MODULE: 5 (Database)

1.What do you understand By Database What is Normalization?

Database: A database is a structured collection of data that is organized in a way to facilitate efficient retrieval, storage, and management. It is designed to support the storage, retrieval, and manipulation of data for various purposes. Databases can be simple, like a single file containing a list of contacts, or complex, like a large relational database management system (RDBMS) handling massive amounts of data in different tables.

Databases typically consist of tables, where each table holds data organized in rows and columns. Relationships between tables are established using keys. Databases also include a system for managing and querying data, often using a query language such as SQL (Structured Query Language).

Normalization: Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. The goal of normalization is to structure the database in a way that minimizes data duplication and dependency, ensuring that the data is stored efficiently and avoids certain types of anomalies.

There are several normal forms (1NF, 2NF, 3NF, BCNF, etc.), each building on the previous one, and they define specific rules for organizing data. The process of normalization involves breaking down tables into smaller, related tables, ensuring that each table serves a specific purpose and avoids unnecessary duplication of information.

Here are some key concepts associated with normalization:

1. **First Normal Form (1NF):** Ensures that each column contains atomic (indivisible) values, and there are no repeating groups or arrays.
2. **Second Normal Form (2NF):** Builds on 1NF and eliminates partial dependencies by ensuring that non-key attributes are fully functionally dependent on the entire primary key.

3. **Third Normal Form (3NF):** Builds on 2NF and removes transitive dependencies, ensuring that non-key attributes are not dependent on other non-key attributes.
4. **Boyce-Codd Normal Form (BCNF):** A stricter form of 3NF that eliminates certain types of anomalies related to functional dependencies.

2. What is Difference between DBMS and RDBMS?

Feature	DBMS	RDBMS
Data Structure	Can manage data in various formats, such as hierarchical, network, and relational models.	Specifically manages data in a tabular format with rows and columns, adhering to the relational model.
Data Relationships	May or may not support relationships between data elements.	Enforces relationships between tables using keys (e.g., primary keys, foreign keys) to maintain data integrity.
Schema	Does not necessarily require a predefined schema. Data can be more flexible in terms of structure.	Requires a predefined schema that defines the structure of tables, columns, data types, and relationships.
Query Language	May have its own query languages, which can vary widely.	Uses SQL (Structured Query Language) as the standard query language for data retrieval, manipulation, and management.

Feature	DBMS	RDBMS
ACID Properties	May or may not fully support ACID properties (Atomicity, Consistency, Isolation, Durability).	Specifically designed to adhere to ACID properties, ensuring data consistency and reliability.
Examples	Examples include MongoDB, Microsoft Access, and SQLite.	Examples include Oracle, MySQL, PostgreSQL, Microsoft SQL Server, and IBM Db2.

3. 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.

4.What do you understand By Data Redundancy?

• In DBMS, when the same data is stored in different tables, it causes data redundancy.

• Sometimes, it is done on purpose for recovery or backup of data, faster access of data, or updating data easily. Redundant data costs extra money, demands higher storage capacity, and requires extra effort to keep all the files up to date.

• Data redundancy means the occurrence of duplicate copies of similar data. It is done intentionally to keep the same piece of data at different places, or it occurs accidentally.

• In DBMS, when the same data is stored in different tables, it causes data redundancy.

- Sometimes, it is done on purpose for recovery or backup of data, faster access of data, or updating data easily.
- Redundant data costs extra money, demands higher storage capacity, and requires extra effort to keep all the files up to date.

5. What is DML Compiler in SQL?

The majority of SQL statements are categorised as DML (Data Manipulation Language), which includes SQL commands that deal with modifying data in a database. It's the section of the SQL statement that controls who has access to the database and data. DML statements and DCL statements are grouped together. Because the DML command isn't auto-committed, it won't be able to save all database changes permanently. There's a chance they'll be rolled back

INSERT INTO Command

This command can be used to insert data into a row of a table. INSERT INTO would insert the values that are mentioned in the 'Student' table below.

Syntax:

INSERT INTO NAME_OF_TABLE (1_column, 2_column, 3_column, …. N_column)

VALUES (1_value, 2_value, 3_value, …. N_value);

Or

INSERT INTO NAME_OF_TABLE

VALUES (1_value, 2_value, 3_value, …. N_value);

UPDATE Command

This statement in SQL is used to update the data that is present in an existing table of a database. The UPDATE statement can be used to update single or multiple columns on the basis of our specific needs.

Syntax:

UPDATE name_of_table SET 1_coumn = 1_value, 2_coumn = 2_value, 3_coumn = 3_value, … , N_coumn = N_value

WHERE condition;

And here,

name_of_table: name of the table

1_column, 2_column, 3_column, …. N_column: name of the first, second, third, …. nth column.

1_value, 2_value, 3_value, …. N_value: the new value for the first, second, third, …. nth column.

condition: the condition used to select those rows for which the column values need to be updated.

DELETE Command

The DELETE statement can be used in SQL to delete various records from a given table. On the basis of the condition that has been set in the WHERE clause, one can delete single or multiple records.

Syntax:

DELETE FROM name_of_table [WHERE condition];

6. What is 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.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

- NOT NULL - Ensures that a column cannot have a NULL value
- UNIQUE - Ensures that all values in a column are different
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- FOREIGN KEY - Prevents actions that would destroy links between tables
- CHECK - Ensures that the values in a column satisfies a specific condition
- DEFAULT - Sets a default value for a column if no value is specified
- CREATE INDEX - Used to create and retrieve data from the database very quickly

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

savepoint is a point within a transaction to which you can later roll back if needed. Savepoints allow you to set intermediate points in a transaction where you can save the current state of the transaction and later roll back to that specific point without affecting the entire transaction.

```
-- Start a transaction
BEGIN;

-- Insert some data
INSERT INTO employees (employee_id, first_name, last_name) VALUES (1, 'John', 'Doe');
INSERT INTO employees (employee_id, first_name, last_name) VALUES (2, 'Jane', 'Smith');

-- Create a savepoint
SAVEPOINT my_savepoint;

-- Continue with the transaction
```

```
UPDATE employees SET last_name = 'Johnson' WHERE employee_id = 1;

-- Check the data at this point
SELECT * FROM employees;

-- If something goes wrong, roll back to the savepoint
ROLLBACK TO my_savepoint;

-- Check the data after rolling back
SELECT * FROM employees;

-- Commit the transaction
COMMIT;
```

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

a trigger is a set of instructions or a program that is automatically executed ("triggered") in response to a specific event on a particular table or view. Triggers are used to enforce business rules, maintain data integrity, and automate certain actions within the database. SQL triggers can be set to execute either before or after the triggering event (e.g., INSERT, UPDATE, DELETE) occurs.

```
CREATE [OR REPLACE] TRIGGER trigger_name
{BEFORE | AFTER} {INSERT | UPDATE | DELETE}
ON table_name
[REFERENCING {OLD AS old | NEW AS new}]
FOR EACH ROW
[WHEN (condition)]
BEGIN
   -- Trigger action statements
END;
```

TASK:1

Q1.Create Table Name : Student and Exam.

- CREATE TABLE student(Rollno int auto_increment not null primary key,Name varchar(15) not null,Branch varchar(20) not null);

- INSERT INTO student (name,branch) VALUES("jay","Computer science"),("suhani","Electronics and Com."),("Kirti","Electronics and Com");

- CREATE TABLE exam(Rollno int not null, foreign key(Rollno) references student(Rollno),scode varchar(20),marks int(100),pcode varchar(20));

- INSERT INTO exam(Rollno,scode,marks,pcode) VALUES(1,"cs11",50,"cs"),(1,"cs12",60,"cs"),(2,"ec101",66,"ec"),(2,"ec102",70,"ec"),(3,"ec101",45,"ec"),(3,"ec102",50,"ec");

Q2. Create table given below

CREATE TABLE emp(FirstName varchar(20) not null,LastName varchar(20) not null,Address varchar(25) not null,City varchar(15) not null,Age int(5) not null);

INSERT INTO emp(FirstName,LastName,Address,City,Age) VALUES
("Mickey","Mouse","123 Fantasy Way","Anaheim",73),
("Bat","Man","321 Cavern Ave","Gotham",54),
("Wonder","Woman","987 Truth Way","Paradise",39),
("Donald","Duck","555 Quack Street","Mallard",56),
("Bugs","Bunny","567 Carrot Street","Rascal",58),
("Wiley","Coyote","999 Acme Way","Canyon",61),
("Cat","Woman","234 Purrfect Street","Hairball",32),
("Tweety","Bird","543 Itotlow",28);

Q3. Create a table Employee and Incentive.

```
CREATE TABLE Employee(Employee_id int auto_increment not null primary key,First_name varchar(15) not null,Last_name varchar(15) not null, Salary int(10) not null,Joining_date datetime not null,Department varchar(20) not null);

INSERT INTO Employee(First_name,Last_name,Salary,Joining_date,Department) VALUES
('John','Abraham',10000000,'2013-01-01 12:00:00','Banking'),
('Michael','Clarke',800000,'2013-01-01 12:00:00','Insurance'),
('Roy','Thomes',700000,'2013-02-01 12:00:00','Banking'),
('Tom','Jose',600000,'2013-02-01 12:00:00','Insurance'),
('Jerry','Pinto',650000,'2013-02-01 12:00:00','Insurance'),
('Philip','Mathew',750000,'2013-01-01 12:00:00','Service'),
('TestName1','123',650000,'2013-01-01 12:00:00','Service'),
('TestName2','Lname%',600000,'13-02-01 12:00:00','Insurance');

CREATE TABLE Incentive(Employee_ref_id int not null,Incentive_date date not null,Incentive_amount int(10) not null);

INSERT INTO Incentive (Employee_ref_id,Incentive_date,Incentive_amount) VALUES
(1,'2013-02-01',5000),
(2,'2013-02-01',3000),
(3,'2013-02-01',4000),
(1,'2013-01-01',4500),
(2,'2013-01-01',3500);
```

A. Get First_Name from the employee table using Tom name "Employee Name".

```
SELECT First_name FROM Employee WHERE First_name = 'Tom';
```

B. Get FIRST_NAME, Joining Date, and Salary from the employee table.

SELECT First_name, Joining_date, Salary FROM Employee;

C. Get all employee details from the employee table order by First_Name Ascending and Salary descending?

SELECT * FROM Employee ORDER BY First_name ASC, Salary DESC;

D. Get employee details from the employee table whose first name contains 'J'.

SELECT * FROM Employee WHERE First_name LIKE '%J%';

E.Get department-wise maximum salary from the employee table order by salary ascending?

SELECT Department, MAX(Salary) AS max_salary FROM Employee GROUP BY Department ORDER BY max_salary ASC;

F. Select first_name, incentive amount from the employee and incentives table for those employees who have incentives and incentive amount greater than 3000.

```
SELECT Employee.First_name, Incentive.Incentive_amount
FROM Employee
JOIN Incentive ON Employee.Employee_id = Incentive.Employee_ref_id
WHERE Incentive.Incentive_amount > 3000;
```

G. Create an After Insert trigger on the Employee table, which inserts records into the view table.

```
CREATE TABLE viewtable (
   employee_id int auto_increment primary key,
   first_name varchar(15) not null,
   last_name varchar(15) not null,
   salary int(10) not null,
```

```
    joining_date datetime not null,
    department varchar(20)
);

CREATE TRIGGER emp_insert_trigger AFTER INSERT ON Employee
FOR EACH ROW
INSERT INTO viewtable (first_name, last_name, salary, joining_date, department)
VALUES ('John', 'Abraham', 10000000, '2013-01-01 12:00:00', 'Banking');
```

Q4. Create table given below: Salesperson and Customer.

```
-- Create table SALESPERSON
CREATE TABLE SALESPERSON (
    SNO INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
    SNAME VARCHAR(15) NOT NULL,
    CITY VARCHAR(20) NOT NULL,
    COMM FLOAT(5) NOT NULL
);

-- Insert data into SALESPERSON
INSERT INTO SALESPERSON (SNO, SNAME, CITY, COMM)
VALUES
(1001, 'Peel', 'London', 0.12),
(1002, 'Serres', 'San Jose', 0.13),
(1004, 'Motika', 'London', 0.11),
(1007, 'Rafkin', 'Barcelona', 0.15),
(1003, 'Axelrod', 'New York', 0.1);

-- Create table CUSTOMER
CREATE TABLE CUSTOMER (
    CNM INT(5) PRIMARY KEY NOT NULL,
    CNAME VARCHAR(15) NOT NULL,
```

```
    CITY VARCHAR(15) NOT NULL,
    RATING INT(5) NOT NULL,
    SNO INT(10),
    FOREIGN KEY (SNO) REFERENCES SALESPERSON (SNO)
);

-- Insert data into CUSTOMER
INSERT INTO CUSTOMER (CNM, CNAME, CITY, RATING, SNO)
VALUES
(201, 'Hoffman', 'London', 100, 1001),
(202, 'Giovanni', 'Roe', 200, 1003),
(203, 'Liu', 'San Jose', 300, 1002),
(204, 'Grass', 'Barcelona', 100, 1002),
(206, 'Clements', 'London', 300, 1007),
(207, 'Pereira', 'Roe', 100, 1004);
```

A) All orders for more than $1000.

```
SELECT * FROM CUSTOMER WHERE RATING > 1000;
```

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

```
SELECT SNAME, CITY, COMM FROM SALESPERSON WHERE CITY = 'London' AND COMM > 0.12;
```

C) 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.
SELECT * FROM CUSTOMER WHERE NOT (RATING <= 100 AND CITY = 'Rome');