sql assignment

1. Entity Relationship Diagram (ERD) for Online Purchasing Database

Entities:

- **Customers** (CustomerID, Name, Email, Phone)
- **Products** (ProductID, Name, Price, Stock)
- **Orders** (OrderID, CustomerID, OrderDate)
- **OrderDetails** (OrderDetailID, OrderID, ProductID, Quantity, TotalAmount)

Relationships:

- One customer **can place many orders**.
- Each order **can include many products** (many-to-many via OrderDetails).

ERD (Text Form):

```
lua
CopyEdit
Customers (CustomerID PK)
   |
   |---< Orders (OrderID PK, CustomerID FK)
            |
            |---< OrderDetails (OrderDetailID PK, OrderID FK, ProductID FK)
                          |
                          ——→ Products (ProductID PK)
```

---

2. Create Database Objects

```
sql
CopyEdit
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY IDENTITY,
    Name NVARCHAR(100),
    Email NVARCHAR(100),
    Phone NVARCHAR(20)
);

CREATE TABLE Products (
    ProductID INT PRIMARY KEY IDENTITY,
    Name NVARCHAR(100),
    Price DECIMAL(10, 2),
    Stock INT
);

CREATE TABLE Orders (
    OrderID INT PRIMARY KEY IDENTITY,
    CustomerID INT,
    OrderDate DATETIME DEFAULT GETDATE(),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

CREATE TABLE OrderDetails (
    OrderDetailID INT PRIMARY KEY IDENTITY,
    OrderID INT,
    ProductID INT,
    Quantity INT,
    TotalAmount AS (Quantity * (SELECT Price FROM Products WHERE Product
ID = OrderDetails.ProductID)),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);
```

3. Stored Procedure with Validation, Transaction, Rollback

```
sql
CopyEdit
CREATE PROCEDURE RegisterOrder
    @CustomerID INT,
    @ProductID INT,
    @Quantity INT
AS
BEGIN
    BEGIN TRANSACTION;
    BEGIN TRY
        -- Check if product exists and in stock
        DECLARE @Stock INT;
        SELECT @Stock = Stock FROM Products WHERE ProductID = @ProductI
D;

        IF @Stock IS NULL
        BEGIN
            RAISERROR('Product not found.', 16, 1);
        END

        IF @Stock < @Quantity
        BEGIN
            RAISERROR('Not enough stock available.', 16, 1);
        END

        -- Create Order
        DECLARE @OrderID INT;
        INSERT INTO Orders (CustomerID) VALUES (@CustomerID);
        SET @OrderID = SCOPE_IDENTITY();

        -- Insert Order Details
        INSERT INTO OrderDetails (OrderID, ProductID, Quantity)
        VALUES (@OrderID, @ProductID, @Quantity);

        -- Update stock
        UPDATE Products SET Stock = Stock - @Quantity WHERE ProductID = @
ProductID;

        COMMIT;
        PRINT 'Order registered successfully.';
    END TRY
    BEGIN CATCH
        ROLLBACK;
        PRINT 'Transaction failed: ' + ERROR_MESSAGE();
    END CATCH
END;
```

4. SQL Aggregate Functions and Examples

Function	Description
`SUM()`	Total sum of a column
`AVG()`	Average value
`MAX()`	Maximum value
`MIN()`	Minimum value
`COUNT()`	Count of rows

Example:

```
sql
CopyEdit
SELECT
    COUNT(*) AS TotalOrders,
    SUM(Quantity) AS TotalItemsSold,
    AVG(Quantity) AS AverageQuantity
```

```
FROM OrderDetails;
```

5. Pivot Query Example

Suppose we want to show **total quantity of products sold per month**:

```
sql
CopyEdit
-- Sample Data Query
SELECT
    DATENAME(MONTH, O.OrderDate) AS Month,
    P.Name AS Product,
    OD.Quantity
FROM Orders O
JOIN OrderDetails OD ON O.OrderID = OD.OrderID
JOIN Products P ON P.ProductID = OD.ProductID;

-- Pivot
SELECT *
FROM (
    SELECT
        DATENAME(MONTH, O.OrderDate) AS Month,
        P.Name AS Product,
        OD.Quantity
    FROM Orders O
    JOIN OrderDetails OD ON O.OrderID = OD.OrderID
    JOIN Products P ON P.ProductID = OD.ProductID
) AS SourceTable
PIVOT (
    SUM(Quantity)
    FOR Month IN ([January], [February], [March], [April]) -- etc.
) AS PivotTable;
```

6. SQL Join Example

Join Types:

- **INNER JOIN**: returns matching rows in both tables
- **LEFT JOIN**: returns all rows from left, matching from right
- **RIGHT JOIN**, **FULL JOIN**, etc.

Example – INNER JOIN:

```
sql
CopyEdit
SELECT C.Name AS Customer, O.OrderID, O.OrderDate
FROM Customers C
JOIN Orders O ON C.CustomerID = O.CustomerID;
```

---

7. Find 4th Highest Value in a Column

Step-by-step:

1. Create a table
2. Use `ROW_NUMBER()` or `DISTINCT + TOP` with `ORDER BY`

Example:

```
sql
CopyEdit
-- Create table
CREATE TABLE Sales (
    SaleID INT PRIMARY KEY IDENTITY,
    Amount INT
);

-- Insert sample data
```

```
INSERT INTO Sales (Amount) VALUES (100), (200), (300), (400), (500), (600);

-- Method 1: Using ROW_NUMBER
WITH RankedSales AS (
   SELECT Amount, ROW_NUMBER() OVER (ORDER BY Amount DESC) AS Ran
k
   FROM Sales
)
SELECT Amount AS FourthHighest
FROM RankedSales
WHERE Rank = 4;
```