

Foundation Technical Training

Assignment No. 1 - SQL - Electronic Gadgets - Task 4

Name: Niranjan Kolpe, Batch: C#-Batch 2

Tasks 4: Subquery and its type:

1. Write an SQL query to find out which customers have not placed any orders.
2. Write an SQL query to find the total number of products available for sale.
3. Write an SQL query to calculate the total revenue generated by TechShop.
4. Write an SQL query to calculate the average quantity ordered for products in a specific category. Allow users to input the category name as a parameter.
5. Write an SQL query to calculate the total revenue generated by a specific customer. Allow users to input the customer ID as a parameter.
6. Write an SQL query to find the customers who have placed the most orders. List their names and the number of orders they've placed.
7. Write an SQL query to find the most popular product category, which is the one with the highest total quantity ordered across all orders.
8. Write an SQL query to find the customer who has spent the most money (highest total revenue) on electronic gadgets. List their name and total spending.
9. Write an SQL query to calculate the average order value (total revenue divided by the number of orders) for all customers.
10. Write an SQL query to find the total number of orders placed by each customer and list their names along with the order count

Program Code in SQL:

```
-- Niranjan Kolpe - C# Batch-2
-- SQL - Assignment 1 - Electronic Gadgets
-- Task 4: Subquery and its type:

--1. Write an SQL query to find out which customers have not placed any orders.
SELECT Customers.CustomerID, FirstName, LastName FROM Customers LEFT JOIN Orders
ON Customers.CustomerID=Orders.CustomerID WHERE Orders.OrderID IS NULL;

--2. Write an SQL query to find the total number of products available for sale.
```

```
SELECT COUNT(*) AS NumberOfProductsAvailable FROM Inventory WHERE  
QuantityInStock!=0;
```

--3. Write an SQL query to calculate the total revenue generated by TechShop.

```
SELECT SUM(TotalAmount) AS TotalRevenue FROM Orders;
```

--4. Write an SQL query to calculate the average quantity ordered for products in a specific category.

--Allow users to input the category name as a parameter.

```
DECLARE @ProductName VARCHAR(20) = 'Keyboard';  
  
SELECT ProductName, AVG(Quantity) AS AverageQuantity  
FROM OrderDetails JOIN Products ON OrderDetails.ProductID=Products.ProductID  
GROUP BY ProductName HAVING ProductName=@ProductName;
```

--5. Write an SQL query to calculate the total revenue generated by a specific customer. Allow users

--to input the customer ID as a parameter.

```
DECLARE @CustomerID INT = 3;  
  
SELECT Customers.CustomerID, FirstName, SUM(TotalAmount) AS Revenue  
FROM Customers JOIN Orders ON Customers.CustomerID=Orders.CustomerID  
GROUP BY Customers.CustomerID, FirstName HAVING Customers.CustomerID=@CustomerID;
```

--6. Write an SQL query to find the customers who have placed the most orders. List their names

--and the number of orders they've placed.

```
SELECT Customers.CustomerID, FirstName, Count(*) AS NumberOfOrdersPlaced  
FROM Customers JOIN Orders ON Customers.CustomerID=Orders.CustomerID
```

```
GROUP BY Customers.CustomerID, FirstName ORDER BY NumberOfOrdersPlaced DESC;
```

--7. Write an SQL query to find the most popular product category, which is the one with the highest

--total quantity ordered across all orders.

```
SELECT Products.ProductID, Products.ProductName, SUM(Quantity) AS TotalQuantity
FROM Products JOIN OrderDetails ON Products.ProductID=OrderDetails.ProductID
GROUP BY Products.ProductID, ProductName ORDER BY TotalQuantity DESC;
```

--8. Write an SQL query to find the customer who has spent the most money (highest total revenue)

--on electronic gadgets. List their name and total spending.

```
SELECT TOP 1 Customers.CustomerID, FirstName, SUM(TotalAmount) AS Revenue
FROM Customers JOIN Orders ON Customers.CustomerID=Orders.CustomerID
GROUP BY Customers.CustomerID, FirstName ORDER BY Revenue DESC;
```

--9. Write an SQL query to calculate the average order value (total revenue divided by the number of orders)

-- for all customers.

```
SELECT Customers.CustomerID, FirstName,
SUM(Orders.TotalAmount/Customers.NumberOfOrders) AS AverageOrderValue
FROM Customers JOIN Orders ON Customers.CustomerID=Orders.CustomerID
GROUP BY Customers.CustomerID, FirstName ORDER BY AverageOrderValue DESC;
```

--10. Write an SQL query to find the total number of orders placed by each customer and list their

--names along with the order count.

```
SELECT Customers.CustomerID, FirstName, COUNT(*) AS NumberOfOrders
```

```
FROM Customers JOIN Orders ON Customers.CustomerID=Orders.CustomerID  
GROUP BY Customers.CustomerID, FirstName;
```

Output 1:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure, including tables like `dbo.Inventory` and `dbo.Orders`. The main query window contains the following SQL code:

```
-- Nirranjan Kolpe - C# Batch-2
-- SQL - Assignment 1 - Electronic Gadgets
-- Task 4: Subquery and its type:

--1. Write an SQL query to find out which customers have not placed any orders.
SELECT Customers.CustomerID, FirstName, LastName FROM Customers LEFT JOIN Orders
ON Customers.CustomerID=Orders.CustomerID WHERE Orders.OrderID IS NULL;

--2. Write an SQL query to find the total number of products available for sale.
SELECT COUNT(*) AS NumberOfProductsAvailable FROM Inventory WHERE QuantityInStock!=0;

--3. Write an SQL query to calculate the total revenue generated by TechShop.
SELECT SUM(TotalAmount) AS TotalRevenue FROM Orders;
```

The query results are displayed in the Results pane, showing two rows of data:

CustomerID	FirstName	LastName
6	Thor	Odinson
10	Peter	Parker

The status bar at the bottom indicates that the query was executed successfully, returning 2 rows.

Output 2:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure, including tables like `dbo.Inventory` and `dbo.Orders`. The main query window contains the following SQL code:

```
-- Nirranjan Kolpe - C# Batch-2
-- SQL - Assignment 1 - Electronic Gadgets
-- Task 4: Subquery and its type:

--1. Write an SQL query to find out which customers have not placed any orders.
SELECT Customers.CustomerID, FirstName, LastName FROM Customers LEFT JOIN Orders
ON Customers.CustomerID=Orders.CustomerID WHERE Orders.OrderID IS NULL;

--2. Write an SQL query to find the total number of products available for sale.
SELECT COUNT(*) AS NumberOfProductsAvailable FROM Inventory WHERE QuantityInStock!=0;

--3. Write an SQL query to calculate the total revenue generated by TechShop.
SELECT SUM(TotalAmount) AS TotalRevenue FROM Orders;
```

The query results are displayed in the Results pane, showing one row of data:

NumberOfProductsAvailable
10

The status bar at the bottom indicates that the query was executed successfully, returning 1 row.

Output 3:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'NiranjanDB', including tables like 'dbo.Inventory', 'dbo.OrderDetails', and 'dbo.Orders'. The main query window contains the following SQL code:

```
-- Niranjan Kolpe - C# Batch-2
-- SQL - Assignment 1 - Electronic Gadgets
-- Task 4: Subquery and its type:

--1. Write an SQL query to find out which customers have not placed any orders.
SELECT Customers.CustomerID, FirstName, LastName FROM Customers LEFT JOIN Orders
ON Customers.CustomerID=Orders.CustomerID WHERE Orders.OrderID IS NULL;

--2. Write an SQL query to find the total number of products available for sale.
SELECT COUNT(*) AS NumberOfProductsAvailable FROM Inventory WHERE QuantityInStock!=0;

--3. Write an SQL query to calculate the total revenue generated by TechShop.
SELECT SUM(TotalAmount) AS TotalRevenue FROM Orders;
```

The Results pane shows the output of the third query, displaying a single row with the total revenue:

TotalRevenue
27700

The status bar at the bottom indicates 'Query executed successfully.' and '1 rows'.

Output 4:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'NiranjanDB'. The main query window contains the following SQL code:

```
SELECT COUNT(*) AS NumberOfProductsAvailable FROM Inventory WHERE QuantityInStock!=0;

--3. Write an SQL query to calculate the total revenue generated by TechShop.
SELECT SUM(TotalAmount) AS TotalRevenue FROM Orders;

--4. Write an SQL query to calculate the average quantity ordered for products in a specific category.
--Allow users to input the category name as a parameter.
DECLARE @ProductName VARCHAR(20) = 'Keyboard';
SELECT ProductName, AVG(Quantity) AS AverageQuantity
FROM OrderDetails JOIN Products ON OrderDetails.ProductID=Products.ProductID
GROUP BY ProductName HAVING ProductName=@ProductName;

--5. Write an SQL query to calculate the total revenue generated by a specific customer. Allow users
--to input the customer ID as a parameter.
```

The Results pane shows the output of the fourth query, displaying a single row with the average quantity for 'Keyboard':

ProductName	AverageQuantity
Keyboard	6

The status bar at the bottom indicates 'Query executed successfully.' and '1 rows'.

Output 5:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure, including tables like Inventory, OrderDetails, and Orders. The main query window contains the following SQL code:

```
--Allow users to input the category name as a parameter.
DECLARE @ProductName VARCHAR(20) = 'Keyboard';
SELECT ProductName, AVG(Quantity) AS AverageQuantity
FROM OrderDetails JOIN Products ON OrderDetails.ProductID=Products.ProductID
GROUP BY ProductName HAVING ProductName=@ProductName;

--5. Write an SQL query to calculate the total revenue generated by a specific customer. Allow users
--to input the customer ID as a parameter.
DECLARE @CustomerID INT = 3;
SELECT Customers.CustomerID, FirstName, SUM(TotalAmount) AS Revenue
FROM Customers JOIN Orders ON Customers.CustomerID=Orders.CustomerID
GROUP BY Customers.CustomerID, FirstName HAVING Customers.CustomerID=@CustomerID;
```

The Results pane shows the output of the second query:

CustomerID	FirstName	Revenue
3	Tony	10000

The status bar at the bottom indicates "Query executed successfully." and "1 rows".

Output 6:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure. The main query window contains the following SQL code:

```
--6. Write an SQL query to find the customers who have placed the most orders. List their names
--and the number of orders they've placed.
SELECT Customers.CustomerID, FirstName, Count(*) AS NumberOfOrdersPlaced
FROM Customers JOIN Orders ON Customers.CustomerID=Orders.CustomerID
GROUP BY Customers.CustomerID, FirstName ORDER BY NumberOfOrdersPlaced DESC;

--7. Write an SQL query to find the most popular product category, which is the one with the highest
--total quantity ordered across all orders.
SELECT Products.ProductID, Products.ProductName, SUM(Quantity) AS TotalQuantity
FROM Products JOIN OrderDetails ON Products.ProductID=OrderDetails.ProductID
GROUP BY Products.ProductID, Products.ProductName ORDER BY TotalQuantity DESC;
```

The Results pane shows the output of the first query:

CustomerID	FirstName	NumberOfOrdersPlaced
2	Steve	2
3	Tony	1
4	Natasha	1
5	Bruce	1
7	Clint	1
8	Wanda	1
9	Sam	1
1	Niranjan	1

The status bar at the bottom indicates "Query executed successfully." and "8 rows".

Output 7:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'NiranjanDB', including tables like 'dbo.Inventory', 'dbo.OrderDetails', and 'dbo.Orders'. The main query editor contains the following SQL code:

```
--and the number of orders they've placed.
SELECT Customers.CustomerID, FirstName, Count(*) AS NumberOfOrdersPlaced
FROM Customers JOIN Orders ON Customers.CustomerID=Orders.CustomerID
GROUP BY Customers.CustomerID, FirstName ORDER BY NumberOfOrdersPlaced DESC;

--7. Write an SQL query to find the most popular product category, which is the one with the highest
--total quantity ordered across all orders.
SELECT Products.ProductID, Products.ProductName, SUM(Quantity) AS TotalQuantity
FROM Products JOIN OrderDetails ON Products.ProductID=OrderDetails.ProductID
GROUP BY Products.ProductID, ProductName ORDER BY TotalQuantity DESC;

--8. Write an SQL query to find the customer who has spent the most money (highest total revenue)
--on electronic gadgets. List their name and total spending.
SELECT TOP 1 Customers.CustomerID, FirstName, SUM(TotalAmount) AS Revenue
FROM Customers JOIN Orders ON Customers.CustomerID=Orders.CustomerID
GROUP BY Customers.CustomerID, FirstName ORDER BY Revenue DESC;
```

The Results pane shows the output of the third query, displaying a table with 7 rows:

ProductID	ProductName	TotalQuantity
9	Keyboard	12
4	Earphones	5
7	Router	4
8	Monitor	1
2	Powerbank	1
3	Airpods	1
10	Charger	1

The status bar at the bottom indicates 'Query executed successfully.' and '7 rows'.

Output 8:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'NiranjanDB'. The main query editor contains the following SQL code:

```
--total quantity ordered across all orders.
SELECT Products.ProductID, Products.ProductName, SUM(Quantity) AS TotalQuantity
FROM Products JOIN OrderDetails ON Products.ProductID=OrderDetails.ProductID
GROUP BY Products.ProductID, ProductName ORDER BY TotalQuantity DESC;

--8. Write an SQL query to find the customer who has spent the most money (highest total revenue)
--on electronic gadgets. List their name and total spending.
SELECT TOP 1 Customers.CustomerID, FirstName, SUM(TotalAmount) AS Revenue
FROM Customers JOIN Orders ON Customers.CustomerID=Orders.CustomerID
GROUP BY Customers.CustomerID, FirstName ORDER BY Revenue DESC;

--9. Write an SQL query to calculate the average order value (total revenue divided by the number of orders)
--for all customers.
SELECT Customers.CustomerID, FirstName, SUM(Orders.TotalAmount/Customers.NumberOfOrders) AS AverageOrderValue
FROM Customers JOIN Orders ON Customers.CustomerID=Orders.CustomerID
GROUP BY Customers.CustomerID, FirstName;
```

The Results pane shows the output of the third query, displaying a table with 1 row:

CustomerID	FirstName	Revenue
3	Tony	10000

The status bar at the bottom indicates 'Query executed successfully.' and '1 rows'.

Output 9:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'NiranjanDB', including tables like 'dbo.Inventory', 'dbo.OrderDetails', and 'dbo.Orders'. The main query window contains the following SQL code:

```
GROUP BY Customers.CustomerID, FirstName ORDER BY Revenue DESC;

--9. Write an SQL query to calculate the average order value (total revenue divided by the number of orders)
-- for all customers.
SELECT Customers.CustomerID, FirstName, SUM(Orders.TotalAmount/Customers.NumberOfOrders) AS AverageOrderValue
FROM Customers JOIN Orders ON Customers.CustomerID=Orders.CustomerID
GROUP BY Customers.CustomerID, FirstName ORDER BY AverageOrderValue DESC;

--10. Write an SQL query to find the total number of orders placed by each customer and list their
--names along with the order count.
SELECT Customers.CustomerID, FirstName, COUNT(*) AS NumberOfOrders
FROM Customers JOIN Orders ON Customers.CustomerID=Orders.CustomerID
GROUP BY Customers.CustomerID, FirstName;
```

The Results pane shows the output of the first query, displaying a table with 8 rows:

CustomerID	FirstName	AverageOrderValue
3	Tony	10000
8	Wanda	6000
4	Natasha	4000
7	Clint	3000
9	Sam	2000
5	Bruce	1000
2	Steve	750
1	Niranjan	200

The status bar at the bottom indicates 'Query executed successfully.' and '8 rows'.

Output 10:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'NiranjanDB'. The main query window contains the following SQL code:

```
GROUP BY Customers.CustomerID, FirstName ORDER BY Revenue DESC;

--9. Write an SQL query to calculate the average order value (total revenue divided by the number of orders)
-- for all customers.
SELECT Customers.CustomerID, FirstName, SUM(Orders.TotalAmount/Customers.NumberOfOrders) AS AverageOrderValue
FROM Customers JOIN Orders ON Customers.CustomerID=Orders.CustomerID
GROUP BY Customers.CustomerID, FirstName ORDER BY AverageOrderValue DESC;

--10. Write an SQL query to find the total number of orders placed by each customer and list their
--names along with the order count.
SELECT Customers.CustomerID, FirstName, COUNT(*) AS NumberOfOrders
FROM Customers JOIN Orders ON Customers.CustomerID=Orders.CustomerID
GROUP BY Customers.CustomerID, FirstName;
```

The Results pane shows the output of the second query, displaying a table with 8 rows:

CustomerID	FirstName	NumberOfOrders
1	Niranjan	1
2	Steve	2
3	Tony	1
4	Natasha	1
5	Bruce	1
7	Clint	1
8	Wanda	1
9	Sam	1

The status bar at the bottom indicates 'Query executed successfully.' and '8 rows'.