* ***TOPICS COVERED***

**\*\*\* Basic of Database \*\*\***

**Qua.1 -> What do you understand By Database**

**Ans.** A database is an organized collection of structured information or data, typically stored electronically in a computer system. It serves as a central repository for storing, managing, and retrieving information efficiently. Databases are essential tools for businesses, organizations, and individuals seeking to store, analyse, and share data effectively.

* **Key Functions of a Database :-**
* **Data Storage:**

Databases provide a centralized location for storing data, ensuring easy accessibility and management.

* **Data Analysis:**

Databases facilitate the analysis of data, enabling the identification of trends, patterns, and valuable insights.

* **Data Sharing:**

Databases enable the sharing of data among users, both within an organization and externally.

* **Core Components of a Database :-**
* **Tables:**

Data is organized into tables, which consist of rows and columns.

* **Relationships:**

Tables can be related to each other through defined relationships, creating complex data structures.

* **Queries:**

Queries are used to retrieve and manipulate data from the database, allowing users to extract specific information.

* **Security:**

Databases incorporate security measures to protect data from unauthorized access and ensure data integrity.

* **Types of Databases :-**
* **Relational Databases:**

These databases store data in tables with predefined relationships between them. They are widely used for business applications due to their structure and query capabilities.

* **NoSQL Databases:**

Designed for large, unstructured data sets, NoSQL databases are suitable for handling flexible and dynamic data, often found in social media and big data applications.

* **Cloud Databases:**

Hosted on cloud computing platforms, these databases offer scalability, flexibility, and reduced infrastructure management.

* **Additional Terms and Concepts :-**
* **Database Management System (DBMS):**

Software that manages and controls the database, providing tools for creating, modifying, and querying data.

* **Data Integrity:**

Ensuring the accuracy, consistency, and reliability of data within the database.

* **Data Redundancy:**

The duplication of data, which can lead to inconsistencies and inefficiencies.

* **Data Warehousing:**

The process of storing and managing large volumes of historical data for analysis and reporting purposes.

* **Data Mining:**

The process of extracting patterns and knowledge from large data sets.

In essence, a database is a powerful tool for managing information and making data-driven decisions. By understanding the key components, types, and functions of databases, you can effectively leverage their capabilities to support your organization's needs.

-------------------------------------------------------------------------------------------------------------------------

**Qua.2 -> What is Normalization?**

**Ans.** Normalization is a process of organizing data in a database to minimize redundancy and improve data integrity. It involves breaking down large tables into smaller, more manageable ones, ensuring that each column in a table depends only on the primary key.

-----------------------------------------------------------------------------------------------------------------------------------

**Qua.3 -> What is Difference between DBMS and RDBMS?**

**Ans.**

|  |  |  |
| --- | --- | --- |
| **Feature** | **DBMS** | **RDBMS** |
| ***Data Storage Format*** | Files | Tables |
| ***Data Access*** | Individual elements | Multiple elements together |
| ***Data Relationships*** | No relationships between data | Data linked through tables |
| ***Distributed Database*** | Not supported | Supported |
| ***Data Volume*** | Small to medium | Large |
| ***User Access*** | Single user | Multiple users |
| ***Software/Hardware*** | Lower requirements | Higher requirements |
| ***Examples*** | XML, Microsoft Access | Oracle, SQL Server, MySQL |
| ***Similar Words*** | Database Management System | Relational Database Management System |
| ***Additional Notes*** | Can handle various data types (structured, semi-structured, unstructured) | Enforces data integrity through normalization |

-----------------------------------------------------------------------------------------------------------------------------------

**Qua.4 -> What is MF Cod Rule of RDBMS Systems?**

**Ans.**

The term "MF Cod Rule" isn't a standard term in RDBMS. It likely refers to the Codd's Normalization Rules, also known as the Data Normalization Rules. These rules are a set of guidelines used to design efficient and effective relational databases.

Data normalization focuses on minimizing data redundancy and ensuring data integrity. It involves structuring data into tables with well-defined relationships between them. Codd's Normalization Rules define different levels of data organization, ensuring data consistency and reducing the risk of errors.

Here's a breakdown of the most common Codd's Normalization Rules:

1. First Normal Form (1NF):

* Similar terms: Atomic Values, Single Valued Attribute
* Explanation: Each cell (intersection of a row and column) in a table must contain a single, atomic value. This means no repeating groups or lists within a single column. For example, instead of storing multiple phone numbers in a single column, you would create a separate "Phone Numbers" table linked to the main table.

2. Second Normal Form (2NF):

* Similar terms: Full Dependency, Partial Dependency
* Explanation: A table must be in 1NF and all non-key attributes (columns that aren't part of the primary key) must be fully dependent on the entire primary key. This eliminates partial dependencies where some non-key attributes depend only on a portion of the primary key.

3. Third Normal Form (3NF):

* Similar terms: Transitive Dependency
* Explanation: 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 eliminates transitive dependencies where a non-key attribute relies on another non-key attribute, which in turn relies on the primary key.

Benefits of Data Normalization:

* Reduced data redundancy
* Improved data integrity
* Efficient data retrieval and manipulation
* Easier maintenance and updates.

Choosing the Right Normalization Level:

The optimal level of normalization depends on the specific needs of your database. Higher levels (3NF and beyond) offer greater data integrity but may increase complexity. Carefully consider the trade-off between data integrity and query efficiency when designing your database structure.

-------------------------------------------------------------------------------------------------------------------------

**Qua.5 -> What do you understand By Data Redundancy?**

**Ans.** Data redundancy occurs when the same data is stored in multiple places within a database. This can lead to inconsistencies and inefficiencies. Normalization is used to minimize data redundancy.

-------------------------------------------------------------------------------------------------------------------------

**Qua.6 -> What is DDL Interpreter?**

**Ans.** A DDL (Data Definition Language) Interpreter is a component of a DBMS that processes DDL statements. DDL statements are used to define the structure of a database, including creating, modifying, and deleting tables, indexes, and constraints.

-------------------------------------------------------------------------------------------------------------------------

**Qua.7 -> What is DML Compiler in SQL?**

**Ans.** A DML (Data Manipulation Language) Compiler is a component of a DBMS that processes DML statements. DML statements are used to manipulate data within a database, such as inserting, updating, and deleting records

-------------------------------------------------------------------------------------------------------------------------

**Qua.8 -> What is SQL Key Constraints writing an Example of SQL Key Constraints**

**Ans.** SQL Key Constraints are rules applied to a table to enforce data integrity and consistency. They help maintain the accuracy and reliability of data by limiting the types of values that can be stored in a column. When a constraint is violated, the action that would have modified the data is typically aborted.

Here's a breakdown of the commonly used SQL key constraints:

**1. NOT NULL:-**

* Purpose: Ensures that a column cannot contain NULL values.
* Similar terms: Mandatory, Required

**2. UNIQUE:-**

* Purpose: Guarantees that all values in a column or combination of columns are distinct.
* Similar terms: One-of-a-kind, Distinct

**3. PRIMARY KEY:-**

* Purpose: A combination of NOT NULL and UNIQUE. Uniquely identifies each row in a table.
* Similar terms: Identifier, Key

**4. FOREIGN KEY:-**

* Purpose: Establishes a relationship between two tables by referencing a primary key in another table. Prevents actions that would disrupt these relationships.
* Similar terms: Referential Constraint, Relationship

**5. CHECK:-**

* Purpose: Enforces a specific condition on the values in a column or combination of columns.
* Similar terms: Condition, Rule

**6. DEFAULT:-**

* Purpose: Sets a default value for a column if no value is provided during data insertion.
* Similar terms: Default Value, Automatic Value

**7. CREATE INDEX:-**

* Purpose: Creates an index on a column or combination of columns to improve query performance.
* Similar terms: Index, Lookup Table

-------------------------------------------------------------------------------------------------------------------------

**Qua.9 -> What is save Point? How to create a save Point write a Query?**

**Ans.** A savepoint is a marker within a transaction that allows you to rollback to a specific point in case of errors or unexpected conditions. It provides more granular control over transactions compared to simply rolling back the entire transaction.

Syntax for Savepoint command : SAVEPOINT SAVEPOINT\_NAME;

-------------------------------------------------------------------------------------------------------------------------

**Qua.10 -> What is trigger and how to create a Trigger in SQL?**

**Ans.**

**Triggers in SQL: Automating Database Actions**

Triggers are powerful tools in SQL that act as stored procedures automatically executed in response to specific events on a table. These events can be data insertion (INSERT), modification (UPDATE), or deletion (DELETE). Triggers play a crucial role in enforcing business rules, maintaining data integrity, and performing database auditing tasks.

Creating a Trigger in SQL: Step-by-Step

1. **CREATE TRIGGER statement:** This statement initiates the creation of a new trigger.
2. **Trigger Name:** Assign a unique name to your trigger within the database.
3. **Trigger Type:** Define the type of event that triggers the execution:
   * **FOR INSERT:** Trigger fires after an INSERT operation.
   * **FOR UPDATE:** Trigger fires after an UPDATE operation.
   * **FOR DELETE:** Trigger fires after a DELETE operation.
4. **Table and Event:** Specify the table on which the trigger should act and the event that triggers its execution.
5. **Trigger Code:** This is the core of the trigger. It comprises the SQL code that will be executed when the trigger is fired. This code can perform various actions like:
   * Inserting data into another table
   * Updating existing data
   * Calling other stored procedures
   * Sending notifications
   * Performing data validation

-------------------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------------------

* ***Topics Covered***

**\*\*\* SQL Queries \*\*\***

|  |  |
| --- | --- |
| **Qua.1 ->** | **Create Table Name : Student and Exam ()** |
|  |  |
| **Ans.** | create table Student  (  Roll\_no int PRIMARY KEY,  Name varchar(40),  Branch varchar(40)  ); |
|  | 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'); |
|  |  |
|  | CREATE TABLE Exam  (  Roll\_no int,  S\_code varchar(40),  Marks int ,  P\_code varchar(40),  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'); |
|  |  |
|  | -------------------------------------------------------------------------------------------------------------------------**-----------** |
| **Qua.2 ->** | **Create table given below: Employee and Incentive Table \*()** |
|  |  |
|  | **Name: Employee**  **Table Name: Incentive** |
|  |  |
| **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  insert into employee values(8, 'TestName2', 'Pinto',600000,'01-FAB-13 12.00.00AM','Insurance');  insert into employee values(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); |
|  |  |
|  | -------------------------------------------------------------------------------------------------------------------------**-----------** |
| **Qua.3 ->** | **Get First\_Name from employee table using Tom name “Employee Name”.** |
| **Ans.** | SELECT First\_name FROM employee WHERE First\_name='Tom'; |
|  |  |
|  | -------------------------------------------------------------------------------------------------------------------------**-----------** |
| **Qua.4 ->** | **Get FIRST\_NAME, Joining Date, and Salary from employee table.** |
| **Ans.** | SELECT First\_name ,Joining\_date,Salary FROM employee; |
|  |  |
|  | -------------------------------------------------------------------------------------------------------------------------**-----------** |
| **Qua.5 ->** | **Get all employee details from the employee table order by First\_Name Ascending and Salary descending?** |
| **Ans.** | SELECT \* FROM employee ORDER BY First\_name ,Salary DESC ; |
|  |  |
|  | -------------------------------------------------------------------------------------------------------------------------**-----------** |
| **Qua.6 ->** | **Get employee details from employee table whose first name contains ‘J’.** |
| **Ans.** | SELECT \* FROM employee WHERE First\_name LIKE'J%'; |
|  |  |
|  | -------------------------------------------------------------------------------------------------------------------------**-----------** |
| **Qua.7 ->**  **Qua.8 ->** | **Get department wise maximum salary from employee table order by**  **salary ascending?** |
| **Ans.** | SELECT Department, MAX(Salary) as max\_Salary  FROM employee  GROUP BY Department  ORDER BY max\_Salary ASC; |
|  |  |
|  | -------------------------------------------------------------------------------------------------------------------------**-----------** |
| **Qua.9 ->** | **Select first\_name, incentive amount from employee and incentives table for those employees who have incentives and incentive amount 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 |
|  |  |
|  | -------------------------------------------------------------------------------------------------------------------------**-----------** |
| **Qua.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  BEGININSERT 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,'Reco rd inserted'); END;  insert into employee values(8, 'TestName2', 'Pinto',600000,'01-FAB-13 12.00.00AM','Insurance'); |
|  |  |
|  | -------------------------------------------------------------------------------------------------------------------------**-----------** |
| **Qua.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'); |
|  |  |
|  | 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); |
|  |  |
| **-------------** | -------------------------------------------------------------------------------------------------------------------------**-----------** |
|  | **Retrieve the below data from above table** |
|  |  |
| **Qua.12 ->** | **All orders for more than $1000** |
| **Ans.** | SELECT \* FROM orders WHERE amount > 1000; |
|  |  |
|  | -------------------------------------------------------------------------------------------------------------------------**-----------** |
| **Qua.13 ->** | **Names and cities of all salespeople in London with commission above 0.12** |
| **Ans.** | SELECT \* FROM salesperson WHERE CITY ='London'AND COMM>.12; |
|  | 🡪 Error |
|  | -------------------------------------------------------------------------------------------------------------------------**-----------** |
| **Qua.14 ->** | **All salespeople either in Barcelona or in London** |
| **Ans.** | SELECT \* FROM salesperson WHERE CITY='Barcelona' OR CITY='London'; |
|  |  |
|  | -------------------------------------------------------------------------------------------------------------------------**-----------** |
| **Qua.15 ->** | **All salespeople with commission between 0.10 and 0.12. (Boundary**  **valuesshould be excluded).** |
| **Ans.** | SELECT \* FROM salesperson WHERE COMM BETWEEN .10 AND .12; |
|  |  |
|  | -------------------------------------------------------------------------------------------------------------------------**-----------** |
| **Qua.16 ->** | **All customers excluding those with rating <= 100 unless they are located**  **In Rome** |
| **Ans.** | SELECT \* FROM customer WHERE RATING <=100 AND CITY='Reo'; |
|  |  |
|  | -------------------------------------------------------------------------------------------------------------------------**-----------** |
| **Qua.17 ->** | **Write a SQL statement that displays all the information about all salespeople** |
|  | **salesman\_id | name | city | commission**  5001 | James Hoog | New York | 0.15  5002 | Nail Knite | Paris | 0.13  5005 | Pit Alex | London | 0.11  5006 | Mc Lyon | Paris | 0.14  5007 | Paul Adam | Rome | 0.13  5003 | Lauson Hen | San Jose | 0.12 |
| **Ans.** | SELECT \* FROM salesperson ; |
|  |  |
|  | -------------------------------------------------------------------------------------------------------------------------**-----------** |
| **Qua.18 ->** | **.From the following table, write a SQL query to find orders that are delivered by a salesperson with ID. 5001. Return ord\_no, ord\_date, purch\_amt.**  ***Sample table*: orders** |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | ord\_no | purch\_amt | ord\_date | customer\_id | **salesman\_id** | | 70001 | 150.5 | 2012-10-05 | 3005 | **5002** | | 70009 | 270.65 | 2012-09-10 | 3001 | **5005** | | 70002 | 65.26 | 2012-10-05 | 3002 | **5001** | | 70004 | 110.5 | 2012-08-17 | 3009 | **5003** | | 70007 | 948.5 | 2012-09-10 | 3005 | **5002** | | 70005 | 2400.6 | 2012-07-27 | 3007 | **5001** | | 70008 | 5760 | 2012-09-10 | 3002 | **5001** | | 70010 | 1983.43 | 2012-10-10 | 3004 | **5006** | | 70003 | 2480.4 | 2012-10-10 | 3009 | **5003** | | 70012 | 250.45 | 2012-06-27 | 3008 | **5002** | | 70011 | 75.29 | 2012-08-17 | 3003 | **5007** | | 70013 | 3045.6 | 2012-04-25 | 3002 | **5001** | |
| **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'); |
|  |  |
|  | 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); |
|  |  |
|  | SELECT ord\_no, ord\_date, purch\_amt FROM orders WHERE salesman\_id = 5001; |
|  |  |
|  | -------------------------------------------------------------------------------------------------------------------------**-----------** |
| **Qua.19 ->** | **From the following table, write a SQL query to select a range of products whose price is in the range Rs.200 to Rs.600. Begin and end values are included. Return pro\_id, pro\_name, pro\_price, and pro\_com.**  ***Sample table*: item\_mast**  **PRO\_ID PRO\_NAME PRO\_PRICE PRO\_COM** |
|  | |  |  |  |  | | --- | --- | --- | --- | | 101 | Mother Board | 3200.00 | 15 | | 102 | Key Board | 450.00 | 16 | | 103 | ZIP drive | 250.00 | 14 | | 104 | Speaker | 550.00 | 16 | | 105 | Monitor | 5000.00 | 11 | | 106 | DVD drive | 900.00 | 12 | | 107 | CD drive | 800.00 | 12 | | 108 | Printer | 2600.00 | 13 | | 109 | Refill cartridge | 350.00 | 13 | | 110 | Mouse | 250.00 | 12 | |
| **Ans.** | 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); |
|  | SELECT pro\_id, pro\_name, pro\_price, pro\_com FROM item\_mast WHERE pro\_price BETWEEN 200 AND 600; |
|  |  |
|  | -------------------------------------------------------------------------------------------------------------------------**-----------** |
| **Qua.20 ->** | **From the following table, write a SQL query to calculate the averageprice for a manufacturer code of 16. Return avg.**  ***Sample table*: item\_mast**  **PRO\_ID PRO\_NAME PRO\_PRICE PRO\_COM** |
| **Ans.** | |  |  |  |  | | --- | --- | --- | --- | | 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 AVG(PRO\_PRICE)AS avg FROM item\_mast WHERE PRO\_COM = 16; |
|  |  |
|  | -------------------------------------------------------------------------------------------------------------------------**-----------** |
| **Qua.21 ->** | **From the following table, write a SQL query to display the pro\_nameas 'Item Name' and pro\_priceas 'Price in Rs.'**  ***Sample table*: item\_mast**  **PRO\_ID PRO\_NAME PRO\_PRICE PRO\_COM** |
| **Ans.** | |  |  |  |  | | --- | --- | --- | --- | | 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 PRO\_NAME AS 'Item Name', PRO\_PRICE AS 'Price in Rs.' FROM item\_mast; |
|  |  |
|  | -------------------------------------------------------------------------------------------------------------------------**-----------** |
| **Qua.22 ->** | **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.**  ***Sample table*: item\_mast**  **PRO\_ID PRO\_NAME PRO\_PRICE PRO\_COM** |
|  | |  |  |  |  | | --- | --- | --- | --- | | 101 | Mother Board | 3200.00 | 15 | | 102 | Key Board | 450.00 | 16 | | 103 | ZIP drive | 250.00 | 14 | | 104 | Speaker | 550.00 | 16 | | 105 | Monitor | 5000.00 | 11 | | 106 | DVD drive | 900.00 | 12 | | 107 | CD drive | 800.00 | 12 | | 108 | Printer | 2600.00 | 13 | | 109 | Refill cartridge | 350.00 | 13 | | 110 | Mouse | 250.00 | 12 | |
| **Ans.** | SELECT PRO\_NAME, PRO\_PRICE FROM item\_mast WHERE PRO\_PRICE >= 250 ORDER BY PRO\_NAME ASC, PRO\_PRICE DESC; |
|  |  |
|  | -------------------------------------------------------------------------------------------------------------------------**-----------** |
| **Qua.23 ->** | **From the following table, write a SQL query to calculate average price ofthe items for each company. Return average price and companycode.**  ***Sample table*: item\_mast**  **PRO\_ID PRO\_NAME PRO\_PRICE PRO\_COM** |
| **Ans.** | SELECT AVG(PRO\_PRICE) AS average\_price, PRO\_COM AS companycode FROM item\_mast GROUP BY PRO\_COM; |
|  |  |
| ------------- | -------------------------------------------------------------------------------------------------------------------------**-----------** |
| ------------- | -------------------------------------------------------------------------------------------------------------------------**-----------** |