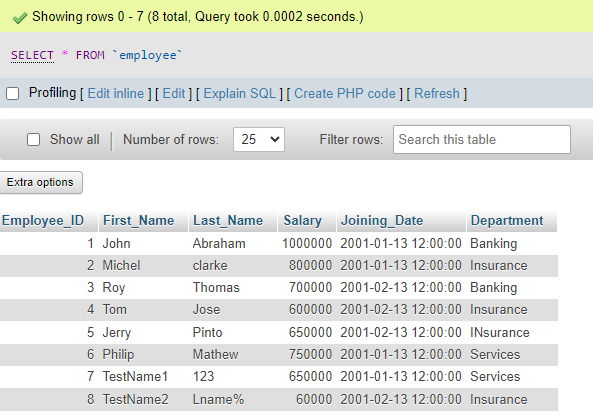
**SQL Queries**

1. Create Table Name: Student and Exam

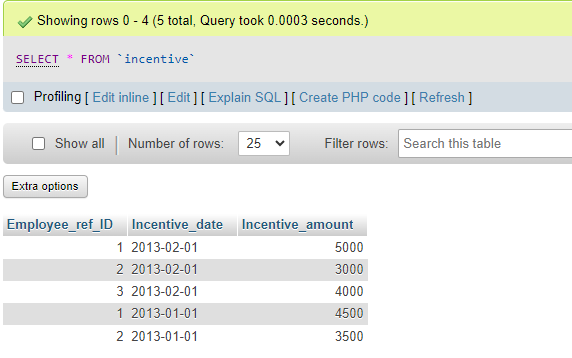




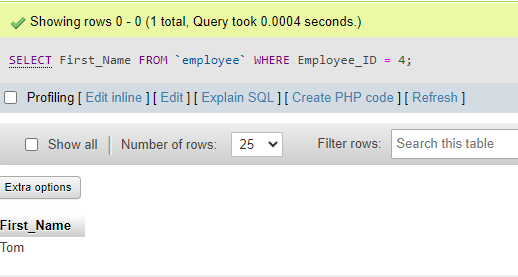
1. Employee and Incentive Table



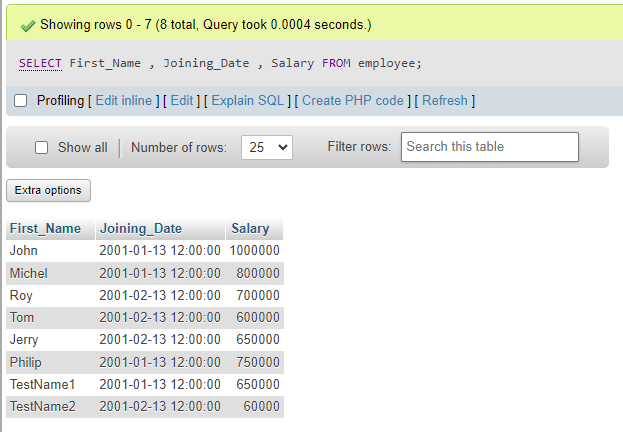
Incentive Table



1. Get First\_Name from employee table using Tom name “Employee Name”.

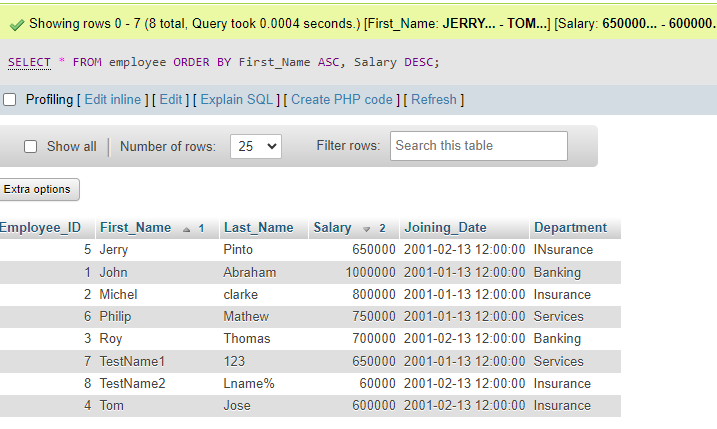


1. Get FIRST\_NAME, Joining Date, and Salary from employee table.

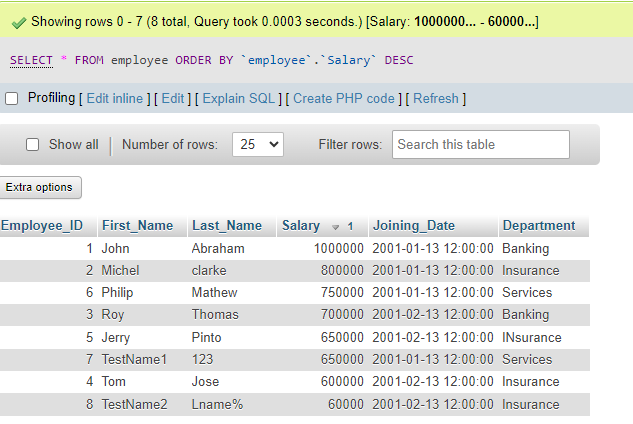


1. Get all employee details from the employee table order by First\_Name Ascending and Salary descending?

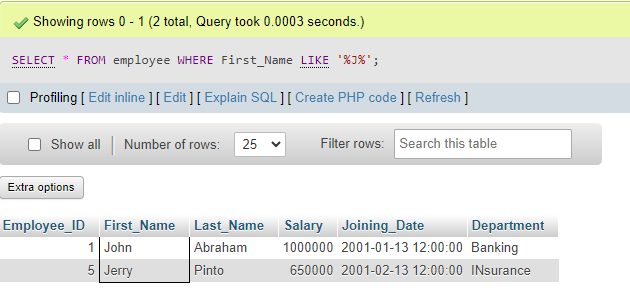
First\_Name Ascending:



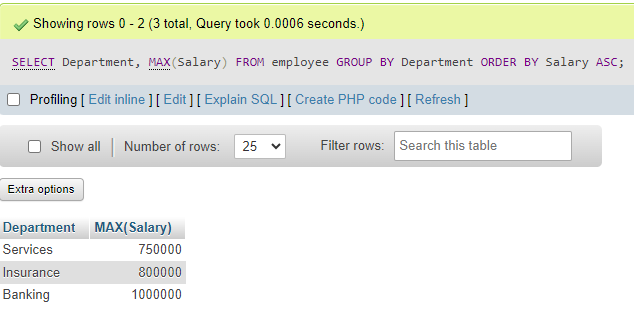
Salary Descending:



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



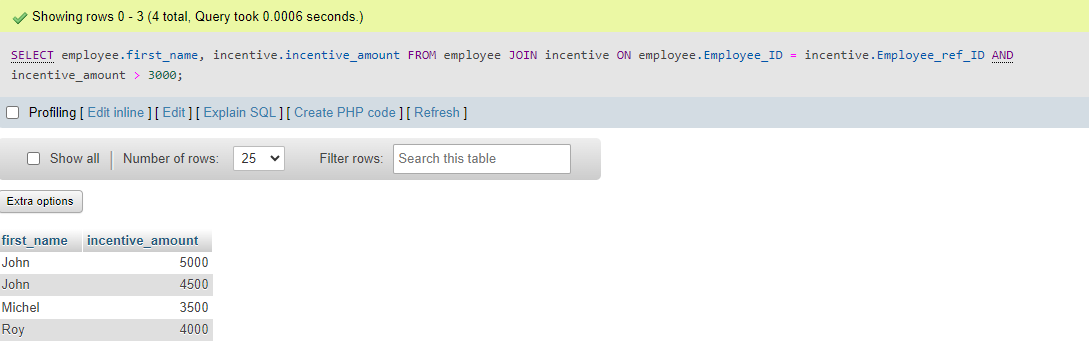
1. Get department wise maximum salary from employee table order by salary ascending?



9. Select first\_name, incentive amount from employee and incentives table

For those employees who have incentives and incentive amount greater than

3000?



10. Create After Insert trigger on Employee table which insert records in view Table.

A trigger is a stored procedure in database which automatically invokes whenever a special event in database occurs.

for example: a trigger can be invoked when a row is inserted into a specific table.

CREATE TRIGGER trg\_InsertEmployee

AFTER INSERT

ON employee

FOR EACH ROW

BEGIN

INSERT INTO reminders

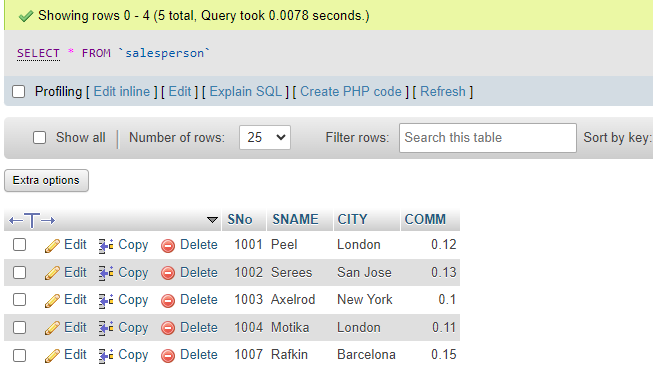
VALUES (NEW.employee\_id, NEW.First\_Name, NEW.Department);

END;

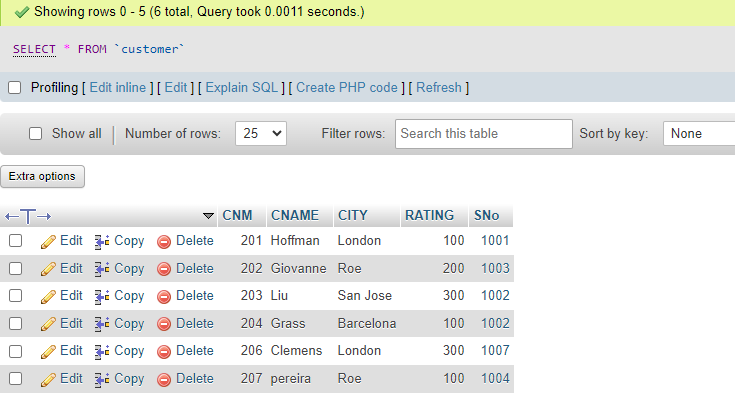
DELIMITER ;

1. Create table given below: Salesperson and Customer:

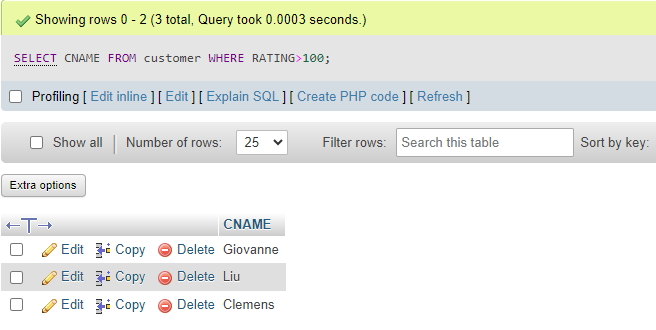
SALES PERSON TABLE



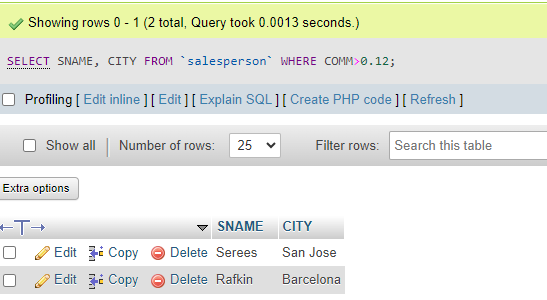
CUSTOMER TABLE



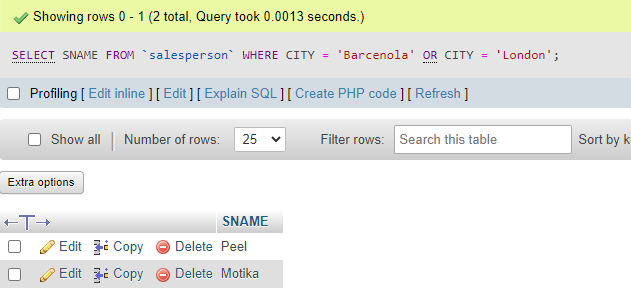
1. Retrieve the below data from above table
2. All Customer name whose rating is more than 100.



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



1. All salespeople either in Barcelona or in London



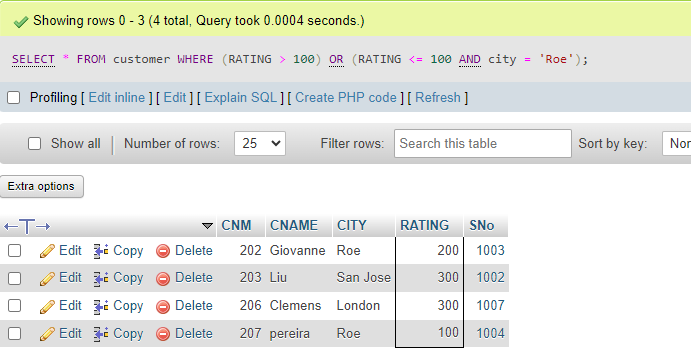
1. All salespeople with commission between 0.10 and 0.12. (Boundary

Values should be excluded).



1. All customers excluding those with rating <= 100 unless they are located

In Rome



1. Write a SQL statement for table

CREATE table sales\_person(salesman\_ID int , name varchar(225), city varchar(225), commission float);

INSERT INTO `sales\_person`(`salesman\_ID`, `name`, `city`, `commission`) VALUES ('5001','James Hoog','New York','0.15');

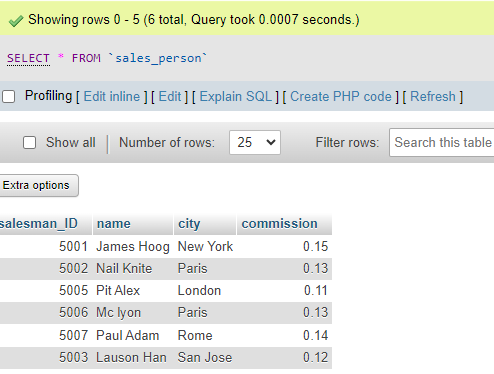
INSERT INTO `sales\_person`(`salesman\_ID`, `name`, `city`, `commission`) VALUES ('5002','Nail Knite','Paris','0.13');

INSERT INTO `sales\_person`(`salesman\_ID`, `name`, `city`, `commission`) VALUES ('5005','Pit Alex','London','0.11');

INSERT INTO `sales\_person`(`salesman\_ID`, `name`, `city`, `commission`) VALUES ('5006','Mc lyon','Paris','0.13');

INSERT INTO `sales\_person`(`salesman\_ID`, `name`, `city`, `commission`) VALUES ('5007','Paul Adam','Rome','0.14');

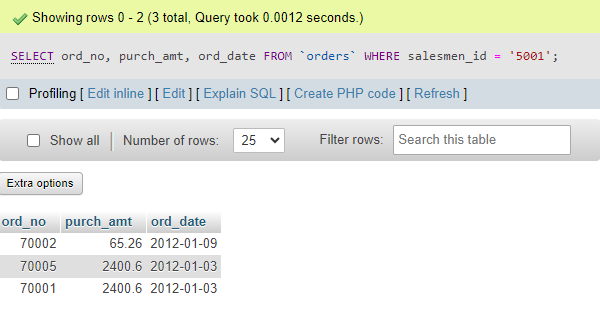
INSERT INTO `sales\_person`(`salesman\_ID`, `name`, `city`, `commission`) VALUES ('5003','Lauson Han','San Jose','0.12');



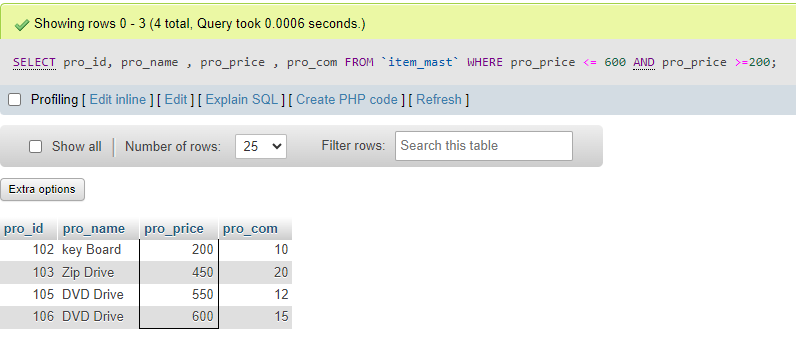
1. From the following table, write a SQL query to find orders that are

delivered by a salesperson with ID. 5001. Return ord\_no, ord\_date,

purch\_amt.



1. 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

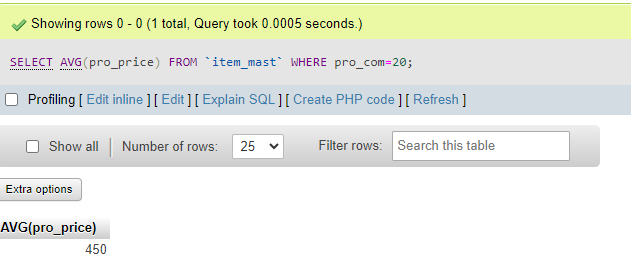


1. From the following table, write a SQL query to calculate the average price for a manufacturer code of 20 Return avg.

SELECT AVG(pro\_price)

FROM `item\_mast`

WHERE pro\_com=20;

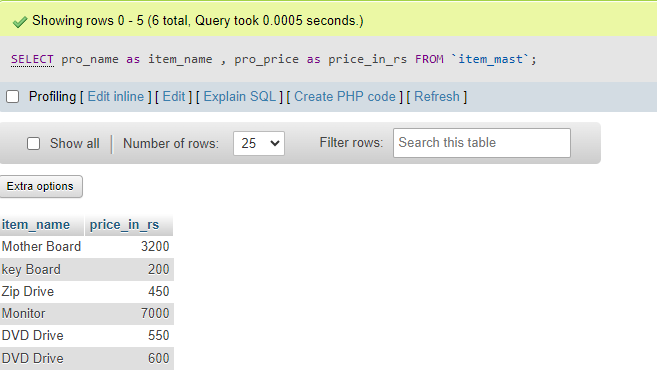


1. From the following table, write a SQL query to display the pro\_name as 'Item Name' and pro\_price as 'Price in Rs.'

Query :

SELECT pro\_name as item\_name , pro\_price as price\_in\_rs

FROM `item\_mast` ;

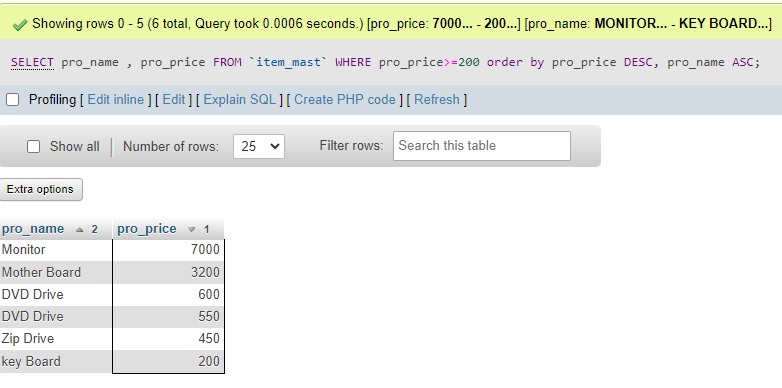


1. From the following table, write a SQL query to find the items whose

prices are higher than or equal to 200. Order the result by product price in

descending, then product name in ascending. Return pro\_name and

pro\_price.



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

