**Assignment - SQL Server Lab**

**Data manipulation operations (DDL, DM L, TCL)**

—-Assignment 1: Create a Database and Table

—-Task:

–1. Create a database named CompanyDB.

—-2. Create a table named Employees with the following columns:

—-EmployeeID (Primary Key, INT, Identity)

—-FirstName (VARCHAR(50))

—-LastName (VARCHAR(50))

—-Department (VARCHAR(50))

—-Salary (DECIMAL(10, 2))

use CompanyDB

create table Employee(

EmployeeID int NOT NULL PRIMARY KEY IDENTITY(1,1),

FirstName varchar(50),

LastName varchar(50),

Department varchar(50),

Salary decimal(10,2)

);

Assignment 2: Insert Data into a Table

insert into Employee(FirstName, LastName, Department, Salary)

values

('John', 'Doe', 'HR', 50000),

('Jane', 'Smith', 'IT', 60000),

('David', 'Lee', 'Marketing', 55000)

-- Dropping the Employee table/Deleting

drop table Employee;

Assignment 3: Select and Query Data

--Retrieve the FirstName, LastName, and Salary of employees in the IT department and

--salary greater than 5000.

select FirstName,LastName,Salary from Employee

select \*from Employee where Department='IT' and Salary>5000;

--Find the highest salary from the Employees table.

select Max(Salary) as MaxSalary From Employee

Assignment 4: Update Data in a Table

--Update the salary of John Doe to 55000.

Update Employee

set Salary = 55000

where FirstName = 'John'

--Change the department of Jane Smith to Marketing.

Update Employee

set Department = 'Marketing'

where FirstName = 'Jane'

Assignment 5: Delete Data from a Table

--Delete the record of David Lee from the Employees table.

delete from Employee

where FirstName ='David'

--Delete all employees in the Marketing department.

delete from Employee

where Department = 'Marketing'

Assignment 6: Product Management

Task:

1. Create a table named Products and insert below records in the table.

create database ProductDB

use ProductDB

create table Products(

ProductID int NOT NULL PRIMARY KEY IDENTITY(1,1),

ProductName varchar(50),

Category varchar(50),

Price decimal(10,2),

Stock decimal(10,2)

);

insert into Products(ProductName, Category, Price, Stock)

values

('Laptop', 'Electronics', 1000, 50),

('Mouse', 'Electronics', 20, 200),

('Keyboard', 'Electronics', 30 ,150),

('Printer', 'Electronics', 150, 75)

select \* from Products

--2. Select all columns from the Products table

select ProductID,ProductName,Category,Price,Stock from Products

--3. Select products, sorted by their price in descending order.

select \*from Products

Order By Price DESC

--4. Update the price of the "Mouse" to 25.

Update Products

set Price = 25

where ProductName = 'Mouse'

--5. Delete the product with ProductID 2 (Mouse).

delete from Products

where ProductID = 2

--6. Select products sorted by ProductName in alphabetical order.

select \*from Products

Order By ProductName ASC

--Assignment 7: Library Management

--Task:

--1. Create a tabled named Books with the below structure and insert below records in the

--table.

create table Books(

BookID int NOT NULL PRIMARY KEY IDENTITY(1,1),

Title varchar(50),

Author varchar(50),

PublishedYear decimal(10,2),

AvailableCopies decimal(10,2)

);

insert into Books(Title, Author, PublishedYear, AvailableCopies)

values

('The Great Gatsby F', 'Scott Fitzgerald' ,1925 ,3),

('1984', 'George Orwell', 1949 ,2),

('To Kill a Mockingbird',' Harper Lee', 1960, 5),

('Brave New World', 'Aldous Huxley', 1932 ,4)

select \* from Books

--2. Update the available copies of "1984" to 5.

Update Books

set AvailableCopies = 5

where Title = '1984'

--3. Delete the book with BookID 1 (The Great Gatsby).

delete from Books

where BookID = 1

--4. Write a query to Select books published after 1950.

select \*from Books where PublishedYear>1950

--Assignment 8: Customer Management

--Task:

--1. Create a Customers table containing details about customers with the below details.

create table Customers(

CustomerID int NOT NULL PRIMARY KEY IDENTITY(201,1),

CustomerName varchar(50),

Email varchar(50),

PhoneNumber varchar(50)

);

insert into Customers(CustomerName, Email, PhoneNumber)

values

('Alice Johnson', 'alice@example.com' ,'555-1234'),

('Bob Smith', 'bob@example.com', '555-5678'),

('Charlie Brown','charlie@example.com', '555-8765')

select \* from Customers

--2.Insert a new customer into the Customers table with the following details:

--a. CustomerID: 204

--b. CustomerName: David Wilson

--c. Email: david@example.com

--d. PhoneNumber: 555-4321

insert into Customers(CustomerName, Email, PhoneNumber)

values

('David Wilson', 'david@example.com' ,'555-4321')

--3. Update the phone number of "Alice Johnson" to "555-9999".

Update Customers

set PhoneNumber = '555-9999'

where CustomerName = 'Alice Johnson'

--4. Delete the customer with CustomerID 202 (Bob Smith).

delete from Customers

where CustomerID = 202

--Assignment 9: Orders Management

--Task:

--1. Create an Orders table that contains information about customer orders with the below

--details:

create table Orders(

OrderID int NOT NULL PRIMARY KEY IDENTITY(301,1),

CustomerID int NOT NULL,

OrderDate varchar(50),

TotalAmount decimal(10,2)

);

insert into Orders(CustomerID, OrderDate ,TotalAmount)

values

(201, '2024-09-01', 250.00),

(202, '2024-09-02', 150.00),

(203 ,'2024-09-03', 200.00)

Select \*from Orders

--2. Insert a new order into the Orders table with the following details:

--a. OrderID: 304

--b. CustomerID: 204

--c. OrderDate: 2024-09-04

--d. TotalAmount: 300.00

insert into Orders(CustomerID, OrderDate ,TotalAmount)

values

(204,'2024-09-04',300.00)

--3. Update the TotalAmount for OrderID 301 to 275.00.

Update Orders

set TotalAmount = 275.00

where OrderID = 301

--4. Delete the order with OrderID 302.

delete from Orders

where OrderID = 302

--5. Select orders with a total amount greater than 200.

select \*from Orders where TotalAmount>200

--6. Select only the OrderID and TotalAmount for all orders.

select OrderID, TotalAmount from Orders

--**Types of keys**

--Assignment 10: Primary Key

--Task:

--Create a Students table with a StudentID as the primary key and include the following columns:

--FirstName, LastName, and Email. Insert three records into the table.

--(1, 'John', 'Doe', 'john.doe@example.com'),

--(2, 'Jane', 'Smith', 'jane.smith@example.com'),

--(3, 'Mike', 'Johnson', 'mike.johnson@example.com');

create table Students(

StudentID int NOT NULL PRIMARY KEY IDENTITY(1,1),

FirstName varchar(50),

LastName varchar(50),

Email varchar(255)

);

insert into Students(FirstName, LastName,Email)

values

( 'John', 'Doe', 'john.doe@example.com'),

( 'Jane', 'Smith', 'jane.smith@example.com'),

('Mike', 'Johnson', 'mike.johnson@example.com')

Select \*from Students

--Assignment 11: Foreign Key

--Task:

--Create a Courses table with a CourseID as the primary key. Create a Enrollments table with a

--StudentID as a foreign key referencing the Students table and a CourseID as a foreign key

--referencing the Courses table. Insert sample data into both tables.

create table Students(

StudentID int NOT NULL PRIMARY KEY IDENTITY(1,1),

FirstName varchar(50),

LastName varchar(50),

Email varchar(255)

);

insert into Students(FirstName, LastName,Email)

values

( 'John', 'Doe', 'john.doe@example.com'),

( 'Jane', 'Smith', 'jane.smith@example.com'),

('Mike', 'Johnson', 'mike.johnson@example.com')

Select \*from Students

--course table

create table Courses(

CourseID int NOT NULL PRIMARY KEY IDENTITY(101,1),

CourseName varchar(50),

CoursePrice decimal(10,2)

);

insert into Courses(CourseName,CoursePrice)

values('Machine Learning', 300.00),

('Cybersecurity', 500.00)

Select \*from Courses

--enroll table

create table Enrollments(

EnrollID int NOT NULL PRIMARY KEY IDENTITY(1,1),

StudentID int,

CourseID int,

FOREIGN KEY (StudentID) REFERENCES Students(StudentID),

FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)

);

insert into Enrollments(StudentID,CourseID)

values (1,102),

(2,101)

Select \*from Enrollments

--Assignment 12: Composite Key

--Task:

--Create an OrderDetails table that contains OrderID, ProductID, and Quantity. Define a

--composite primary key using OrderID and ProductID.

create table OrderDetails(

OrderID int,

ProductID int ,

Quantity int ,

PRIMARY KEY(OrderID, ProductID)

);

insert into OrderDetails(OrderID, ProductID, Quantity)

values(1,101,300),

(2,102,400)

select \*from OrderDetails

--Assignment 13: Candidate Key and Alternate Key

--Task:

--Create a Users table with UserID as the primary key, and include Email and Username as

--candidate keys. Set the Email column as a unique key.

create table Users(

userId int PRIMARY KEY,

emailId varchar(100) UNIQUE,

username varchar(50)

);

--Assignment 14: Surrogate Key

--Task:

--Create a Products table with an AutoID as a surrogate key that automatically increments. Include

--ProductName and Price columns.

CREATE TABLE ProductsTable (

AutoID INT IDENTITY(1,1) PRIMARY KEY,

ProductName NVARCHAR(100) NOT NULL,

Price DECIMAL(10, 2) NOT NULL

);

--Assignment 15: Natural Key

--Task:

--Create a Employees table that uses SSN (Social Security Number) as a natural key. Include

--columns for FirstName, LastName, and Position.

create table Employees(

SSN char(9) PRIMARY KEY,

FirstName varchar(50),

LastName varchar(50),

Position varchar(50)

);

--Assignment 16: Unique Key

--Task:

--Create a Customers table with a CustomerID as the primary key and a PhoneNumber column

--that must be unique.

create table Customerss(

CustomerID int PRIMARY KEY,

CustomerName varchar(100),

PhoneNumber varchar(15) unique,

);

**View - Starting from Assignment 17**

use BankDB

create table Employee(

EmployeeID int Primary Key identity(1,1),

FirstName varchar(50),

LastName varchar(50),

Department varchar(50),

Salary decimal(10,2)

);

insert into Employee

values ('Max','Noel','IT',30000),

('Alfred','Roy','UX',40000),

('Seema','Wioli','Marketing',24000)

insert into Employee

values('Rahul','Dravid','UX',40000)

Select \*from Employee

-- To drop this table

drop table Employee

--Assignment 17

---creating view

—--Task:

1. Create a simple view named EmployeeView that displays the FirstName, LastName, and

Department columns from the Employees table.

2. Query the view to display the details

Create View EmployeeView

as

Select FirstName, LastName ,Department

from Employee;

Select \*from EmployeeView

--Assignment 18: Updating Data Through a View

--Task:

--1. Create a view named EditableEmployeeView that shows EmployeeID, FirstName,

--LastName, and Salary.

--2. Update the salary of an employee using the view.

--3. Verify the update

--Updating a view

Create View EditableEmployeeView

as

Select EmployeeID,FirstName, LastName ,Salary

from Employee;

Select \*from EditableEmployeeView

Update EditableEmployeeView

set Salary = 36000

where EmployeeID=3

Select \*from EditableEmployeeView

-- To drop this view

drop view EditableEmployeeView

--Assignment 19: View with Calculated Columns

--Task:

--1. Create a view named EmployeeSalaryBonusView that displays each employee�s

--FirstName, LastName, Salary, and a calculated column Bonus (10% of the salary).

--2. Query the view

create view EmployeeSalaryBonusView

as

select FirstName, LastName, Salary,Salary\*0.10 as Bonus

from Employee

Select \*from EmployeeSalaryBonusView

-- To drop this view

drop view EmployeeSalaryBonusView

--Assignment 20: Altering a View

--Task:

--1. Alter the EmployeeView from Assignment 1 to include a new column Salary.

--2. Verify the view with the updated column

Alter View EmployeeView

as

Select FirstName, LastName ,Department,Salary

from Employee;

Select \*from EmployeeView

--Assignment 21: View with Parameters (Simulated via Filters)

--Task:

--1. Create a view named FilteredEmployeeView from employees table that displays

--EmployeeID,FirstName,LastName & Department.

--2. Use a query on the view to return employees in a specific department (e.g.,

--DepartmentID = 2).

create view FilteredEmployeeView

as

select EmployeeID,FirstName,LastName,Department

from Employee

select \*from FilteredEmployeeView where Department='UX'

-- To drop this view

drop view FilteredEmployeeView

--Assignment 22: Dropping a View

--Task:

--1. Drop the EmployeeView that was created in the previous assignments.

--2. Try to query the view again.

drop view EmployeeView

select \*from EmployeeView