|  |
| --- |
| 1. What do you understand By Database  A database is a structured collection of data organized in a way that a computer program can quickly search and retrieve specific pieces of data. It is designed to efficiently manage, store, and retrieve information. Databases are used in various applications and scenarios where large amounts of structured data need to be organized, accessed, and manipulated.  Key characteristics of databases include:   1. **Structured Data:** Data in a database is typically organized in tables with predefined columns and data types. This structured format allows for efficient storage and retrieval of information. 2. **Query Language:** Databases are accessed and manipulated using a query language, commonly SQL (Structured Query Language). SQL allows users to interact with the database by performing operations such as inserting, updating, deleting, and retrieving data. 3. **ACID Properties:** Transactions in databases adhere to ACID properties—Atomicity, Consistency, Isolation, and Durability. These properties ensure the reliability and consistency of the data, even in the event of system failures. 4. **Concurrent Access:** Databases allow multiple users or applications to access and modify data concurrently. Proper mechanisms are in place to manage concurrent access and maintain data integrity. 5. **Scalability:** Databases can scale to handle large amounts of data and increasing numbers of users. This scalability is crucial for applications that need to accommodate growth over time.   There are different types of databases, including:   * **Relational Databases:** Use tables to store data and establish relationships between tables. Examples include MySQL, PostgreSQL, and Oracle Database. * **NoSQL Databases:** Designed to handle unstructured or semi-structured data. Examples include MongoDB (document-oriented), Cassandra (wide-column store), and Redis (key-value store). * **In-memory Databases:** Store data primarily in the system's main memory (RAM) for faster access. Examples include Redis and Memcached.   Databases are fundamental to many applications, ranging from simple data storage to complex systems powering enterprise-level applications, e-commerce platforms, content management systems, and more.  2. What is Normalization?  Normalization is a process used in database design to organize a relational database structure efficiently and reduce data redundancy. The goal of normalization is to eliminate or minimize data anomalies, such as insertion, update, and deletion anomalies, by breaking down large tables into smaller, well-structured tables. This helps ensure data integrity and maintainability in a relational database.  The normalization process involves applying a set of rules, usually up to a certain normal form, to a database schema. The most commonly used normal forms are the first normal form (1NF), second normal form (2NF), and third normal form (3NF). Higher normal forms, such as Boyce-Codd normal form (BCNF) and fourth normal form (4NF), exist but are less commonly used in practice.  Here's a brief overview of the first three normal forms:   1. **First Normal Form (1NF):** A table is in 1NF if it contains only atomic (indivisible) values, and there are no repeating groups or arrays of values in any column. Each attribute must contain only a single value. 2. **Second Normal Form (2NF):** A table is in 2NF if it is in 1NF and all non-key attributes are fully functionally dependent on the primary key. This means that there should be no partial dependencies, where only part of a composite key determines an attribute. 3. **Third Normal Form (3NF):** A table is in 3NF if it is in 2NF and there are no transitive dependencies. Transitive dependencies occur when a non-key attribute is dependent on another non-key attribute.   By applying normalization techniques, databases become more flexible, scalable, and less prone to data anomalies. However, it's important to note that normalization also introduces additional tables and relationships, which may slightly impact query performance. Database designers must strike a balance between normalization and denormalization based on the specific requirements and performance considerations of the application.  3. What is Difference between DBMS and RDBMS?  DBMS (Database Management System) and RDBMS (Relational Database Management System) are terms often used in the context of databases, but they have distinct differences:  **DBMS (Database Management System):**   1. **Definition:**    * DBMS is a software that manages databases. It provides an interface for interacting with the database, and it offers tools for creating, retrieving, updating, and managing data in the database. 2. **Data Model:**    * DBMS may or may not follow a specific data model. It can be hierarchical, network, or relational, among others. 3. **Schema:**    * In a DBMS, the schema defines the structure of the database, but it may not enforce relationships or constraints between tables. 4. **Data Integrity:**    * DBMS may or may not support referential integrity and other advanced constraints. 5. **Example:**    * Examples of DBMS include Microsoft Access, SQLite, and FileMaker.   **RDBMS (Relational Database Management System):**   1. **Definition:**    * RDBMS is a type of DBMS that specifically follows the relational model of data. It organizes data into tables with rows and columns, and it establishes relationships between tables. 2. **Data Model:**    * RDBMS strictly adheres to the relational data model. It uses tables to store data and supports the principles of normalization. 3. **Schema:**    * RDBMS enforces relationships and constraints between tables, maintaining data integrity. 4. **Data Integrity:**    * RDBMS typically supports referential integrity, ensuring that relationships between tables are maintained. 5. **Example:**    * Examples of RDBMS include MySQL, PostgreSQL, Oracle Database, and Microsoft SQL Server.   In summary, while all RDBMS systems are DBMS, not all DBMS systems are RDBMS. RDBMS is a specific type of DBMS that follows the principles of the relational model, enforces relationships and constraints, and supports SQL (Structured Query Language) for managing and querying data.  5. What do you understand By Data Redundancy?  Data redundancy refers to the duplication of data in a database or information system. It occurs when the same piece of data is stored in multiple places within a database or across different databases. While some level of redundancy is often unavoidable, excessive redundancy can lead to various issues in a database system. Here are some key points to consider:   1. **Wasted Storage:** Storing the same information in multiple locations consumes additional storage space, which may be unnecessary and inefficient. 2. **Update Anomalies:** Redundant data can lead to inconsistencies when updates are made to one instance of the data but not to others. This can result in data inconsistencies and errors. 3. **Insertion Anomalies:** Inserting new data into the database may become problematic if all instances of the redundant data are not updated simultaneously. This can lead to incomplete or inaccurate information. 4. **Deletion Anomalies:** Removing data from one location may cause issues if the same data is needed elsewhere. This can result in unintentional loss of related information. 5. **Complexity:** Redundant data can make database systems more complex and harder to maintain, especially as the volume of data increases.   To mitigate data redundancy, database designers often use normalization techniques. Normalization involves organizing data in a way that minimizes redundancy and dependency. This typically includes breaking down large tables into smaller, related tables and establishing relationships between them.  It's important to strike a balance between normalization and performance, as overly normalized databases may require complex joins and impact query performance. Database designers need to carefully consider the specific requirements and characteristics of the application when addressing data redundancy issues.  Top of Form  6. What is DDL Interpreter?  DDL (Data Definition Language) is a subset of SQL (Structured Query Language) used to define and manage the structure of a relational database. DDL statements are responsible for defining, altering, and dropping database objects such as tables, indexes, and views. The DDL interpreter, in the context of a database management system (DBMS), is the component or module that processes and executes these DDL statements.  The DDL interpreter performs the following key functions:   1. **Syntax Checking:** It checks the syntax of the DDL statements to ensure they adhere to the rules and structure of the SQL language. If there are syntax errors, it reports them to the user. 2. **Authorization and Security:** The DDL interpreter checks whether the user executing the DDL statement has the necessary privileges to perform the requested operation. It enforces security rules defined in the database. 3. **Metadata Management:** DDL statements define the structure of the database objects, and the DDL interpreter updates the metadata or data dictionary of the database. Metadata includes information about tables, columns, constraints, and other database objects. 4. **Object Creation and Modification:** The DDL interpreter is responsible for creating new database objects (e.g., tables) or modifying existing ones based on the DDL statements provided by the user. 5. **Transaction Handling:** DDL statements often implicitly commit the current transaction, as they involve changes to the database structure that should be permanent. The DDL interpreter manages these transactions accordingly.   Example DDL statements include:   * **CREATE TABLE:** Defines a new table with its columns, data types, and constraints. * **ALTER TABLE:** Modifies the structure of an existing table, such as adding or dropping columns. * **DROP TABLE:** Removes an existing table from the database.   In summary, the DDL interpreter plays a crucial role in processing and executing DDL statements, ensuring the integrity and security of the database structure.  7. What is DML Compiler in SQL?  In the context of SQL (Structured Query Language) and database management systems (DBMS), the term "DML Compiler" is not a standard term used to describe a specific component or process. However, I can provide information about the components related to SQL and DML (Data Manipulation Language).  DML (Data Manipulation Language) in SQL consists of the statements used for querying and modifying data in a database. Common DML statements include SELECT (for querying data), INSERT (for adding new records), UPDATE (for modifying existing records), and DELETE (for removing records).  In a database system, the processing of SQL queries and DML statements generally involves several components:   1. **Parser:** The parser checks the syntax of the SQL statements to ensure they adhere to the language rules. It breaks down the statements into a parse tree, a hierarchical structure that represents the syntactic structure of the SQL statement. 2. **Optimizer:** The optimizer analyzes the parse tree and generates an optimized execution plan for the query or DML statement. The goal is to determine the most efficient way to access and manipulate the data, considering factors such as indexes and available resources. 3. **Executor:** The executor is responsible for executing the optimized plan generated by the optimizer. It interacts with the storage engine to retrieve or modify the data based on the SQL statements. 4. **Transaction Manager:** This component manages transactions, ensuring the consistency and integrity of the database. It handles aspects such as commit and rollback operations.   While these components are involved in processing SQL queries and DML statements, the term "DML Compiler" is not a standard term used in the same way. If you have a specific context or a different term in mind, please provide additional details, and I'll do my best to assist you.  Top of Form    8. What is SQL Key Constraints writing an Example of SQL Key Constraints  In SQL, key constraints are used to enforce the integrity and uniqueness of data in a relational database. There are different types of key constraints, including PRIMARY KEY, UNIQUE, and FOREIGN KEY.  **PRIMARY KEY Constraint:**   * The PRIMARY KEY constraint uniquely identifies each record in a table. * It must contain unique values and cannot have NULL values.   **UNIQUE Constraint:**   * The UNIQUE constraint ensures that all values in a column are distinct, similar to the PRIMARY KEY constraint. However, unlike PRIMARY KEY, a table can have multiple UNIQUE constraints.   **FOREIGN KEY Constraint:**   * The FOREIGN KEY constraint establishes a link between two tables by referencing a column in one table to the primary key column in another table. * It ensures referential integrity by preventing actions that would break the relationships between tables.   9. What is save Point? How to create a save Point write a Query?  A savepoint in a relational database is a point within a transaction to which you can roll back the transaction without rolling back the entire transaction. Savepoints are useful in situations where you want to apply partial changes and have the option to roll back to a specific point if needed.  10.What is trigger and how to create a Trigger in SQL?  In SQL, a trigger is a set of instructions that are automatically executed ("triggered") in response to certain events on a particular table or view. These events can include data manipulation events (such as INSERT, UPDATE, DELETE), or specific database actions (such as database startup). Triggers are commonly used to enforce business rules, perform logging, or maintain data integrity. |
|  |
| 1. Create Table Name : Student and Exam    [create](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [table](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) student  (   Roll\_no int primary key,   Name varchar(20),  Address varchar(30),  Branch varchar(30);  [create](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [table](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) Exam  (   Roll\_no int,   S\_code varchar(20),  Marks varchar(30),  P\_code varchar(30)  Foreign key(Roll\_no) reference student(Roll\_no); |
| 2. Create table given below: Employee and IncentiveTable  [create](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [table](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) sco  (   firs\_tname varchar(20),   Last\_name varchar(20),  Address varchar(30),  City varchar(10),  age int  );  insert into sco  VALUES  ('micky','mouse','123fantasy way','anahelm',73 ),  ('bat','man','321fantasy way','gotham',54),  ('wonder','woman',' 987truth way','paradise',39 ),  ('donald','duck','555quack stree','mallard',65 ),  ('bugs','bunny','567carrot street','rascal',58 ),  ('cat','woman','234purrfect street','hairball',32 ),  ('micky','mouse','999fantasy way','anahelm',73 ),  ('tweety','bird','543fantasy way','ititlaw',28 );    [CREATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) incentives(employe\_ref\_id int,incentives\_date date,inc\_amount INT, FOREIGN KEY(employe\_ref\_id) REFERENCES employee(employe\_id));  [INSERT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/insert.html) INTO incentives(employe\_ref\_id,incentives\_date,inc\_amount) [VALUES](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_values) (1,2013-01-02,5000), (2,2013-01-02,3000), (3,2013-01-02,4000), (1,2013-01-01,4500), (1,2013-01-01,3500);    3. Get First Name from employee table using Tom name “Employee Name”.  [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM employee WHERE first\_name='tom;';    4. Get FIRST\_NAME, Joining Date, and Salary from employee table.  [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) DISTINCT first\_name,joining\_date,salary FROM employee;    5. Get all employee details from the employee table order by First Name Ascending and Salary descending?  (a) First name ascending  [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM employee ORDER by first\_name;    (b) Salary descending  [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM employee ORDER by salary DESC;    6. Get employee details from employee table whose first name contains ‘J’.  [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM employee WHERE first\_name [LIKE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/string-comparison-functions.html%23operator_like) 'j%';    7. Get department wise maximum salary from employee table order by  [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) [max](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/aggregate-functions.html%23function_max)(salary) as maximum\_salary FROM employee;    8. salary ascending?  [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* from employee ORDER by salary;    9. Select first name, incentive amount from employee and incentives table  for those employees who have incentives and incentive amount greater than 3000  first give primary id to employe id in employe table  ALTER TABLE employee ADD PRIMARY KEY employee(employe\_id);  Than give foreign key id to employe\_ref\_id in incentives table.  than  [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) e.first\_name, i.inc\_amount FROM employee e INNER JOIN incentives i ON e.employe\_id = i.employe\_ref\_id WHERE i.inc\_amount > 3000;    10. Create After Insert trigger on Employee table which insert records in view table  [CREATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) view\_table  (  id int,  name varchar(30),  date\_time timestamp,  action\_ text);  [CREATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-trigger.html) [TRIGGER](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-trigger.html) tri\_insert AFTER [INSERT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/insert.html) on employee   for EACH ROW  BEGIN  [INSERT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/insert.html) INTO view\_table(id,name,action\_) [VALUES](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_values) (new.first\_name,'Record inserted');  END; |
| 11.Create table given below: Salesperson and Customer    [CREATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) salesperson  (Sno int PRIMARY KEY,  Sname varchar(20),  city varchar(20),  COMM int);  [INSERT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/insert.html) into salesperson  (Sno,Sname,city,COMM)[VALUES](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_values)  (1001,'Peel','London',12),  (1002,'Serres','San jose',13),  (1004,'Mortika','London',11),  (1007,'Rafkin','Barcelona',15),  (1003,'Axelord','New york',1);    For coustomer  [CREATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [table](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) Customer  (CNM int PRIMARY KEY,  CNAME varchar(20),  CITY varchar(20),  RATING int,  Sno1 int,  FOREIGN key(Sno1) REFERENCES salesperson(Sno));  [INSERT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/insert.html) INTO customer(CNM,CNAME,CITY,RATING,Sno1)  [VALUES](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_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);    12.Retrieve the below data from above table  SELECT  SALSEPERSON.SNAME AS Salesperson\_Name,  CUSTOMER.CNAME AS Customer\_Name,  CUSTOMER.CITY AS City,  CUSTOMER.RATING AS Rating,  CUSTOMER.COMM AS Commission  FROM  SALSEPERSON  JOIN  CUSTOMER ON SALSEPERSON.SNo = CUSTOMER.`FK SNo`;  13.All orders for more than $1000.  SELECT CNAME, CITY, RATING, COMM  FROM CUSTOMER c  JOIN TABLE\_I i ON c.SNo = i.SNo  WHERE RATING \* 100 > 1000;  14.Names and cities of all salespeople in London with commission above 0.12  [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM salesperson WHERE city='london' [AND](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_and) COMM>0.12;    15.All salespeople either in Barcelona or in London    [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM `salesperson` WHERE city='london' [or](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_or) city='barcelona';    16.All salespeople with commission between 0.10 and 0.12.(Boundary valuesshould be excluded).  [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM salesperson WHERE COMM>0.10 [AND](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_and) COMM<0.12;  17.All customers excluding those with rating <= 100 unless they are locatedinRome  [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM customer WHERE RATING<=100; |
| 18.Write a SQL statement that displays all the information about all salespeople    [CREATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) salesperson\_1  (salesman\_id int,name varchar(20),city varchar(20),commission float );  [INSERT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/insert.html) INTO salesperson\_1  (salesman\_id,name,city,commission)  [VALUES](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_values)  (5001,'James Hoog','New York',0.15),  (5002,'Nail Alex','Paris',0.13),  (5005,'Pit Alex','London',0.11),  (5006,'Mc Lyon','Paris',0.14),  (5003,'Lauson hen','San Jose',0.12); |
| 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](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) orders  (ord\_no int,punch\_amt int,  ord\_date date,  customer\_id int,salesman\_id\_1 INT,  FOREIGN KEY(salesman\_id\_1)REFERENCES salesperson\_1(salesman\_id));  [INSERT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/insert.html) INTO orders(ord\_no,ord\_date,customer\_id,salesman\_id\_1,purch\_amt)  [VALUES](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_values)  (70001,2012-10-05,3005,5002,1505),  (70009,2012-09-10,3005,5002,270.65),  (70002,2012-10-05,3002,5001,65.26),  (70004,2012-08-17,3009,5003,110.5),  (70007,2012-09-17,3005,5002,948.5),  (70005,2012-07-27,3007,5001,2400.6),   (70008,2012-09-10,3002,5001,5760),  (70010,2012-10-10,3004,5006,1983.43),   (70003,2012-10-10,3009,5003,2480.4),   (70012,2012-06-27,3008,5002,250.45),   (70011,2012-08-17,3003,null,75.29),  (70013,2012-04-25,3002,5001,3045.6);    [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM orders WHERE salesman\_id\_1=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. Sample table: item mast    [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM `item\_mast` WHERE pro\_price BETWEEN '200' [AND](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_and) '600'; |
| 21.From the following table, write a SQL query to calculate the average price for a manufacturer code of 16. Return avg. Sample table: item mast  [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) [AVG](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/aggregate-functions.html%23function_avg) (pro\_price)avg\_pro\_price FROM item\_mast; |
| 22.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  [ALTER](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) item\_mast CHANGE pro\_name item\_name varchar(30);  [ALTER](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) item\_mast CHANGE pro\_price price\_in\_rs int; |
| 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. Sample table: item mast  [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM item\_mast WHERE price\_in\_rs>250; |
| 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. Sample table: item\_mast  [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) PRO\_COM AS company\_code, [AVG](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/aggregate-functions.html%23function_avg)(price\_in\_rs) AS average\_price FROM item\_mast GROUP BY PRO\_COM; |