**SQL Assignment 2**

1)For an online purchasing database, create entity relationship diagrams. Create a database object from your entity diagram.

A) Creating a comprehensive entity-relationship diagram (ERD) for an online purchasing database.

Entities:

Customer

Product

Order

OrderItem

Attributes:

Customer: CustomerID (Primary Key), Name, Email, Address, etc.

Product: ProductID (Primary Key), Name, Price, StockQuantity, etc.

Order: OrderID (Primary Key), CustomerID (Foreign Key), OrderDate, TotalAmount, etc.

CREATE TABLE Customer (

CustomerID INT PRIMARY KEY,

Name NVARCHAR(100),

Email NVARCHAR(100),

Address NVARCHAR(255)

);

CREATE TABLE Product (

ProductID INT PRIMARY KEY,

Name NVARCHAR(100),

Price DECIMAL(10, 2),

StockQuantity INT

);

CREATE TABLE Order (

OrderID INT PRIMARY KEY,

CustomerID INT FOREIGN KEY REFERENCES Customer(CustomerID),

OrderDate DATETIME,

TotalAmount DECIMAL(10, 2)

);

CREATE TABLE OrderItem (

OrderItemID INT PRIMARY KEY,

OrderID INT FOREIGN KEY REFERENCES Order(OrderID),

ProductID INT FOREIGN KEY REFERENCES Product(ProductID),

Quantity INT,

Subtotal DECIMAL(10, 2)

);

2)Create a SQL store process to register the use of the database, complete it with proper validation and transaction rollback and commit.

CREATE PROCEDURE RegisterDatabaseUsage

@UserID INT,

@Activity NVARCHAR(255)

AS

BEGIN

BEGIN TRY

-- Start a transaction

BEGIN TRANSACTION;

-- Check if the user exists

IF NOT EXISTS (SELECT 1 FROM Users WHERE UserID = @UserID)

BEGIN

THROW 50000, 'Invalid user ID. User does not exist.', 1;

END

-- Check if the activity is valid (You can customize the validation logic)

IF NOT EXISTS (SELECT 1 FROM ValidActivities WHERE ActivityName = @Activity)

BEGIN

THROW 50001, 'Invalid activity specified.', 1;

END

-- Log the user activity in the audit table

INSERT INTO AuditLog (UserID, Activity, LogDateTime)

VALUES (@UserID, @Activity, GETDATE());

-- Commit the transaction

COMMIT;

END TRY

BEGIN CATCH

-- An error occurred, rollback the transaction

IF @@TRANCOUNT > 0

ROLLBACK;

-- Log the error or re-throw the error as needed

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

SELECT

@ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE();

3)List the SQL aggregate function and demonstrate how to utilize it.

SQL aggregate functions perform a calculation on a set of values and return a single value. Here are some commonly used SQL aggregate functions:

COUNT(): Counts the number of rows in a set.

SELECT COUNT(\*) AS TotalRows

FROM Employees;

SUM(): Calculates the sum of a numeric column.

SELECT SUM(Salary) AS TotalSalary

FROM Employees;

AVG(): Calculates the average value of a numeric column.

SELECT AVG(Age) AS AverageAge

FROM Customers;

4)In SQL, create a pivot query.

CREATE TABLE Sales (

Product NVARCHAR(50),

Month NVARCHAR(10),

Revenue DECIMAL(10, 2)

);

INSERT INTO Sales (Product, Month, Revenue)

VALUES

('Product\_A', 'Jan', 1000),

('Product\_A', 'Feb', 1200),

('Product\_A', 'Mar', 800),

('Product\_B', 'Jan', 1500),

('Product\_B', 'Feb', 1800),

('Product\_B', 'Mar', 1200);

5) With an example, describe how to join in SQL.

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

CustomerName NVARCHAR(50),

Email NVARCHAR(100)

);

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

TotalAmount DECIMAL(10, 2),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

INSERT INTO Customers (CustomerID, CustomerName, Email)

VALUES

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

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

(3, 'Bob Johnson', 'bob@example.com');

INSERT INTO Orders (OrderID, CustomerID, OrderDate, TotalAmount)

VALUES

(101, 1, '2022-01-15', 500.00),

(102, 2, '2022-02-20', 800.00),

(103, 1, '2022-03-10', 300.00),

(104, 3, '2022-03-25', 1200.00);

SELECT Customers.CustomerID, CustomerName, OrderID, OrderDate, TotalAmount

FROM Customers

INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID;

6) How to locate the 4th highest value in a column in a row. Create your table.

WITH RankedValues AS (

SELECT

RowID,

Value,

ROW\_NUMBER() OVER (PARTITION BY RowID ORDER BY Value DESC) AS RowNum

FROM

SampleData

UNPIVOT

(Value FOR Col IN (Value1, Value2, Value3, Value4, Value5)) AS UnpivotedData

)

SELECT

RowID,

MAX(CASE WHEN RowNum = 4 THEN Value END) AS FourthHighestValue

FROM

RankedValues

WHERE

RowNum <= 4

GROUP BY

RowID;