**Basics of Database**

1. What do you understand By Database

A database is an organized collection of data, stored and accessed electronically. Databases are used to store and manage large amounts of structured and unstructured data, and they can be used to support a wide range of activities, including data storage, data analysis, and data management.

1. What is Normalization?

Normalization is a design of databases methodological method used in the to create a neat, structured, and structured table in which each table relates to just one subject or one-to-one correspondence. The objective is to extensively reduce data redundancy and dependency.

1. What is Difference between DBMS and RDBMS?

|  |  |
| --- | --- |
| DBMS | RDBMS |
| Database management system. | Relation database management system. |
| Data stored is in file format . | Data Stored is in table format. |
| Individual access of data element. | Multiple data element is accessible together. |
| No connection between data. | Data in the form of a table are linked together. |
| No support for distributed database . | Support distributed database. |
| Data stored is a small quantity. | Data is Stored in large amount. |
| DBMS support a single user. | RDBMS supports multiple users. |
| The software and hardware requirements are low. | The software and hardware requirement are higher. |
| Example: - XML, Microsoft Assess. | Example: - Oracle, SQL, Server. |

1. What is MF Cod Rule of RDBMS Systems?

The MF Cod Rule of RDBMS Systems states that for a system to qualify as an RDBMS, it must be able to manage database entirely through the relational capabilities Rule 0 of the MF Cod Rules states that the system must qualify as relational, as a database, and as a management system. For a system to qualify as an RDBMS, that system must use its relational facilities exclusively to manage the database.

1. What do you understand By Data Redundancy?

Data redundancy refers to the situation where the same pieces of data are stored in multiple places within a database or data storage system. This can happen intentionally or accidentally. Redundancy can be useful for data recovery in case of corruption or loss. In computer memory and storage, data redundancy allows for error correction.

1. What is DDL Interpreter?

DML Compiler: It processes the DML statements into low level instruction (machine language), so that they can be executed.

DDL Interpreter: It processes the DDL statements into a set of tables containing meta data (data about data).

1. What is DML Compiler in SQL?

The Data Manipulation Language, or DML for short, is the group of commands responsible for manipulating data in a database; this generally entails inserting, editing, or deleting rows in SQL tables.

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

Constraints are the rules that we can apply on the type of data in a table. That is, we can specify the limit on the type of data that can be stored in a particular column in a table using constraints.

The available constraints in SQL are:

* **NOT NULL:** This constraint tells that we cannot store a null value in a column. That is, if a column is specified as NOT NULL then we will not be able to store null in this particular column any more.
* **UNIQUE:** This constraint when specified with a column, tells that all the values in the column must be unique. That is, the values in any row of a column must not be repeated.
* **PRIMARY KEY:** A primary key is a field which can uniquely identify each row in a table. And this constraint is used to specify a field in a table as primary key.
* **FOREIGN KEY:** A Foreign key is a field which can uniquely identify each row in another table. And this constraint is used to specify a field as foreign key.

* **CHECK:** This constraint helps to validate the values of a column to meet a particular condition. That is, it helps to ensure that the value stored in a column meets a specific condition.
* **DEFAULT:** This constraint specifies a default value for the column when no value is specified by the user.

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

A save point in SQL is a logical rollback point within a transaction. It allows you to specify a point in a transaction that you can roll back to without affecting the entire transaction.

To create a, save point, use the following syntax: ‘SAVEPOINT savepoint\_name’. You can then perform various SQL operations within the transaction.

To roll back to a specific save point, use ‘ROLLBACK TO save\_point\_name’

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

Trigger is a statement that a system executes automatically when there is any modification to the database. In a trigger, we first specify when the trigger is to be executed and then the action to be performed when the trigger executes. Triggers are used to specify certain integrity constraints and referential constraints that cannot be specified using the constraint mechanism of SQL.  
**Example** –  
Suppose, we are adding a tuple to the ‘Donors’ table that is some person has donated blood. So, we can design a trigger that will automatically add the value of donated blood to the ‘Blood\_record’ table.

**Types of Triggers** –  
We can define 6 types of triggers for each table:

1. **AFTER INSERT:** activated after data is inserted into the table.
2. **AFTER UPDATE:** activated after data in the table is modified.
3. **AFTER** **DELETE:** activated after data is deleted/removed from the table.
4. **BEFORE** **INSERT:** activated before data is inserted into the table.
5. **BEFORE** **UPDATE:** activated before data in the table is modified.
6. **BEFORE** **DELETE:** activated before data is deleted/removed from the table.

**SQL** **Queries**

|  |  |
| --- | --- |
| 1. | Create Table Name : Student and Exam |
|  |  |
|  | CREATE DATABASE student\_db;  CREATE TABLE student  (  rollno int PRIMARY KEY,  s\_name VARCHAR(20),  branch VARCHAR(20)  );  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');  select \* FROM student; |
|  |  |
|  |  |
|  | CREATE TABLE Exam  (  rollno int ,  s\_code VARCHAR(10),  marks int,  p\_code VARCHAR(10)  FOREIGN KEY(rollno) REFERENCES student(rollno)  );  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');  SELECT \* FROM Exam; |
|  |  |
| 2. | Create table given below: Employee and IncentiveTable |
|  |  |
|  | CREATE TABLE Employee  (  e\_id int PRIMARY key,  First\_name VARCHAR(50),  Last\_name VARCHAR(50),  Salary int,  Joining\_Date VARCHAR(100),  Department VARCHAR(100)  );  INSERT INTO Employee VALUES(1,'Jhon','Abraham',1000000,'1-jan-13 12.00.00 AM','Banking');  INSERT INTO Employee VALUES(2,'Michael','Clarke',800000,'1-Jan-13 12.00.00 AM','Insurance');  INSERT INTO Employee VALUES(3,'Roy','Thomas',700000,'1-Feb-13 12.00.00 AM','Banking');  INSERT INTO Employee VALUES(4,'Tom','Josh',600000,'1-Feb-13 12.00.00 AM','Insurance');  INSERT INTO Employee VALUES(5,'Jerry','pinto',650000,'1-Feb-13 12.00.00 AM','Insurance');  INSERT INTO Employee VALUES(6,'Philip','Mathew',750000,'1-Jan-13 12.00.00 AM','Services');  INSERT INTO Employee VALUES(7,'TestName1','123',650000,'1-Jan-13 12.00.00 AM','Services');  INSERT INTO Employee VALUES(8,'TestName2','Lname%',600000,'1-Jan-13 12.00.00 AM','Insurance');  UPDATE Employee SET Joining\_Date='1-Jan-13 12.00.00 AM' WHERE e\_id=1;  SELECT \* FROM Employee; |
|  |  |
|  |  |
|  | CREATE TABLE Incentive  (  Employee\_ref\_id INT FOREIGN KEY REFERENCES Employee(e\_id),  Incentive\_date VARCHAR(50),  Incentive\_Ammount int,  );  INSERT INTO Incentive VALUES(1,'01-Feb-13',5000);  INSERT INTO Incentive VALUES(2,'01-Feb-13',3000);  INSERT INTO Incentive VALUES(3,'01-Feb-13',4000);  INSERT INTO Incentive VALUES(1,'01-Jan-13',4500);  INSERT INTO Incentive VALUES(2,'01-Jan-13',3500); |
|  |  |
| 3. | . Get First\_Name from employee table using Tom name “Employee Name” |
|  | SELECT First\_name AS Employee\_name FROM Employee WHERE First\_name='tom'; |
|  |  |
| 4. | . Get FIRST\_NAME, Joining Date, and Salary from employee table. |
|  | SELECT First\_name , Joining\_Date ,salary FROM Employee; |
|  |  |
| 5. | Get all employee details from the employee table order by First\_Name |
|  | SELECT \* FROM Employee ORDER BY First\_name , Salary DESC; |
|  |  |
| 6. | . Get employee details from employee table whose first name contains ‘J’. |
|  | SELECT \* FROM Employee WHERE First\_name LIKE 'J%'; |
|  |  |
| 7. | Get department wise maximum salary from employee table order by |
|  | SELECT Department , max(Salary) FROM employee  GROUP BY Department  ORDER BY max(Salary); |
|  |  |
| 8. | Salary ascending ? |
|  | SELECT Salary FROM Employee ORDER BY Salary; |
|  |  |
| 9. | Select first\_name, incentive amount from employee and incentives table forthose employees who have incentives and incentive amount greater than 3000 |
|  | SELECT First\_name , Incentive\_Ammount FROM Employee ,Incentive WHERE Salary>=3000; |
|  |  |
| 10. | Create After Insert trigger on Employee table which insert records in viewtable |
|  | create TRIGGER AfterinsertEmployee After INSERT ON Employee for each ROW  BEGIN  INSERT INTO viewtable (e\_id,First\_name,Last\_name,salary,Joining\_Date,Department)  VALUES (new.e\_id,new.First\_name,new.Last\_name,new.salary,new.Joining\_Date,new.Department)  END |
|  |  |
| 11. | Create table given below: Salesperson and Customer |
|  |  |
|  | CREATE TABLE SALSEPERSON  (  PK\_S\_No INT PRIMARY key,  S\_name VARCHAR(60),  City VARCHAR(40),  COMM VARCHAR(50)  );  INSERT INTO SALSEPERSON VALUES(1001,'peel','London','0.12');  INSERT INTO SALSEPERSON VALUES(1002,'Serres','Sen josh','0.13');  INSERT INTO SALSEPERSON VALUES(1004,'Motika','London','0.11');  INSERT INTO SALSEPERSON VALUES(1007,'Refkin','Barcelona','0.15');  INSERT INTO SALSEPERSON VALUES(1003,'Axelrod','New York','0.1');  update SALSEPERSON SET S\_name='Peel' WHERE S\_name='peel';  SELECT \* FROM SALSEPERSON; |
|  |  |
|  |  |
|  | CREATE TABLE CUSTOMER  (  PK\_CNM int PRIMARY KEY,  c\_name VARCHAR(100),  City VARCHAR(100),  RATING int,  FK\_SNo INT FOREIGN KEY REFERENCES SALSEPERSON(PK\_S\_No)  );  INSERT INTO CUSTOMER VALUES(201,'Hoffman','London',100,1001);  INSERT INTO CUSTOMER VALUES(202,'Giovanne','Roe',200,1003);  INSERT INTO CUSTOMER VALUES(203,'Liu','San josh',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','Roe',100,1004);  SELECT \* FROM CUSTOMER; |
|  |  |
| 12. | Retrieve the below data from above table |
| 13 | .All orders for more than $1000. |
|  | SELECT \* FROM SALSEPERSON WHERE orderr=1000; |
| 14. | Names and cities of all salespeople in London with commission above 0.12 |
|  | SELECT S\_name,City from SALSEPERSON WHERE City='london' AND COMM >=0.12 ; |
|  |  |
| 15. | All salespeople either in Barcelona or in London |
|  | SELECT S\_name , City FROM SALSEPERSON WHERE City='Barcelona' OR City='London'; |
|  |  |
| 16. | All salespeople with commission between 0.10 and 0.12. (Boundary valuesshould be excluded). |
|  | SELECT \* FROM SALSEPERSON WHERE COMM >0.10 and COMM < 0.12; |
|  |  |
| 17. | All customers excluding those with rating <= 100 unless they are located inRome |
|  | SELECT \* FROM CUSTOMER WHERE RATING > 100 OR (RATING <= 100 AND City='ROME' ); |
|  |  |
| 18. | .Write a SQL statement that displays all the information about all salespeople |
|  |  |
|  | CREATE TABLE salespeople  (  salesman\_id int,  name varchar(30),  city text,  commission VARCHAR(20)  ); |
|  |  |
|  | INSERT INTO salespeople VALUES(5001,'James Hoog','New York',0.15);  INSERT INTO salespeople VALUES(5002,'Nail Knite','paris',0.13);  INSERT INTO salespeople VALUES(5005,'Pit Alex','London',0.11);  INSERT INTO salespeople VALUES(5006,'Mc Lyon','paris',0.14);  INSERT INTO salespeople VALUES(5007,'Paul Adam','Rome',0.13);  INSERT INTO salespeople VALUES(5003,'Lauson Hen','San Jose',0.12);  SELECT \* FROM salespeople; |
|  |  |
| 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 orders  (  ord\_no int,  purch\_amt text,  ord\_date date,  customer\_id int,  salesman\_id int  );  SELECT \* FROM orders; |
|  |  |
|  | 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 \* FROM orders; |
|  |  |
|  | SELECT ord\_no,ord\_date,purch\_amt FROM orders WHERE salesman\_id=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 item\_mast  (  pro\_id int,  pro\_name varchar(30),  pro\_price VARCHAR(30),  pro\_com int  );  SELECT \* FROM item\_mast; |
|  |  |
|  | 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 catridge',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; |
|  |  |
| 21. | From the following table, write a SQL query to calculate the averageprice for a manufacturer code of 16. Return avg. |
|  | SELECT AVG(pro\_price) AS AVG\_Price FROM item\_mast WHERE pro\_com = 16; |
|  |  |
| 22. | From the following table, write a SQL query to display the pro\_nameas 'Item Name' and pro\_priceas 'Price in Rs.' |
|  | SELECT pro\_name AS "Item\_name",pro\_Price AS "Price in Rs." FROM item\_mast; |
|  |  |
| 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 pro\_name , pro\_price FROM item\_mast WHERE pro\_price >= 250 ORDER BY pro\_price DESC,pro\_name ASC; |
|  |  |
| 24. | .From the following table, write a SQL query to calculate average price ofthe items for each company. Return average price and companycode. |
|  | SELECT pro\_com, AVG(pro\_Price) AS AVG\_Price FROM item\_mast GROUP BY pro\_com |
|  |  |