**2.To create a database and signup and login tables with all the field requirements ,and then implement insert,select,update,and delete operations by using sql queries.**

Create database Project

Use Project

Signup

Create table signup(UserId int primary key,Username varchar(20),DOB date,Email varchar(20),Gender varchar(5))

Insert into Signup(100,’Dev’,’2000-11-11’,’dev@gmail.com’,’M’)

Insert into Signup(101,’Krishna’,’2002-11-11’,’krishna@gmail.com’,’F’)

Update Signup set Username=’Hari’ where UserId=100

Delete table signup

Login

Create table login(Username int primary key,Email varchar(20))

Insert into login(‘Deepak’,’deepak@gmail.com’)

Insert into login(‘Siva’,’siva@gmail.com’)

Update login set Username=’Isha’ where Username=’Siva’

Delete table login;

**3.To create an employee table and how to achieve or get the second highest salary from the table**

Create table Employee(

EmpId int primary key,

Empname varchar(50),

Salary int

)

insert into Employee (EmpId,Empname,Salary,Department)values(6,'Sree',3000,'IT');

select \* from Employee;

SELECT MAX(Salary) FROM Employee;

ALTER TABLE Employee ADD Department varchar(40);

UPDATE Employee SET Department = 'Marketing' WHERE EmpId = 3

SELECT MAX(Salary) FROM Employee

WHERE Salary < (SELECT MAX(Salary) FROM Employee);

**4.Perform the sql query to list the number of employee in each dpt.**

SELECT COUNT(EmpId) FROM Employee WHERE Department='IT';

**5.To create two tables and implement the SQL join concepts**

create table Student( StudId int primary key,Studname varchar(30),Course varchar(20));

create table Department(DeptId int primary key,Dptname varchar(30),College varchar(20));

insert into Student(StudId,Studname,Course) values(3,'Bhavya','MSc')

insert into Department(DeptId,Dptname,College) values(3,'MSc','Kerala University')

select \*from Student;

select \*from Department;

alter table Student add DeptId int;

update Student set DeptId= 101 where StudId=1;

update Department set DeptId= 103 where Dptname='MSc';

//inner join

select S.StudId,S.Studname,D.Dptname from Student S INNER JOIN Department D ON S.DeptId=D.DeptId;

//left join

select Studname from Student S LEFT JOIN Department D ON S.DeptId=D.DeptId;

//right join

select Dptname from Department D RIGHT JOIN Student S ON S.DeptId=D.DeptId;

//full outer join

select College from Department D FULL OUTER JOIN Student S ON S.DeptId=D.DeptId;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

alter table Department add Salary int;

Update Department set Salary =1000 where DeptId=103

select \* from Department

SELECT D.Dptname, D.College, D.Salary FROM Department D INNER JOIN Student S ON S.DeptId = D.DeptId;

SELECT D.Dptname, D.College FROM Department D LEFT JOIN Student S ON S.DeptId = D.DeptId;

SELECT D.Dptname, D.College FROM Department D RIGHT JOIN Student S ON S.DeptId = D.DeptId;

**6.To create different stored procedures to implement CRUD operations on Register page**

create table Register(UserId int primary key,Username varchar(30),Email varchar(20),DOB date,Gender varchar (4))

insert into Register(UserId,Username,Email,DOB ,Gender) values(102,'Deepu','deepu@gmail.com','2006-03-10','M')

select \*from Register;

**CREATE**

CREATE PROCEDURE Registerdetails

-- Add the parameters for the stored procedure here

@UserId int,

@Username varchar(20),

@Email varchar(20),

@DOB date,

@Gender varchar(5)

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

INSERT INTO Register(

UserId,Username,Email,DOB,Gender)

values(

@UserId,@Username,@Email,@DOB,@Gender

)

SET @UserId = SCOPE\_IDENTITY()

END

GO

**READ**

CREATE PROC Registerdetails\_read

@UserId int

AS

BEGIN

SELECT UserId,Username,Email,DOB,Gender

FROM Register

WHERE (UserId=@UserId)

END

GO

**UPDATE**

CREATE PROC Registerdetails\_update

@UserId int,

@Username varchar(20),

@Email varchar(20),

@DOB date,

@Gender varchar(5)

AS

BEGIN

UPDATE Register

SET UserId=@UserId,

Username=@Username,

Email=@Email,

DOB=@DOB,

Gender=@Gender where UserId=@UserId

END

GO

**DELETE**

CREATE PROC Registerdetails\_delete

@UserId int

AS

BEGIN

DELETE FROM Register WHERE UserId=@UserId

END

GO

\*\*\*\*\*\*\*\*\*\*\*\*\*\*select

CREATE PROC Registerdetails\_select

AS

BEGIN

SELECT \* from Register

END

GO

//-------------------------

EXECUTE Registerdetails

@UserId=1,

@Username='Joe',

@Email='Joe@gmail.com',

@DOB='2000-03-05',

@Gender='M'

EXECUTE Registerdetails\_read @UserId=1

EXECUTE Registerdetails\_update @UserId=1,@Username='Dev',

@Email='dev@gmail.com',@DOB='1989-03-04',@Gender='M'

/////

**7.To create single stored procedure for implementing CRUD operations on Student admission page**

CREATE PROCEDURE [dbo].[Studentadmission]

@Operation VARCHAR(10),

@StudID INT = NULL,

@StudName NVARCHAR(50) = NULL,

@BirthDate DATE = NULL,

@Email NVARCHAR(100) = NULL,

@Gender VARCHAR(10) = NULL

AS

BEGIN

IF @Operation = 'CREATE'

BEGIN

CREATE TABLE Students(StudId int,Studname varchar(20),BirthDate date,Email varchar(30),Gender varchar(5))

END

IF @Operation = 'INSERT'

BEGIN

INSERT INTO Students(StudId, StudName, BirthDate, Email, Gender)

VALUES (@StudID, @StudName, @BirthDate, @Email, @Gender);

END

IF @Operation = 'UPDATE'

BEGIN

UPDATE Students SET

Studname = @StudName,

BirthDate = @BirthDate,

Email = @Email,

Gender = @Gender

WHERE StudId = @StudID;

END

IF @Operation = 'DELETE'

BEGIN

DELETE FROM Students WHERE StudId = @StudId;

END

IF @Operation = 'SELECT'

BEGIN

SELECT StudId, StudName, Email, BirthDate, Gender

FROM Students

WHERE StudId = @StudId;

END

END

EXEC Studentadmission 'CREATE';

EXEC Studentadmission 'INSERT', 100, 'Fahad', '2004-11-11', 'fahad@gmail.com', 'M';

EXEC Studentadmission 'UPDATE', 100, 'Dhruv', '2003-10-23', 'Dhruv@gmail.com', 'M';

EXEC Studentadmission 'SELECT', 100;

EXEC Studentadmission 'DELETE', 100;

**8(a)Normalisation**

CREATE TABLE Studentlist (

StudentId INT PRIMARY KEY,

Studname VARCHAR(20)

);

INSERT INTO Studentlist (StudentId, Studname) VALUES (1, 'Raghav');

INSERT INTO Studentlist (StudentId, Studname) VALUES (2, 'Dhruv');

Create table Courses(StudentId int,Courses varchar(30))

INSERT INTO Courses (StudentID, Courses)VALUES (1, 'Math');

INSERT INTO Courses (StudentID, Courses)VALUES (1, 'Physics');

INSERT INTO Courses (StudentID, Courses)VALUES (2, 'Chemistry');

select \* from Studentlist;

select \* from Courses;

///////////////////////////////

**8(b) CLUSTERED INDEX**

Create database Student;

Use Student

Create table studentform(Id int primary key,

name varchar(50) not null,

gender varchar(30) not null,

DOB date not null,

city varchar(30)not null)

Insert into studentform values

(3,'Ram','M','1999-01-02','Europe'),

(1,'Rudra','F','1998-04-11','Kochi'),

(2,'Sreyas','M','2008-05-10','Delhi')

select \* from studentform

CREATE CLUSTERED INDEX IX\_studentform\_Id ON studentform(Id);

select \* from studentform;

NON CLUSTERED INDEX

CREATE NONCLUSTERED INDEX IX\_studentform\_name ON studentform(name);

CREATE NONCLUSTERED INDEX IX\_studentform\_city ON studentform(city);

///////////////////////////////

**8(C) PIVOT AND UNPIVOT VALUES**

Query: SELECT (ColumnNames)

FROM (TableName)

PIVOT

(

AggregateFunction(ColumnToBeAggregated)

FOR PivotColumn IN (PivotColumnValues)

) AS (Alias) //Alias is a temporary name for a table

USE Student;

Create table masterstudents(

Coursename varchar(40),

CourseCategory varchar(50),

Coursefee int

)

Insert into masterstudents values('C','Programming',30000)

Insert into masterstudents values('C++','Programming',24000)

select \* from masterstudents

SELECT Coursename, [Programming]

FROM

(

SELECT Coursename, CourseCategory, Coursefee

FROM masterstudents

) AS SourceTable

PIVOT

(

SUM(Coursefee)

FOR CourseCategory IN ([Programming])

) AS PivotTable;

\*\*\*\*\*\*\*\*Unpivot

Query: SELECT (ColumnNames)

FROM (TableName)

UNPIVOT

(

AggregateFunction(ColumnToBeAggregated)

FOR PivotColumn IN (PivotColumnValues)

) AS (Alias)

SELECT Coursename, CourseCategory, Price

FROM

(

SELECT Coursename, PROGRAMMING FROM masterstudents

PIVOT

( SUM(Coursefee)

FOR CourseCategory IN (PROGRAMMING)

) AS PivotTable

) P

UNPIVOT

(

Coursefee FOR CourseCategory IN (PROGRAMMING) )

AS UnpivotTable

///////////////////////////////////////////

**8(d) Merge concepts**

CREATE TABLE mainProducts(

ProductID INT,

ProductName VARCHAR(50),

Price int

)

GO

INSERT INTO mainProducts(ProductID,ProductName, Price) VALUES(1,'Table',100)

INSERT INTO mainProducts(ProductID,ProductName, Price) VALUES(2,'Desk',80)

INSERT INTO mainProducts(ProductID,ProductName, Price) VALUES(3,'Chair',50)

INSERT INTO mainProducts(ProductID,ProductName, Price) VALUES(4,'Computer',300)

GO

CREATE TABLE SubProducts(

ProductID INT,

ProductName VARCHAR(50),

Price int

)

GO

INSERT INTO SubProducts(ProductID,ProductName, Price) VALUES(1,'Table',100)

INSERT INTO SubProducts(ProductID,ProductName, Price) VALUES(2,'Desk',180)

INSERT INTO SubProducts(ProductID,ProductName, Price) VALUES(5,'Bed',50)

INSERT INTO SubProducts(ProductID,ProductName, Price) VALUES(6,'Cupboard',300)

GO

SELECT \* FROM mainProducts

SELECT \* FROM SubProducts

CREATE TABLE MergedProducts (

ProductID INT,

ProductName VARCHAR(50),

Price INT

);

INSERT INTO MergedProducts (ProductID, ProductName, Price)

SELECT ProductID, ProductName, Price

FROM mainProducts;

Select \* from MergedProducts;

INSERT INTO MergedProducts (ProductID, ProductName, Price)

SELECT sp.ProductID, sp.ProductName, sp.Price

FROM SubProducts sp

WHERE NOT EXISTS (

SELECT 1

FROM MergedProducts mp

WHERE mp.ProductID = sp.ProductID

);

SELECT \* FROM MergedProducts;