**MODULE-5**

**01-What do you understand by databas?**

**ANS:** 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).

**02-What is Normalization?**

**ANS:**Normalization is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies. Normalization rules divides larger tables into smaller tables and links them using relationships. The purpose of Normalisation in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

**03-What is difference between DBMS and RDBMS?  
ANS-**DBMS and RDBMS? DBMS stands for Database Management System, and RDBMS is the acronym for the Relational Database Management system. In DBMS, the data is stored as a file, whereas in RDBMS, data is stored in the form of tables.

**04-What is MF Cod Rule of RDBMS Systems?**

**ANS-**E.F Codd was a Computer Scientist who invented the Relational model for Database management. Based on relational model, the Relational database was created. Codd proposed 13 rules popularly known as Codd's 12 rules to test DBMS's concept against his relational model. Codd's rule actualy define what quality a DBMS requires in order to become a Relational Database Management System(RDBMS). Till now, there is hardly any commercial product that follows all the 13 Codd's rules. Even Oracle follows only eight and half(8.5) out of 13. The Codd's 12 rules are as follows.

**Rules Zero:**This rule states that for a system to qualify as an RDBMS, it must be able to manage database entirely through the relational capabilities.

**Rule 2: Guaranted Access:**Each unique piece of data(atomic value) should be accesible by : Table Name + Primary Key(Row) + Attribute(column).

**Rule 3: Systematic treatment of NULL:**Null has several meanings, it can mean missing data, not applicable or no value. It should be handled consistently. Also, Primary key must not be null, ever. Expression on NULL must give null.

**Rule 4: Active Online Catalog:** Database dictionary(catalog) is the structure description of the complete Database and it must be stored online. The Catalog must be governed by same rules as rest of the database. The same query language should be used on catalog as used to query database.

**Rule 5: Powerful and Well-Structured Language:** One well structured language must be there to provide all manners of access to the data stored in the database. Example: SQL, etc. If the database allows access to the data without the use of this language, then that is a violation.

**Rule 6: View Updation Rule :**All the view that are theoretically updatable should be updatable by the system as well.

**Rule 7: Relational Level Operation :**There must be Insert, Delete, Update operations at each level of relations. Set operation like Union, Intersection and minus should also be supported.

**Rule 8: Physical Data Independence:**The physical storage of data should not matter to the system. If say, some file supporting table is renamed or moved from one disk to another, it should not effect the application.

**Rule 9: Logical Data Independence:** If there is change in the logical structure(table structures) of the database the user view of data should not change. Say, if a table is split into two tables, a new view should give result as the join of the two tables. This rule is most difficult to satisfy.

**Rule 10: Integrity Independence :** The database should be able to enforce its own integrity rather than using other programs. Key and Check constraints, trigger etc, should be stored in Data Dictionary. This also make RDBMS independent of front-end.

**Rule 11: Distribution Independence :** A database should work properly regardless of its distribution across a network. Even if a database is geographically distributed, with data stored in pieces, the end user should get an impression that it is stored at the same place. This lays the foundation of distributed database.

**Rule 12: Nonsubversion Rule :**If low level access is allowed to a system it should not be able to subvert or bypass integrity rules to change the data. This can be achieved by some sort of looking or encryption.

**5.What do you understand By Data Redundancy?**

**Ans:** 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. Unfortunately, data redundancy can cause data inconsistency, which can provide a company with unreliable and/or meaningless information.

**6. What is DDL Interpreter?**

DDL Interpreter DDL expands to Data Definition Language. DDL Interpreter as the name suggests interprets the DDL statements such as schema definition statements like create, delete, etc. The result of this interpretation is a set of a table that contains the meta-data which is stored in the data dictionary.

**7. What is DML Compiler in SQL?**

A data manipulation language (DML) is a computer programming language used for adding (inserting), deleting, and modifying (updating) data in a database. A DML is often a sublanguage of a broader database language such as SQL, with the DML comprising some of the operators in the language**.**

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

SQL constraints are used to specify rules for the data in a table. Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

The PRIMARY KEY constraint is a combination of both NOT NULL and UNIQUE constraints. This means that when you define a column with PRIMARY KEY, it will not accept any null or duplicate values. You will create the Student2 table in the code below and determine the ID column with the PRIMARY KEY constraint.

9. What is save Point? How to create a save Point write a Query? Ans-Savepoint is a command in SQL that is used with the rollback command.

mysql> SAVEPOINT ini;

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

A trigger is a special type of stored procedure that automatically runs when an event occurs in the database server. DML triggers run when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or view. These triggers fire when any valid event fires, whether table rows are.

**create trigger [trigger\_name]**

**[before | after]**

**{insert | update | delete}**

**on [table\_name]**

**[for each row]**

**[trigger\_body]**

**TASK:**

**1. Create table student and exam**

**Create database Assignments;**

**Create table student; //creating table//**

**Create table student(rollno int auto\_increment,name varchar[50],branch varchar[100] ) primerkey Rollno;**

**//Insert data//**

**Insert into student(name,branch) values (“jay”,”CS”);**

**Insert into student(name,branch) values (“SUHANI”,”EC”);**

**Insert into student(name,branch)values(“KRITI”,”EC”);**

**create table exam;**

**Create table student(Rollno int ,s\_code varchar,marks int,p\_code varchar[20] ) foreign key rollno references student(rollno);**

**//INSERTING DATA INTO EXAM TABLE//**

**Insert into student(s\_code, marks, p\_code) values (“CS11”,’50’,”CS”);**

**Insert into student(s\_code, marks, p\_code) values (“CS12”,’50’,”CS”);**

**values Insert into student(s\_code, marks, p\_code)value(“EC102”,’50’,”CS”);**

**Insert into student(s\_code, marks, p\_code) values (“EC101”,’50’,”CS”);**

**Insert into student(s\_code, marks, p\_code) values (“EC102”,’50’,”CS”);**

**2.create table given below.**

* **Create table personal\_Detail;**

**Create table personal\_Detaiil**

**(First\_Name varchar[50],**

**Last\_Name varchar[50],**

**Address varchar[100],**

**city varchar[50],age int);**

**//inserting data into personal\_Detail Table//**

**Insert into student(First\_Name, Last\_Name,Address,age) values (“Mickey”,”Mouse”,  
“123 fantasy way”,”anaheim”,73);**

**Insert into student(First\_Name, Last\_Name,Address,age) values (“bat”,”Man”,  
“321 cavern ave”,”gotham”,54);**

**Insert into student(First\_Name, Last\_Name,Address,age) values (“Wonder”,”Women”,  
“987 truth way”,”paradise”,39);**

**Insert into student(First\_Name, Last\_Name,Address,age) values (“Donald”,”Duck”,  
“555 quarck street”,”mallard”,65);**

**Insert into student(First\_Name, Last\_Name,Address,age) values (“Bugs”,”Buny”,  
“567 carrot street”,”rascal”,58);**

**Insert into student(First\_Name, Last\_Name,Address,age) values (“wiley”,”coyote”,  
“999 acme way”,”canyon”,61);**

**3.create table given below:** **Employee and Incentive**

* **TABLE NAME:Employee**

**Create table employee;**

**Create table employee(Emp\_Id int auto\_increament,First\_name varchar[50],Last\_Name varchar[50],salary float,join\_date datetime,Department varchar[50] ) primary key (Emp\_Id);**

**//insering data into employee table//**

**Insert into student(First\_Name, Last\_Name,salary,join\_date,Department) values (“john”,”abraham”,1000000,”2013-01-01 12:00:00””banking”);**

**Insert into student(First\_Name, Last\_Name,salary,join\_date,Department) values (“Michael”,clarke”,800000,”2013-01-01 12:00:00”,”Insurance”);**

**Insert into student(First\_Name, Last\_Name,salary,join\_date,Department) values (“Roy”,”thomas”,700000,”2013-02-01 12:00:00”,”banking”);**

**Insert into student(First\_Name, Last\_Name,salary,join\_date,Department) values (“tom”,”jose”,600000,”2013-01-01 12:00:00”Insurance”);**

**Insert into student(First\_Name, Last\_Name,salary,join\_date,Department) values (“jerry”,”pinto”,650000,”2013-01-01 12:00:00”,” Insurance”);**

**Insert into student(First\_Name, Last\_Name,salary,join\_date,Department) values (“philip”,”mathew”,750000,”2013-01-01 12:00:00”,”Services”);**

**Insert into student(First\_Name, Last\_Name,salary,join\_date,Department) values (“testname1”,”123”,650000,”2013-01-01 12:00:00”,”Services”);**

**Insert into student(First\_Name, Last\_Name,salary,join\_date,Department) values (“testname2”,”Lname%”,600000,”2013-01-01 12:00:00”,”Services”);**

* **TABLE NAME :** **Incentive**

**Create table Incentive(emp\_ref\_id int,Incentive\_Date,Incentive int)foreignkey emp\_ref\_id references employee (Emp\_Id);**

**(A)Get First\_Name from employee table using Tom name “Employee Name”**

**Query:**

**select Frist\_Name from table employee where First\_Name =”TOM”;**

**(b) Get FIRST\_NAME, Joining Date, and Salary from employee table.**

**Query:**

**Select Frist\_Name, join\_date,** **Salary from Employee;**

**(C) Get all employee details from the employee table order by First\_Name Ascending and Salary descending?**

**QUERY:**

**Select \* from employee order by First\_Name,salary desc;**

**d) Get employee details from employee table whose first name contains ‘J’.**

**QUERY:**

**Select \* from employee where First\_Name like “%j”;**

**e) Get department wise maximum salary from employee table order by salary ascending?**

**QUERY:**

**Group By**

**Select max(salary) from employee group by department,order by salary;**

**f) Select first\_name, incentive amount from employee and incentives table for those employees who have incentives and incentive amount greater than 3000**

**QUERY:**

**Select First\_name, Incentive\_amount from employee,incentive where count(Incentive\_amount)>3000 where group by incentive;**

**4.** **Create table given below: Salesperson and Customer.**

**TABLE 1: SALESPERSON**

**Create table salesperson(SNO int,SNAME varchar[50],CITY varchar[50],comm float) primarykey SNO;**

**Insert into salesperson(SNO, SNAME, CITY,comm) values (1001,”peel”,”london”,.12);**

**Insert into salesperson(SNO, SNAME, CITY,comm) values (1002,”serres”,”san jose”,.13);**

**Insert into salesperson(SNO, SNAME, CITY,comm) values (1004,”motika”,”london”,.11);**

**Insert into salesperson(SNO, SNAME, CITY,comm) values (1007,”rafkin”,”barcelona”,.15);**

**Insert into salesperson(SNO, SNAME, CITY,comm) values (1003,”axelrod”,”new york”,.1);**

**TABLE 2: CUSTOMER**

**Create table customer(CNM int,CNAME varchar[50],city varchar[20],RATING int,SNO)primarykey CNM foreignkey SNO references salesperson (SNO);**

**Insert into customer(CNM, CNAME, city, RATING) values(201,”Hoffman”,”london”,100);**

**Insert into customer(CNM, CNAME, city, RATING) values(202,”giovanne”,”roe”,200);**

**Insert into customer(CNM, CNAME, city, RATING) values(203,”Liu”,”san jose”300,);**

**Insert into customer(CNM, CNAME, city, RATING) values(204,”Grass”,”Bracelona”,100);**

**Insert into customer(CNM, CNAME, city, RATING) values(206,”clemens”,”London”,300);**

**Insert into customer(CNM, CNAME, city, RATING) values(207,”pereira”,”roe”,100);**

**Retrieve the below data from above table**

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

**Select SNAME,city from salesperson where city=”london”, group by salesperson having count(comm)>0.12;**

1. **All salespeople either in Barcelona or in London.**

**SELECT \* FROM salesperson where city=” Barcelona” or city=” London”;**

**D)** **All salespeople with commission between 0.10 and 0.12. (Boundary values**

**should be excluded).**

**SELECT \* from salesperson where comm=>0.10 AND comm<=0.12;**

**E)** **All customers excluding those with rating <= 100 unless they are located in**

**Rome**

**(NOTE FROM STUDENT-ROME city not in customertable)**

**Select \* from customer where rating<=100 or city=Rome;**