Sushmit Partakke

23070521156

Sec B2

Prac 5 assignment

Table

CustomerID	Name	Email	TotalSpent
1	John Doe	john@example.com	1500.00
2	Jane Smith	jane@example.com	2000.00
3	Alice Brown	alice@example.com	1200.00

EmployeeID	Name	Salary	ManagerID
1	Mike Johnson	5000.00	NULL
2	Sarah White	4000.00	1
3	James Black	4500.00	1
4	Emily Davis	3000.00	2

ProductID	Name	Category	Price
1	Laptop	Electronics	800.00
2	Smartphone	Electronics	600.00
3	Headphones	Accessories	100.00

```
+---------+------------+------------+------------+-------------+
| OrderID | CustomerID | EmployeeID | OrderDate  | TotalAmount |
+---------+------------+------------+------------+-------------+
|       1 |          1 |          2 | 2024-01-15 |      800.00 |
|       2 |          2 |          3 | 2024-02-20 |      600.00 |
|       3 |          3 |          4 | 2023-05-10 |      100.00 |
+---------+------------+------------+------------+-------------+
```

```
+---------------+---------+-----------+----------+
| OrderDetailID | OrderID | ProductID | Quantity |
+---------------+---------+-----------+----------+
|             1 |       1 |         1 |        1 |
|             2 |       2 |         2 |        2 |
|             3 |       3 |         3 |        5 |
+---------------+---------+-----------+----------+
```

Subquery Tasks

1. Find the **highest-spending customer** in 2024.

   ```
   SELECT Name, TotalSpent
   FROM Customers
   WHERE TotalSpent = (SELECT MAX(TotalSpent) FROM Customers);
   ```

   Output:

   ```
   +------------+------------+
   | Name       | TotalSpent |
   +------------+------------+
   | Jane Smith |    2000.00 |
   +------------+------------+
   ```

2. Retrieve the **most ordered product** based on quantity.

   ```
   SELECT ProductID, SUM(Quantity) AS TotalQuantity
   FROM OrderDetails
   GROUP BY ProductID
   ORDER BY TotalQuantity DESC
   LIMIT 1;
   ```

   Output:

   ```
   +-----------+---------------+
   | ProductID | TotalQuantity |
   +-----------+---------------+
   |         3 |             5 |
   +-----------+---------------+
   ```

3. Find employees who **earn more than the lowest-paid manager**.

```
SELECT Name, Salary
FROM Employees
WHERE Salary > (SELECT MIN(Salary) FROM Employees WHERE ManagerID IS NULL);
```

Output:

```
Program did not output anything!
```

4. Retrieve customers who **placed orders only in 2023 but not in 2024**.

```
SELECT DISTINCT c.Name
FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID
WHERE YEAR(o.OrderDate) = 2023
AND c.CustomerID NOT IN (SELECT CustomerID FROM Orders WHERE YEAR(OrderDate)
= 2024);
```

```
+-------------+
| Name        |
+-------------+
| Alice Brown |
+-------------+
```

5. Find the **total revenue generated in February 2024**.

```
SELECT SUM(TotalAmount) AS TotalRevenue

FROM Orders

WHERE MONTH(OrderDate) = 2 AND YEAR(OrderDate) = 2024;
```

```
+--------------+
| TotalRevenue |
+--------------+
|       600.00 |
+--------------+
```

Joins Tasks

1. Find the **top 3 customers** with the **highest total spending**.

```
SELECT Name, TotalSpent
FROM Customers
ORDER BY TotalSpent DESC
LIMIT 3;
```

Output:

```
+-------------+------------+
| Name        | TotalSpent |
+-------------+------------+
| Jane Smith  |    2000.00 |
| John Doe    |    1500.00 |
| Alice Brown |    1200.00 |
+-------------+------------+
```

2. Retrieve **employee names** along with the **total revenue generated from their assigned orders**.

```
SELECT e.Name, SUM(o.TotalAmount) AS TotalRevenue
FROM Employees e
JOIN Orders o ON e.EmployeeID = o.EmployeeID
GROUP BY e.Name;
```

Output:

```
+-------------+--------------+
| Name        | TotalRevenue |
+-------------+--------------+
| Sarah White |       800.00 |
| James Black |       600.00 |
| Emily Davis |       100.00 |
+-------------+--------------+
```

3. Show the **most ordered product category** and its total quantity sold.

```
SELECT p.Category, SUM(od.Quantity) AS TotalQuantity
FROM Products p
JOIN OrderDetails od ON p.ProductID = od.ProductID
GROUP BY p.Category
ORDER BY TotalQuantity DESC
LIMIT 1;
```

Output:

```
+-------------+---------------+
| Category    | TotalQuantity |
+-------------+---------------+
| Accessories |             5 |
+-------------+---------------+
```

4. Retrieve employees who **earn more than their colleagues** using a **SELF JOIN**.

```
SELECT e1.Name, e1.Salary
FROM Employees e1
JOIN Employees e2 ON e1.Salary > e2.Salary
GROUP BY e1.Name, e1.Salary;
```

Output:

```
+--------------+---------+
| Name         | Salary  |
+--------------+---------+
| James Black  | 4500.00 |
| Mike Johnson | 5000.00 |
| Sarah White  | 4000.00 |
+--------------+---------+
```

5. Find employees who **work under the same manager** using a **SELF JOIN**.

   SELECT e1.Name AS Employee, e2.Name AS Manager

   FROM Employees e1

   JOIN Employees e2 ON e1.ManagerID = e2.EmployeeID;

```
+-------------+--------------+
| Employee    | Manager      |
+-------------+--------------+
| Sarah White | Mike Johnson |
| James Black | Mike Johnson |
| Emily Davis | Sarah White  |
+-------------+--------------+
```