

Foundation Technical Training

Assignment No. 1 - SQL - Electronic Gadgets - Task 3

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

Tasks 3: Aggregate functions, Having, Order By, GroupBy and Joins:

1. Write an SQL query to retrieve a list of all orders along with customer information (e.g., customer name) for each order.
2. Write an SQL query to find the total revenue generated by each electronic gadget product. Include the product name and the total revenue.
3. Write an SQL query to list all customers who have made at least one purchase. Include their names and contact information.
4. Write an SQL query to find the most popular electronic gadget, which is the one with the highest total quantity ordered. Include the product name and the total quantity ordered.
5. Write an SQL query to retrieve a list of electronic gadgets along with their corresponding categories.
6. Write an SQL query to calculate the average order value for each customer. Include the customer's name and their average order value.
7. Write an SQL query to find the order with the highest total revenue. Include the order ID, customer information, and the total revenue.
8. Write an SQL query to list electronic gadgets and the number of times each product has been ordered.
9. Write an SQL query to find customers who have purchased a specific electronic gadget product. Allow users to input the product name as a parameter.
10. Write an SQL query to calculate the total revenue generated by all orders placed within a specific time period. Allow users to input the start and end dates as parameters.

Program Code in SQL:

```
-- Niranjan Kolpe - C# Batch-2
```

```
-- SQL - Assignment 1 - Electronic Gadgets
```

```
-- Task 3
```

```
--1. Write an SQL query to retrieve a list of all orders along with customer  
information (e.g., customer name)
```

```
-- for each order.
```

```
SELECT Orders.OrderID, Orders.OrderDate, Customers.FirstName, Customers.LastName  
, Orders.TotalAmount
```

```
FROM Orders JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

--2. Write an SQL query to find the total revenue generated by each electronic gadget product.

-- Include the product name and the total revenue.

```
ALTER TABLE Products ALTER COLUMN ProductName VARCHAR(20) NOT NULL;
```

```
SELECT Products.ProductName, SUM(OrderDetails.Quantity * Products.Price) AS  
TotalRevenue
```

```
FROM OrderDetails JOIN Products ON OrderDetails.ProductID = Products.ProductID  
GROUP BY Products.ProductName;
```

--3. Write an SQL query to list all customers who have made at least one purchase.

-- Include their names and contact information.

```
ALTER TABLE Customers ALTER COLUMN FirstName VARCHAR(20) NOT NULL;
```

```
ALTER TABLE Customers ALTER COLUMN LastName VARCHAR(20) NOT NULL;
```

```
SELECT DISTINCT Customers.FirstName, Customers.LastName, Customers.Email,  
Customers.Phone, Customers.Address
```

```
FROM Customers JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```

--4. Write an SQL query to find the most popular electronic gadget,

-- which is the one with the highest total quantity ordered.

-- Include the product name and the total quantity ordered.

```
SELECT TOP 1 Products.ProductName, SUM(OrderDetails.Quantity) AS  
TotalQuantityOrdered
```

```
FROM OrderDetails JOIN Products ON OrderDetails.ProductID = Products.ProductID
```

```
GROUP BY Products.ProductName ORDER BY TotalQuantityOrdered DESC;
```

--5. Write an SQL query to retrieve a list of electronic gadgets along with their corresponding categories.

```
SELECT ProductName, Description FROM Products;
```

--6. Write an SQL query to calculate the average order value for each customer.

-- Include the customer's name and their average order value.

```
SELECT Customers.FirstName, Customers.LastName, AVG(Orders.TotalAmount) AS
AverageOrderValue

FROM Orders JOIN Customers ON Orders.CustomerID = Customers.CustomerID

GROUP BY Customers.FirstName, Customers.LastName;
```

--7. Write an SQL query to find the order with the highest total revenue.

-- Include the order ID, customer information, and the total revenue.

```
SELECT TOP 1 Orders.OrderID, Customers.CustomerID, Customers.FirstName,
Customers.LastName, Orders.TotalAmount AS TotalRevenue

FROM Orders JOIN Customers ON Orders.CustomerID = Customers.CustomerID ORDER BY
TotalRevenue DESC;
```

--8. Write an SQL query to list electronic gadgets and the number of times each product has been ordered.

```
SELECT Products.ProductName, COUNT(OrderDetails.ProductID) AS NumberOfOrders

FROM OrderDetails JOIN Products ON OrderDetails.ProductID = Products.ProductID
GROUP BY Products.ProductName;
```

--9. Write an SQL query to find customers who have purchased a specific electronic gadget product.

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

--ALTER TABLE Orders DROP COLUMN Status;

--ALTER TABLE Orders ADD OrdersStatus VARCHAR(20);

```
--UPDATE Orders SET OrdersStatus='Pending';  
  
--UPDATE Orders SET OrdersStatus='Shipped' WHERE OrderID=8 AND CustomerID=3;
```

```
DECLARE @ProductName VARCHAR(20) = 'Keyboard';  
  
SELECT * FROM Customers  
  
JOIN Orders ON Customers.CustomerID = Orders.CustomerID  
  
JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID  
  
JOIN Products ON OrderDetails.ProductID = Products.ProductID  
  
WHERE Products.ProductName = @ProductName;
```

--10. Write an SQL query to calculate the total revenue generated by all orders placed within a specific time period.

-- Allow users to input the start and end dates as parameters.

```
DECLARE @StartDate DATE='2024-09-03', @EndDate DATE = '2024-09-16';  
  
SELECT SUM(Orders.TotalAmount) AS TotalRevenue FROM Orders WHERE Orders.OrderDate  
BETWEEN @StartDate AND @EndDate;
```

Output 1:

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 'Orders' and 'Products'. The main window shows a SQL query in the 'SQLQuery.sql' file. The query is as follows:

```
-- Niranjandb - C# Batch-2
-- SQL - Assignment 1 - Electronic Gadgets
-- Task 3

--1. Write an SQL query to retrieve a list of all orders along with customer information (e.g., customer name)
-- for each order.
SELECT Orders.OrderID, Orders.OrderDate, Customers.FirstName, Customers.LastName, Orders.TotalAmount
FROM Orders JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

--2. Write an SQL query to find the total revenue generated by each electronic gadget product.
-- Include the product name and the total revenue.
ALTER TABLE Products ALTER COLUMN ProductName VARCHAR(20) NOT NULL;
SELECT Products.ProductName, SUM(OrderDetails.Quantity * Products.Price) AS TotalRevenue
FROM OrderDetails JOIN Products ON OrderDetails.ProductID = Products.ProductID GROUP BY Products.ProductName;
```

The results of the first query are displayed in the 'Results' pane, showing a list of orders with columns: OrderID, OrderDate, FirstName, LastName, and TotalAmount.

OrderID	OrderDate	FirstName	LastName	TotalAmount
1	2024-09-01	Bruce	Banner	1000
2	2024-09-02	Clint	Barton	3000
3	2024-09-03	Natasha	Romanoff	4000
4	2024-09-04	Sam	Wilson	2000
5	2024-09-06	Steve	Rogers	500
6	2024-09-15	Niranjandb	Kolpe	200
7	2024-09-16	Tony	Stark	10000
8	2024-09-17	Wanda	Maximoff	6000
9	2024-09-19	Steve	Rogers	1000

The status bar at the bottom indicates that the query was executed successfully, returning 9 rows.

Output 2:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'Niranjandb'. The main window shows a SQL query in the 'SQLQuery.sql' file. The query is as follows:

```
--1. Write an SQL query to retrieve a list of all orders along with customer information (e.g., customer name)
-- for each order.
SELECT Orders.OrderID, Orders.OrderDate, Customers.FirstName, Customers.LastName, Orders.TotalAmount
FROM Orders JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

--2. Write an SQL query to find the total revenue generated by each electronic gadget product.
-- Include the product name and the total revenue.
ALTER TABLE Products ALTER COLUMN ProductName VARCHAR(20) NOT NULL;
SELECT Products.ProductName, SUM(OrderDetails.Quantity * Products.Price) AS TotalRevenue
FROM OrderDetails JOIN Products ON OrderDetails.ProductID = Products.ProductID GROUP BY Products.ProductName;

--3. Write an SQL query to list all customers who have made at least one purchase.
-- Include their names and contact information.
ALTER TABLE Customers ALTER COLUMN FirstName VARCHAR(20) NOT NULL;
ALTER TABLE Customers ALTER COLUMN LastName VARCHAR(20) NOT NULL;
SELECT DISTINCT Customers.FirstName, Customers.LastName, Customers.Email, Customers.Phone, Customers.Address
```

The results of the second query are displayed in the 'Results' pane, showing a list of products with columns: ProductName and TotalRevenue.

ProductName	TotalRevenue
Airpods	1000
Charger	200
Earphones	500
Keyboard	12000
Monitor	4000
Powerbank	3000
Router	6000

The status bar at the bottom indicates that the query was executed successfully, returning 7 rows.

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.Orders' and 'dbo.Products'. The main query editor contains the following SQL code:

```
SELECT Products.ProductName, SUM(OrderDetails.Quantity * Products.Price) AS TotalRevenue
FROM OrderDetails JOIN Products ON OrderDetails.ProductID = Products.ProductID GROUP BY Products.ProductName;

--3. Write an SQL query to list all customers who have made at least one purchase.
-- Include their names and contact information.
ALTER TABLE Customers ALTER COLUMN FirstName VARCHAR(20) NOT NULL;
ALTER TABLE Customers ALTER COLUMN LastName VARCHAR(20) NOT NULL;
SELECT DISTINCT Customers.FirstName, Customers.LastName, Customers.Email, Customers.Phone, Customers.Address
FROM Customers JOIN Orders ON Customers.CustomerID = Orders.CustomerID;

--4. Write an SQL query to find the most popular electronic gadget,
-- which is the one with the highest total quantity ordered.
-- Include the product name and the total quantity ordered.
SELECT TOP 1 Products.ProductName, SUM(OrderDetails.Quantity) AS TotalQuantityOrdered
```

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

FirstName	LastName	Email	Phone	Address
Bruce	Banner	bruce@gmail.com	9999999995	Delhi
Clint	Barton	clintbarton@gmail.com	9999999997	Chennai
Natasha	Romanoff	natasha@gmail.com	9999999994	New York
Niranjan	Kolpe	niranjan@gmail.com	9999999991	San Francisco
Sam	Wilson	sam@gmail.com	9999999999	San Francisco
Steve	Rogers	steve@gmail.com	9999999992	Brooklyn
Tony	Stark	tony@gmail.com	9999999993	New York
Wanda	Maximoff	wanda@gmail.com	9999999998	Mumbai

The status bar at the bottom indicates the query was executed successfully, returning 8 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 editor contains the following SQL code:

```
--4. Write an SQL query to find the most popular electronic gadget,
-- which is the one with the highest total quantity ordered.
-- Include the product name and the total quantity ordered.
SELECT TOP 1 Products.ProductName, SUM(OrderDetails.Quantity) AS TotalQuantityOrdered
FROM OrderDetails JOIN Products ON OrderDetails.ProductID = Products.ProductID
GROUP BY Products.ProductName ORDER BY TotalQuantityOrdered DESC;

--5. Write an SQL query to retrieve a list of electronic gadgets along with their corresponding categories.
SELECT ProductName, Description FROM Products;

--6. Write an SQL query to calculate the average order value for each customer.
-- Include the customer's name and their average order value.
SELECT Customers.FirstName, Customers.LastName, AVG(Orders.TotalAmount) AS AverageOrderValue
```

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

ProductName	TotalQuantityOrdered
Keyboard	12

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

Output 5:

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.Orders' and 'dbo.Products'. The main query window contains the following SQL code:

```
--4. Write an SQL query to find the most popular electronic gadget,
-- which is the one with the highest total quantity ordered.
-- Include the product name and the total quantity ordered.
SELECT TOP 1 Products.ProductName, SUM(OrderDetails.Quantity) AS TotalQuantityOrdered
FROM OrderDetails JOIN Products ON OrderDetails.ProductID = Products.ProductID
GROUP BY Products.ProductName ORDER BY TotalQuantityOrdered DESC;

--5. Write an SQL query to retrieve a list of electronic gadgets along with their corresponding categories.
SELECT ProductName, Description FROM Products;

--6. Write an SQL query to calculate the average order value for each customer.
-- Include the customer's name and their average order value.
SELECT Customers.FirstName, Customers.LastName, AVG(Orders.TotalAmount) AS AverageOrderValue
```

The Results pane shows the output of the third query, displaying a list of products and their descriptions:

ProductName	Description
Smartphone	Android Smartphone
Powerbank	Powerbank for Smartphones
Airpods	Wireless Airpods
Earphones	Wired Earphones
USB Cable	USB Cable
Mouse	Wired Mouse
Router	WiFi Router
Monitor	Desktop Monitor
Keyboard	PC Keyboard
Charger	Mobile Charger
Speaker	Bluetooth Speaker

The status bar at the bottom indicates that the query was executed successfully, returning 11 rows.

Output 6:

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 ProductName, Description FROM Products;

--6. Write an SQL query to calculate the average order value for each customer.
-- Include the customer's name and their average order value.
SELECT Customers.FirstName, Customers.LastName, AVG(Orders.TotalAmount) AS AverageOrderValue
FROM Orders JOIN Customers ON Orders.CustomerID = Customers.CustomerID
GROUP BY Customers.FirstName, Customers.LastName;

--7. Write an SQL query to find the order with the highest total revenue.
-- Include the order ID, customer information, and the total revenue.
SELECT TOP 1 Orders.OrderID, Customers.CustomerID, Customers.FirstName, Customers.LastName, Orders.TotalAmount AS TotalRevenue
FROM Orders JOIN Customers ON Orders.CustomerID = Customers.CustomerID ORDER BY TotalRevenue DESC;

--8. Write an SQL query to list electronic gadgets and the number of times each product has been ordered.
```

The Results pane shows the output of the second query, displaying a list of customers and their average order values:

FirstName	LastName	AverageOrderValue
Bruce	Banner	1000
Clint	Barton	3000
Niranjan	Kolpe	200
Wanda	Maximoff	6000
Steve	Rogers	750
Natasha	Romanoff	4000
Tony	Stark	10000
Sam	Wilson	2000

The status bar at the bottom indicates that the query was executed successfully, returning 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 'dbo.Orders' and 'dbo.Products'. The main query window contains the following SQL code:

```
-- Include the customer's name and their average order value.
SELECT Customers.FirstName, Customers.LastName, AVG(Orders.TotalAmount) AS AverageOrderValue
FROM Orders JOIN Customers ON Orders.CustomerID = Customers.CustomerID
GROUP BY Customers.FirstName, Customers.LastName;

--7. Write an SQL query to find the order with the highest total revenue.
-- Include the order ID, customer information, and the total revenue.
SELECT TOP 1 Orders.OrderID, Customers.CustomerID, Customers.FirstName, Customers.LastName, Orders.TotalAmount AS TotalRevenue
FROM Orders JOIN Customers ON Orders.CustomerID = Customers.CustomerID ORDER BY TotalRevenue DESC;

--8. Write an SQL query to list electronic gadgets and the number of times each product has been ordered.
SELECT Products.ProductName, COUNT(OrderDetails.ProductID) AS NumberOfOrders
FROM OrderDetails JOIN Products ON OrderDetails.ProductID = Products.ProductID GROUP BY Products.ProductName;
```

The Results pane shows the output of the second query, displaying a single row with the following data:

OrderID	CustomerID	FirstName	LastName	TotalRevenue
1	8	Tony	Stark	10000

The status bar at the bottom indicates 'Query executed successfully.' and 'DESKTOP-55ISL9F (16.0 RTM) | DESKTOP-55ISL9F\Niran... | NiranjanDB | 00:00:00 | 1 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 window contains the following SQL code:

```
--8. Write an SQL query to list electronic gadgets and the number of times each product has been ordered.
SELECT Products.ProductName, COUNT(OrderDetails.ProductID) AS NumberOfOrders
FROM OrderDetails JOIN Products ON OrderDetails.ProductID = Products.ProductID GROUP BY Products.ProductName;

--9. Write an SQL query to find customers who have purchased a specific electronic gadget product.
-- Allow users to input the product name as a parameter.
--ALTER TABLE Orders DROP COLUMN Status;
--ALTER TABLE Orders ADD OrdersStatus VARCHAR(20);
--UPDATE Orders SET OrdersStatus='Pending';
--UPDATE Orders SET OrdersStatus='Shipped' WHERE OrderID=8 AND CustomerID=3;

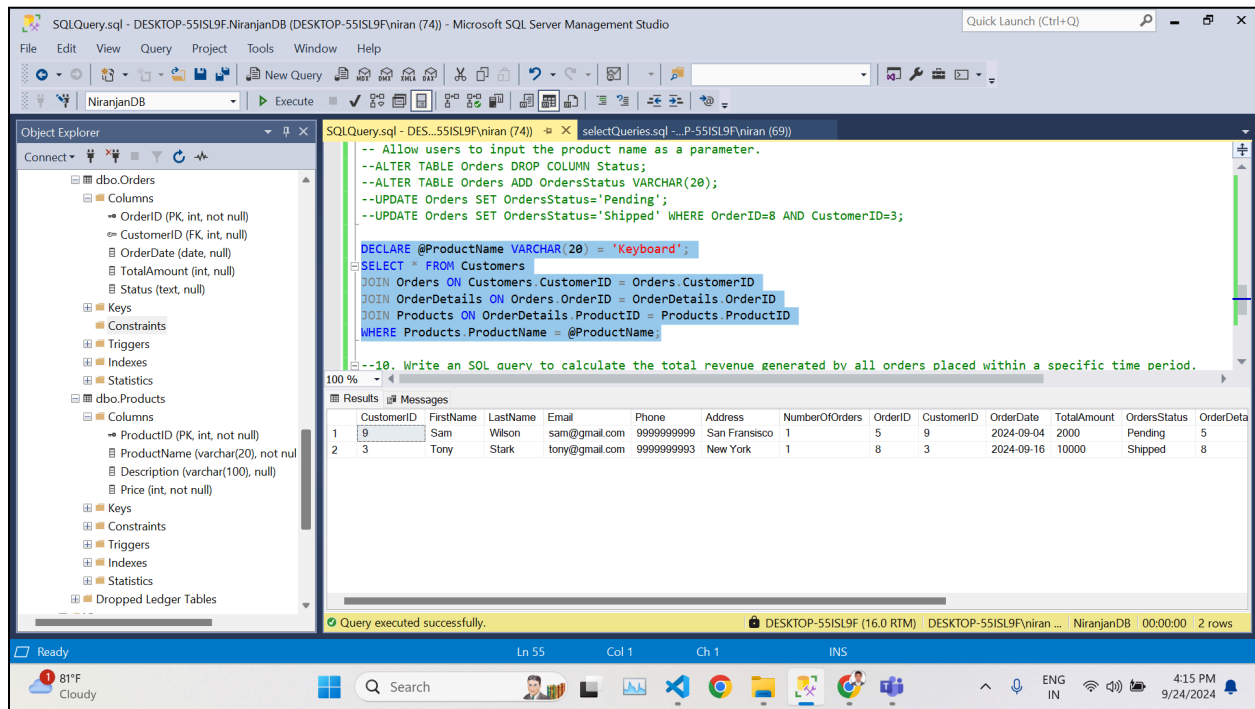
DECLARE @ProductName VARCHAR(20) = 'Keyboard';
SELECT * FROM Customers
```

The Results pane shows the output of the first query, displaying a list of products and their order counts:

ProductName	NumberOfOrders
1 Airpods	1
2 Charger	1
3 Earphones	1
4 Keyboard	2
5 Monitor	1
6 Powerbank	1
7 Router	1

The status bar at the bottom indicates 'Query executed successfully.' and 'DESKTOP-55ISL9F (16.0 RTM) | DESKTOP-55ISL9F\Niran... | NiranjanDB | 00:00:00 | 7 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 'dbo.Orders' and 'dbo.Products'. The main query window shows the following SQL code:

```
-- Allow users to input the product name as a parameter.
--ALTER TABLE Orders DROP COLUMN Status;
--ALTER TABLE Orders ADD OrdersStatus VARCHAR(20);
--UPDATE Orders SET OrdersStatus='Pending';
--UPDATE Orders SET OrdersStatus='Shipped' WHERE OrderID=8 AND CustomerID=3;

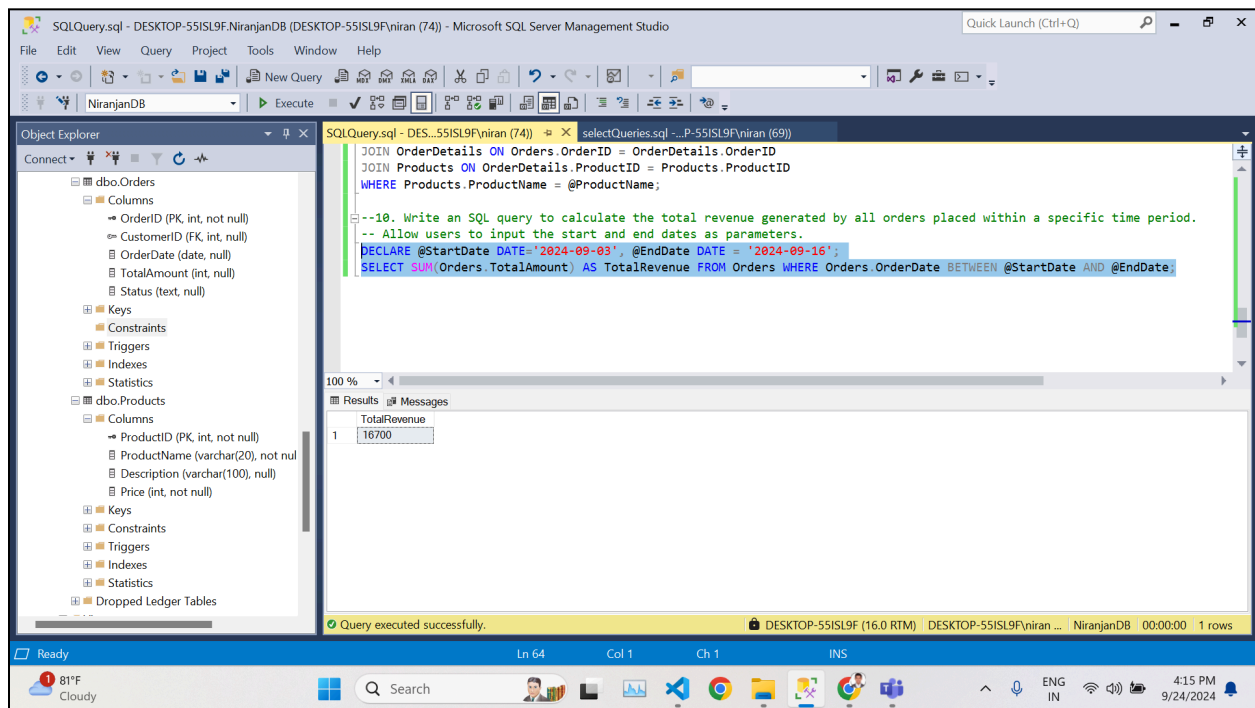
DECLARE @ProductName VARCHAR(20) = 'Keyboard';
SELECT * FROM Customers
JOIN Orders ON Customers.CustomerID = Orders.CustomerID
JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
JOIN Products ON OrderDetails.ProductID = Products.ProductID
WHERE Products.ProductName = @ProductName;
```

The Results pane shows the output of the query, displaying a list of orders with columns: CustomerID, FirstName, LastName, Email, Phone, Address, NumberOfOrders, OrderID, CustomerID, OrderDate, TotalAmount, OrdersStatus, and OrderData. The results are as follows:

CustomerID	FirstName	LastName	Email	Phone	Address	NumberOfOrders	OrderID	CustomerID	OrderDate	TotalAmount	OrdersStatus	OrderData
1	Sam	Wilson	sam@gmail.com	9999999999	San Francisco	1	5	9	2024-09-04	2000	Pending	5
2	Tony	Stark	tony@gmail.com	9999999993	New York	1	8	3	2024-09-16	10000	Shipped	8

The status bar at the bottom indicates 'Query executed successfully.' and 'DESKTOP-55ISL9F (16.0 RTM)'. The taskbar shows the system clock as 4:15 PM on 9/24/2024.

Output 10:



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

```
JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
JOIN Products ON OrderDetails.ProductID = Products.ProductID
WHERE Products.ProductName = @ProductName;

--10. Write an SQL query to calculate the total revenue generated by all orders placed within a specific time period.
-- Allow users to input the start and end dates as parameters.
DECLARE @StartDate DATE='2024-09-03', @EndDate DATE = '2024-09-16';
SELECT SUM(Orders.TotalAmount) AS TotalRevenue FROM Orders WHERE Orders.OrderDate BETWEEN @StartDate AND @EndDate;
```

The Results pane shows the output of the query, displaying a single row with the column 'TotalRevenue' and the value '16700'.

TotalRevenue
16700

The status bar at the bottom indicates 'Query executed successfully.' and 'DESKTOP-55ISL9F (16.0 RTM)'. The taskbar shows the system clock as 4:15 PM on 9/24/2024.