**PROGRAM - 1**

**SIGNUP :**

CREATE DATABASE formDB;

USE formDB;

**--CREATE TABLE**

CREATE TABLE signUp(signup\_id INT PRIMARY KEY, signup\_name VARCHAR(20), signup\_password VARCHAR(20));

--VIEW signUP TABLE

SP\_HELP signUp;

**--INSERT VALUES**

INSERT INTO signUp VALUES(1,'Anusha','Anusha123'),(2,'Sneha','Sneha123'),(3,'Arun','Arun123'),(4,'Ashik','Ashik123');

**--SELECT**

SELECT \* FROM signUp;

SELECT signup\_name FROM signUp;

**--UPDATE**

UPDATE signUp SET signup\_password = 'Anu123' WHERE signup\_id = 1;

SELECT \* FROM signUp;

**--DELETE**

DELETE FROM signUp WHERE signup\_id=3;

SELECT \* FROM signUp;

**LOGIN :**

CREATE TABLE loginTable(login\_id INT PRIMARY KEY, first\_name VARCHAR(20), last\_name VARCHAR(20), age INT, department VARCHAR(20));

SP\_HELP loginTable;

**--INSERT VALUES**

INSERT INTO loginTable VALUES(1,'Elrin','Kakkassery',23,'MCA'),(2,'Krishna','Sidh',24,'Mechanical'),(3,'Thrishna','Jaypal',25,'Electrical'),(4,'Aswathy','Krishna',27,'Computer Science');

SELECT \* FROM loginTable;

**--SELECT**

SELECT first\_name,last\_name,department FROM loginTable;

**--UPDATE**

UPDATE loginTable SET last\_name='Jayan' WHERE login\_id=3;

SELECT \* FROM loginTable;

**--DELETE**

DELETE FROM loginTable WHERE login\_id=4;

SELECT \* FROM loginTable;

**PROGRAM - 2**

CREATE DATABASE employees;

USE employees;

**--CREATE TABLE**

CREATE TABLE employee(emp\_id INT PRIMARY KEY, emp\_name VARCHAR(20), designation VARCHAR(20), salary INT);

SP\_HELP employee;

**--INSERT VALUES**

INSERT INTO employee VALUES(123,'Anusha','Manager',100000),(456,'Ajmal','Peon',80000),(789,'Arun','Clerk',65000),(101,'John','Staff',55000);

SELECT \* FROM employee;

**--SECOND HIGHEST SALARY**

SELECT MAX(salary) FROM employee WHERE salary<(SELECT MAX(salary) FROM employee);

**PROGRAM - 3**

CREATE DATABASE department\_data;

USE department\_data;

CREATE TABLE departments(dept\_id INT PRIMARY KEY, dept\_name VARCHAR(20), emp\_name VARCHAR(20), salary INT);

INSERT INTO departments VALUES(1,'MCA','Anusha',80000),(2,'ME','Arun',85000),(3,'MCA','Aathira',75000),(4,'MCA','Ashik',60000),

(5,'ME','Aparna',80000),(6,'CSE','Sanju',90000),(7,'EEE','Sriya',85000);

SELECT \* FROM departments;

SELECT dept\_name, COUNT(emp\_name) FROM departments GROUP BY(dept\_name);

**PROGRAM – 4**

CREATE DATABASE mydb;

USE mydb;

CREATE TABLE employee(emp\_id INT PRIMARY KEY, first\_name VARCHAR(20), last\_name VARCHAR(20), dept\_id VARCHAR(20));

INSERT INTO employee VALUES(1,'Anusha','Satheesh','D01'),(2,'Arun','Satheesh','D02'),(3,'Sneha','Sarasan','D03'),(4,'Ashik','Sukumar','D04');

SELECT \* FROM employee;

CREATE TABLE departments(dept\_id VARCHAR(20) PRIMARY KEY, dept\_name VARCHAR(20));

INSERT INTO departments VALUES('D01','Sales'),('D02','Marketing'),('D03','HR'),('D04','IT'),('D05','Buiseness');

SELECT \* FROM departments;

**--INNER JOIN**

SELECT e.first\_name, e.last\_name, d.dept\_id FROM employee AS e INNER JOIN departments AS d ON e.dept\_id=d.dept\_id;

**--LEFT JOIN**

SELECT e.first\_name, e.last\_name, d.dept\_name FROM employee AS e LEFT JOIN departments AS d ON e.dept\_id=d.dept\_id;

**--RIGHT JOIN**

SELECT e.first\_name, e.last\_name, d.dept\_name FROM employee AS e RIGHT JOIN departments AS d ON e.dept\_id=d.dept\_id;

**--FULL JOIN**

SELECT e.first\_name, e.last\_name, d.dept\_name FROM employee AS e FULL JOIN departments AS d ON e.dept\_id=d.dept\_id;

**--CROSS JOIN**

SELECT \* FROM employee CROSS JOIN departments;

**PROGRAM – 5**

CREATE DATABASE signup;

USE signup;

CREATE TABLE users(users\_id INT PRIMARY KEY, users\_name VARCHAR(20), users\_email VARCHAR(20), users\_password VARCHAR(20));

**--STORED PROCEDURES**

**--CREATE - ADD A NEW USER**

CREATE PROCEDURE AddNewUser

@users\_id INT,

@users\_name VARCHAR(20),

@users\_email VARCHAR(20),

@users\_password VARCHAR(20)

AS

BEGIN

INSERT INTO users (users\_id, users\_name, users\_email, users\_password)

VALUES (@users\_id, @users\_name, @users\_email, @users\_password);

END;

EXEC AddNewUser

@users\_id = 1,

@users\_name = 'Anusha',

@users\_email = 'anusha123@gmail.com',

@users\_password = 'Anusha123';

**--READ - GET USER BY ID**

CREATE PROCEDURE UserByID

@users\_id INT

AS

BEGIN

SELECT users\_id, users\_name, users\_email, users\_password

FROM users

WHERE users\_id = @users\_id;

END;

EXEC UserByID @users\_id = 1;

**--UPDATE - USERS**

CREATE PROCEDURE UpdateUser

@users\_id INT,

@users\_name VARCHAR(20),

@users\_email VARCHAR(20),

@users\_password VARCHAR(20)

AS

BEGIN

UPDATE users

SET users\_name = @users\_name,

users\_email = @users\_email,

users\_password = @users\_password

WHERE users\_id = @users\_id;

END;

EXEC UpdateUser

@users\_id = 1,

@users\_name = 'Anusha Satheesh',

@users\_email = 'anusha123@gmail.com',

@users\_password = 'Anusha123';

--DELETE - USER

CREATE PROCEDURE DeleteUsers

@users\_id INT

AS

BEGIN

DELETE FROM users

WHERE users\_id = @users\_id;

END;

EXEC DeleteUsers @users\_id = 1;

**PROGRAM – 6**

CREATE DATABASE admissionForm;

USE admissionForm;

CREATE TABLE admission(stud\_id INT PRIMARY KEY, first\_name VARCHAR(20), last\_name VARCHAR(20), dob VARCHAR(20), gender VARCHAR(10), address VARCHAR(20),

phone\_number VARCHAR(20), email VARCHAR(20));

CREATE PROCEDURE Admission\_Forms

@Operation VARCHAR(10),

@stud\_id INT = NULL,

@first\_name VARCHAR(20) = NULL,

@last\_name VARCHAR(20) = NULL,

@dob VARCHAR(20) = NULL,

@gender VARCHAR(10) = NULL,

@address VARCHAR(20) = NULL,

@phone\_number VARCHAR(20) = NULL,

@email VARCHAR(20) = NULL

AS

BEGIN

IF @Operation = 'INSERT'

BEGIN

INSERT INTO admission (stud\_id, first\_name, last\_name, dob, gender, address, phone\_number, email)

VALUES (@stud\_id, @first\_name, @last\_name, @dob, @gender, @address, @phone\_number, @email);

END

ELSE IF @Operation = 'UPDATE'

BEGIN

UPDATE admission

SET first\_name = @first\_name,

last\_name = @last\_name,

dob = @dob,

gender = @gender,

address = @address,

phone\_number = @phone\_number,

email = @email

WHERE stud\_id = @stud\_id;

END

ELSE IF @Operation = 'DELETE'

BEGIN

DELETE FROM admission

WHERE stud\_id = @stud\_id;

END

ELSE IF @Operation = 'SELECT'

BEGIN

SELECT \* FROM admission

WHERE stud\_id = @stud\_id;

END

ELSE

BEGIN

RAISERROR('Invalid operation.', 16, 1);

END

END

EXEC Admission\_Forms

@Operation = 'INSERT',

@stud\_id = 1,

@first\_name = 'Anusha',

@last\_name = 'K S',

@dob = '2000-02-08',

@gender = 'Female',

@address = 'ABC Street',

@phone\_number = '9974521846',

@email = 'anusha123@gmail.com';

EXEC Admission\_Forms

@Operation = 'SELECT',

@stud\_id = 1;

EXEC Admission\_Forms

@Operation = 'UPDATE',

@stud\_id = 1,

@first\_name = 'Anusha',

@last\_name = 'K Satheesh',

@dob = '2000-02-08',

@gender = 'Female',

@address = 'ABC Street',

@phone\_number = '9974521858',

@email = 'anusha123@gmail.com';

EXEC Admission\_Forms

@Operation = 'DELETE',

@stud\_id = 1;

**PROGRAM – 7**

CREATE DATABASE example;

USE example;

**--First Normal Form (1NF)**

CREATE TABLE customer(customer\_id INT PRIMARY KEY,customer\_name VARCHAR(50), addresses VARCHAR(200));

**-- This table is not in 1NF because the Addresses column contains multiple values. To normalize it, split the Addresses column into separate columns**

CREATE TABLE customers(customers\_id INT PRIMARY KEY,customers\_name VARCHAR(50));

CREATE TABLE customerAddress(customers\_id INT,address VARCHAR(100),FOREIGN KEY (customers\_id) REFERENCES customers(customers\_id));

SP\_HELP customers;

SP\_HELP customerAddress;

**--Second Normal Form (2NF)**

CREATE TABLE Orders(OrderID INT PRIMARY KEY,CustomerID INT,OrderDate DATE,Items VARCHAR(200));

**--This table is not in 2NF because the Items column is dependent on the OrderID and CustomerID composite primary key. To normalize it, create a separate table for order items**

CREATE TABLE Orders\_item (OrderID INT PRIMARY KEY,CustomerID INT,OrderDate DATE);

CREATE TABLE OrderItem (OrderID INT,Item VARCHAR(100),FOREIGN KEY (OrderID) REFERENCES Orders\_item(OrderID));

SP\_HELP Orders\_item;

SP\_HELP OrderItem;

**--Third Normal Form (3NF)**

CREATE TABLE Products (ProductID INT PRIMARY KEY,ProductName VARCHAR(50),Price DECIMAL(10,2));

CREATE TABLE OrderItem (OrderID INT,ProductID INT,Quantity INT,FOREIGN KEY (OrderID) REFERENCES Order(OrderID),FOREIGN KEY (ProductID) REFERENCES Product(ProductID));

SP\_HELP Products;

SP\_HELP OrderItem;

**--This table is in 3NF because there are no transitive dependencies.**

**-- Clustered Index**

CREATE TABLE Customer (CustomerID INT PRIMARY KEY CLUSTERED,CustomerName VARCHAR(50),Address VARCHAR(100));

**--A clustered index physically orders the rows in the table based on the indexed column (in this case, CustomerID).**

**--Non-Clustered Index**

CREATE NONCLUSTERED INDEX IX\_CustomerName ON Customer (CustomerName);

**-- A non-clustered index creates a separate structure that contains pointers to the actual rows in the table, allowing for efficient searching based on the indexed column (CustomerName).**

**--Pivot**

CREATE TABLE Sales (Products VARCHAR(50),Months VARCHAR(20),Amount INT);

INSERT INTO Sales VALUES('Product A', 'January', 100),('Product A', 'February', 150),('Product B', 'January', 200),('Product B', 'February', 250);

SELECT Products,SUM(CASE WHEN Month = 'January' THEN Amount ELSE 0 END) AS January,SUM(CASE WHEN Month = 'February' THEN Amount ELSE 0 END) AS February

FROM Sales GROUP BY Products;

**--The CASE statement checks each row to see if the month is 'January' or 'February'.**

**--If it is, it adds the Amount to the corresponding column. Otherwise, it adds 0.**

**--The SUM function is used to get the total for each product per month.**

**--GROUP BY Product groups the results by product.**

**--UnPivot**

CREATE TABLE PivotedSales (Products VARCHAR(50),January INT,February INT);

INSERT INTO PivotedSales VALUES ('Product A', 100, 150),('Product B', 200, 250);

SELECT Products, Months, Amount FROM PivotedSales UNPIVOT (Amount FOR Month IN (January, February)) AS Unpivoted;

**--The UNPIVOT operation takes the column names (January and February) and turns them into row values under the Month column.**

**--Amount FOR Month IN (January, February) specifies that Amount is the value column and Month is the name column for the unpivoted values.**

**--Merge**

CREATE DATABASE merge\_example;

USE merge\_example;

CREATE TABLE products(product\_id INT PRIMARY KEY,product\_name VARCHAR(50),price DECIMAL(10, 2));

INSERT INTO products VALUES (1, 'Apple', 0.50),(2, 'Banana', 0.30),(3, 'Orange', 0.80);

SELECT \* FROM products;

CREATE TABLE updated\_products (product\_id INT PRIMARY KEY,product\_name VARCHAR(50),price DECIMAL(10, 2));

INSERT INTO updated\_products VALUES (2, 'Banana', 0.35), (3, 'Orange', 0.75), (4, 'Grapes', 1.20);

SELECT \* FROM updated\_products;

MERGE INTO products AS target USING updated\_products AS source ON target.product\_id = source.product\_id WHEN MATCHED THEN UPDATE SET

target.product\_name = source.product\_name , target.price = source.price WHEN NOT MATCHED THEN INSERT(product\_id, product\_name, price)

VALUES (source.product\_id, source.product\_name, source.price);

SELECT \* FROM products;

**PRACTISE**

CREATE DATABASE tasks;

USE tasks;

CREATE TABLE employee(emp\_id VARCHAR(20) PRIMARY KEY, emp\_name VARCHAR(20), dob DATE, emp\_address VARCHAR(20), mobile\_no VARCHAR(20), dept\_id VARCHAR(20), salary INT);

CREATE TABLE department(dept\_id VARCHAR(20) PRIMARY KEY, dept\_name VARCHAR(20), dept\_location VARCHAR(20));

SP\_HELP employee;

SP\_HELP department;

INSERT INTO employee VALUES('E01','Anusha','2000-02-08','ABC','9914521786','D01',100000),('E02','Ajmal','1999-04-06','DEF','9914532786','D02',200000),

('E03','Aparna','2001-01-25','GHI','9914521748','D03',80000),('E04','Ajay','2002-08-02','JKL','9972521786','D04',70000);

SELECT \* FROM employee;

INSERT INTO department VALUES('D01','MCA','North Block'),('D02','ME','North Block'),('D03','EEE','South Block'),('D04','CSE','South Block');

SELECT \* FROM department;

ALTER TABLE department DROP COLUMN dept\_location;

SELECT \* FROM department;

ALTER TABLE employee ADD designation VARCHAR(20);

SELECT \* FROM employee;

ALTER TABLE employee ALTER COLUMN mobile\_no INT;

SELECT \* FROM employee;

SELECT emp\_name , dept\_id , salary FROM employee ORDER BY salary ASC;

SELECT \* FROM employee WHERE emp\_name LIKE '%a';

SELECT \* FROM employee WHERE salary BETWEEN 80000 AND 200000;

UPDATE employee SET designation = 'Manager' WHERE emp\_id='E01';

UPDATE employee SET designation = 'Peon' WHERE emp\_id='E02';

UPDATE employee SET designation = 'Clerk' WHERE emp\_id='E03';

UPDATE employee SET designation = 'Manager' WHERE emp\_id='E04';

SELECT \* FROM employee;

SELECT DISTINCT (designation) FROM employee;

DELETE FROM employee WHERE emp\_id = 'E04';

SELECT \* FROM employee;

SELECT emp\_id , emp\_name , salary FROM employee WHERE designation='Manager' OR designation='Clerk';

SELECT COUNT(emp\_name),designation FROM employee GROUP BY (designation);