



# **NEPAL COLLEGE OF INFORMATION AND TECHNOLOGY**

**BALKUMARI, LALITPUR**

## **DBMS**

### **Assignment -5**

# **BCA**

**Name:** Sudip dahal  
**Roll no:** 222039  
**Year:** 2022

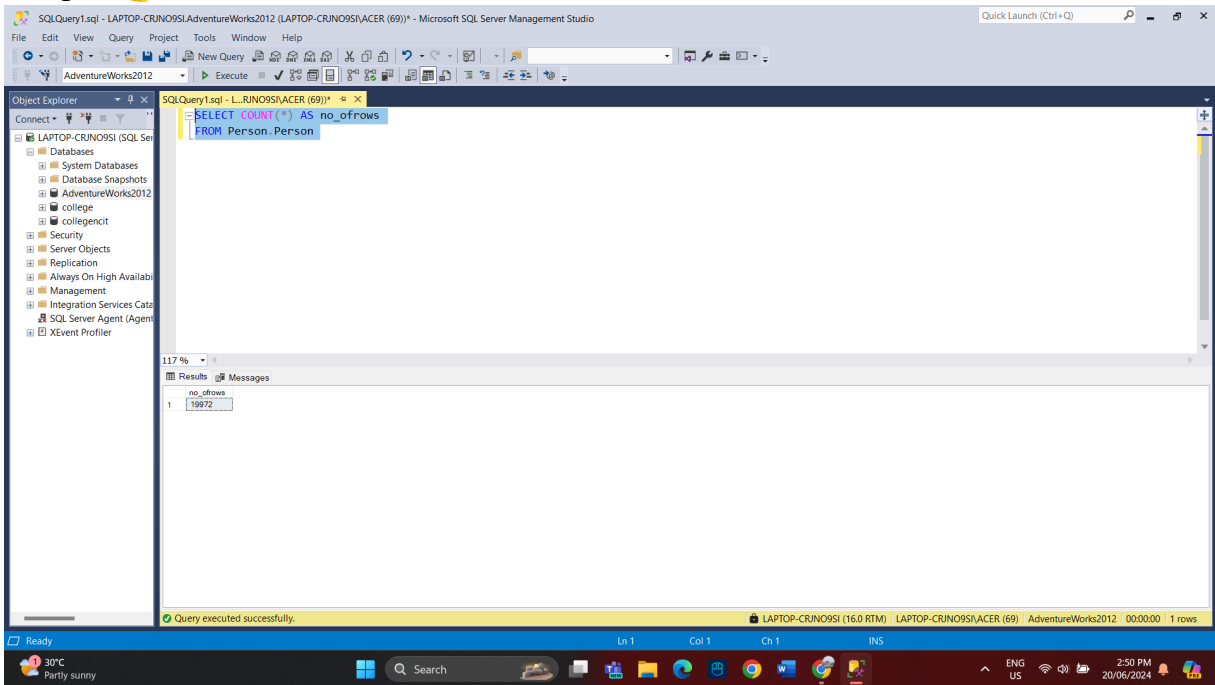
## Aggregate Function Practice Problems

1) How many rows are in the Person.Person table? Use an aggregate function NOT "SELECT \*".

### SOURCE CODE:

```
SELECT COUNT(*) AS no_ofrows  
FROM Person.Person
```

Out put 👍



The screenshot displays the Microsoft SQL Server Enterprise Manager interface. The 'Object Explorer' on the left shows the 'AdventureWorks2012' database. The 'SQLQuery1.sql' window in the center contains the query: `SELECT COUNT(*) AS no_ofrows FROM Person.Person`. The 'Results' pane at the bottom shows the output of the query, which is a single row with the value 19972 under the column 'no\_ofrows'. The status bar at the bottom indicates 'Query executed successfully' and '1 rows'.

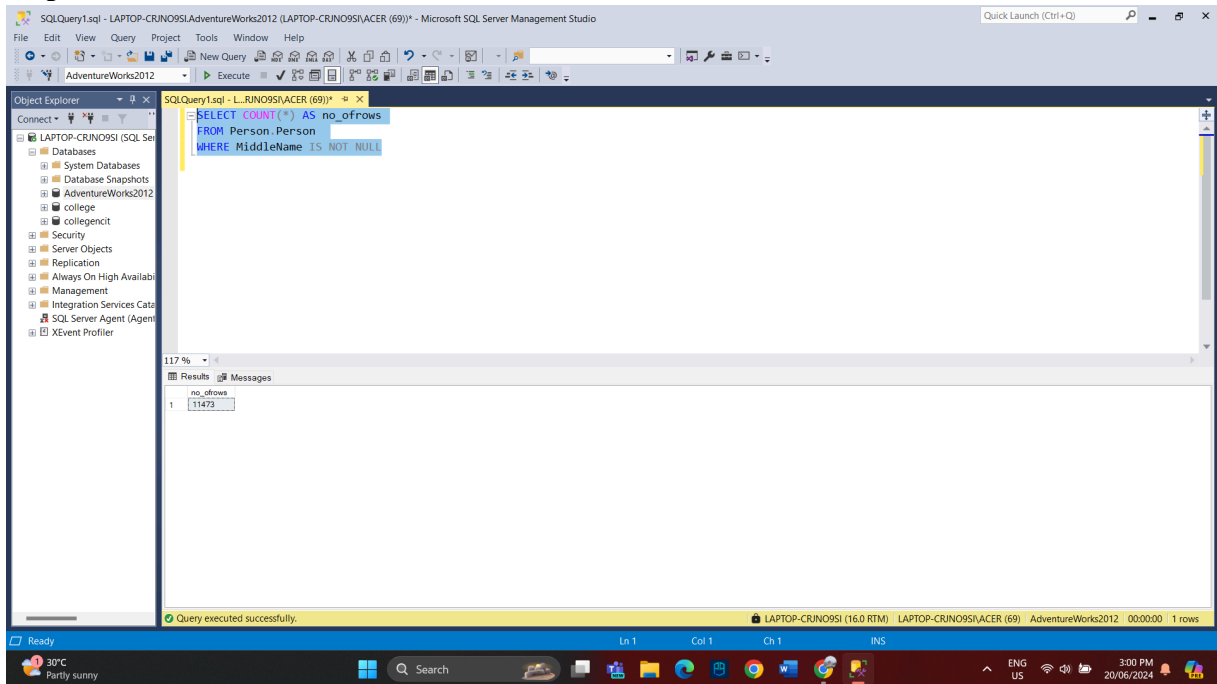
no_ofrows
19972

2) How many rows in the Person.Person table do not have a NULL value in the MiddleName column?

**Source code:**

```
SELECT COUNT(*) AS no_ofrows  
FROM Person.Person  
WHERE MiddleName IS NOT NULL
```

**output:**



3) What is the average StandardCost (located in Production.Product) for each product where the StandardCost is greater than \$0.00?

**Source code:**

```
SELECT ProductID, AVG(StandardCost) AS Avgstandardcost  
FROM Production.Product  
WHERE StandardCost > 0.00  
GROUP BY ProductID
```

output:

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL query:

```
SELECT ProductID, AVG(StandardCost) AS Avgstandardcost
FROM Production.Product
WHERE StandardCost > 0.00
GROUP BY ProductID
```

The query results are displayed in a table with the following data:

ProductID	AverageStandardCost
514	98.77
515	108.99
516	145.87
517	98.77
518	108.99
519	145.87
520	98.77
521	108.99
522	145.87
680	1059.31
706	1059.31
707	13.0863
708	13.0863
709	3.3963
710	3.3963
711	13.0863

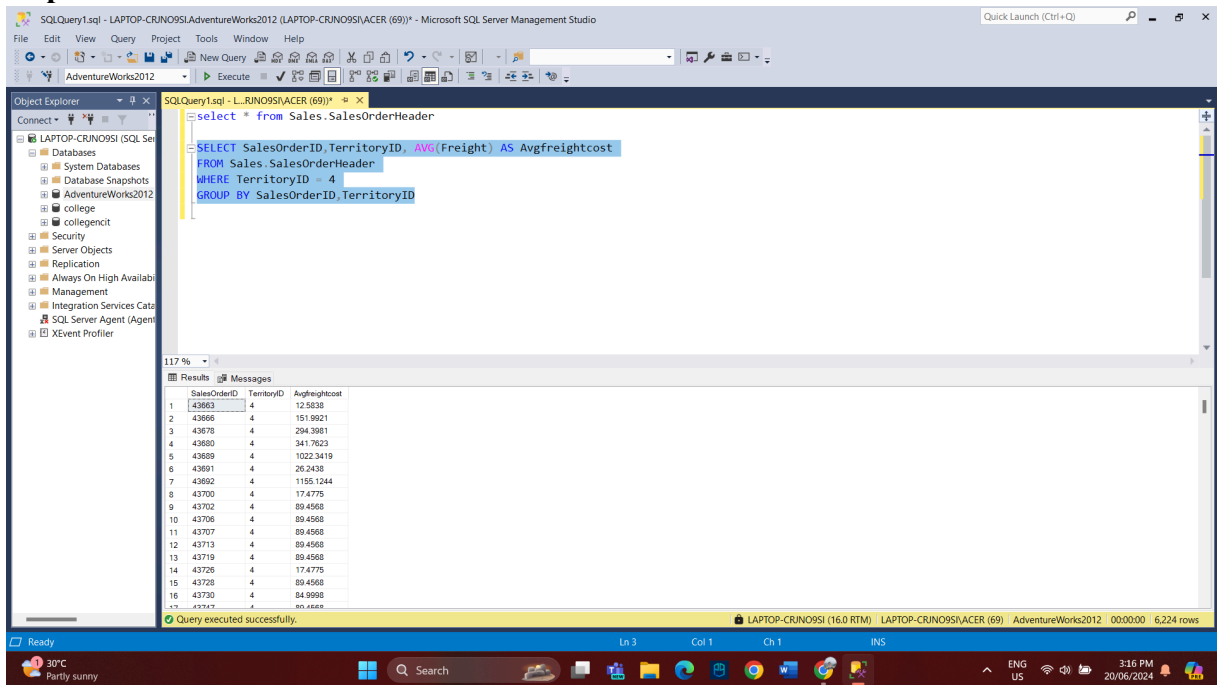
The status bar at the bottom indicates "Query executed successfully." and "304 rows".

4) What is the average Freight amount for each sale (found in Sales.Sales OrderHeader) where the sale took place in TerritoryID 4?

**Source code:**

```
SELECT SalesOrderID, TerritoryID, AVG(Freight) AS Avgfreightcost
FROM Sales.SalesOrderHeader
WHERE TerritoryID = 4
GROUP BY SalesOrderID, TerritoryID
```

output:



The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
select * from Sales.SalesOrderHeader
SELECT SalesOrderID, TerritoryID, AVG(Freight) AS Avgfreightcost
FROM Sales.SalesOrderHeader
WHERE TerritoryID = 4
GROUP BY SalesOrderID, TerritoryID
```

The Results pane displays the following data:

SalesOrderID	TerritoryID	Avgfreightcost
43663	4	12.5838
43666	4	151.9921
43678	4	294.3081
43680	4	341.7623
43689	4	1022.3419
43691	4	26.2438
43692	4	1155.1244
43700	4	17.4775
43702	4	89.4568
43706	4	89.4568
43707	4	89.4568
43713	4	89.4568
43719	4	89.4568
43726	4	17.4775
43728	4	89.4568
43730	4	84.9990
43742	4	89.4568

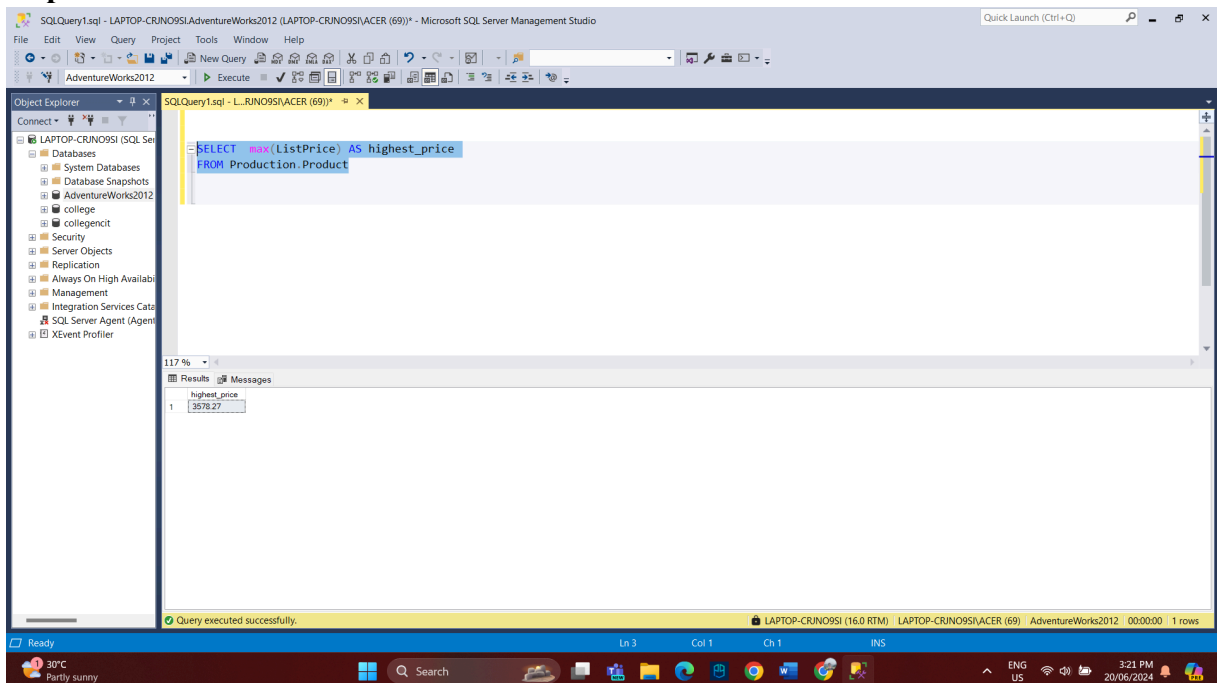
The status bar at the bottom indicates "Query executed successfully." and "6,224 rows".

5) How expensive is the most expensive product, by ListPrice, in the table Production.Product?

Source code:

```
SELECT max(ListPrice) AS highest_price
FROM Production.Product
```

output:



The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
SELECT max(ListPrice) AS highest_price
FROM Production.Product
```

The Results pane displays the following data:

highest_price
3576.27

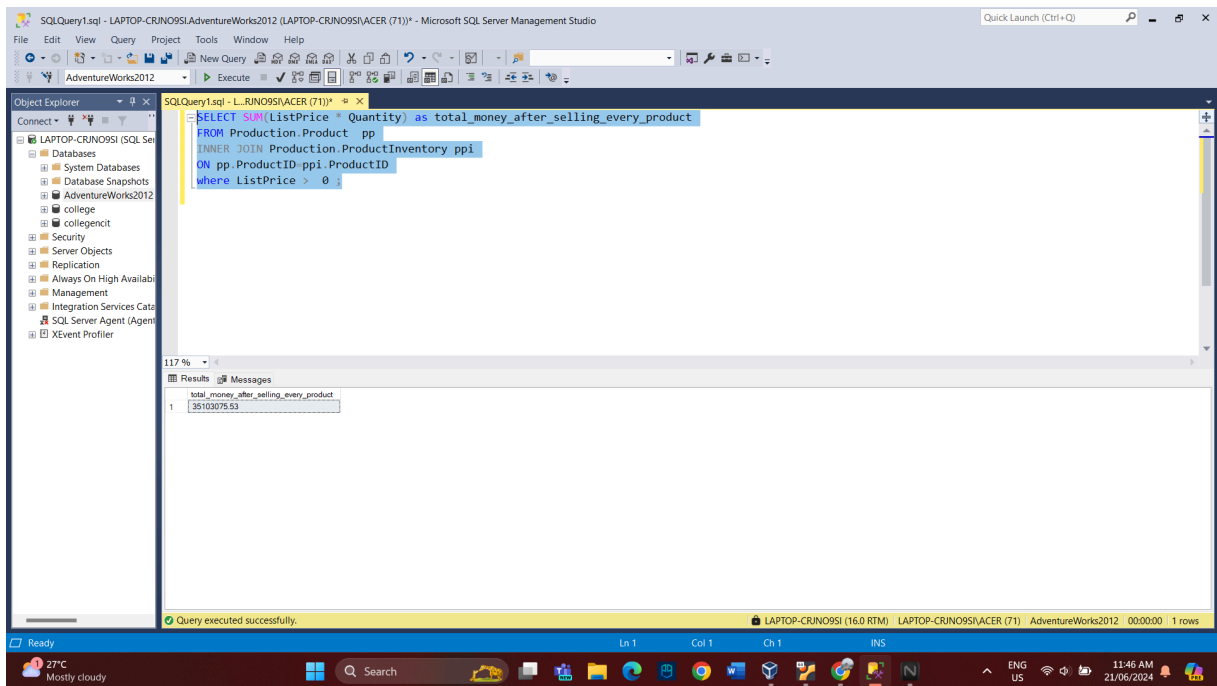
The status bar at the bottom indicates "Query executed successfully." and "1 rows".

6) Join the Production.Product table and the Production.ProductInventory table for only the products that appear in both table. Use the ProductID as the joining column. Production.ProductInventory contains the quantity of each product (several rows can appear for each product to indicate the product appears in multiple locations). Your goal is to determine how much money we would earn if we sold every product for its list price for each product with a ListPrice greater than \$0. That is, if you summed the product of each product's inventory by its list price, what would that value be? (Hint: This is intentionally challenging. You must use an aggregate function with a mathematical expression to accomplish your goal)

**Source code:**

```
SELECT SUM(ListPrice * Quantity) as total_money_after_selling_every_product  
FROM Production.Product pp  
INNER JOIN Production.ProductInventory ppi  
ON pp.ProductID=ppi.ProductID  
where ListPrice > 0 ;
```

**Output:**



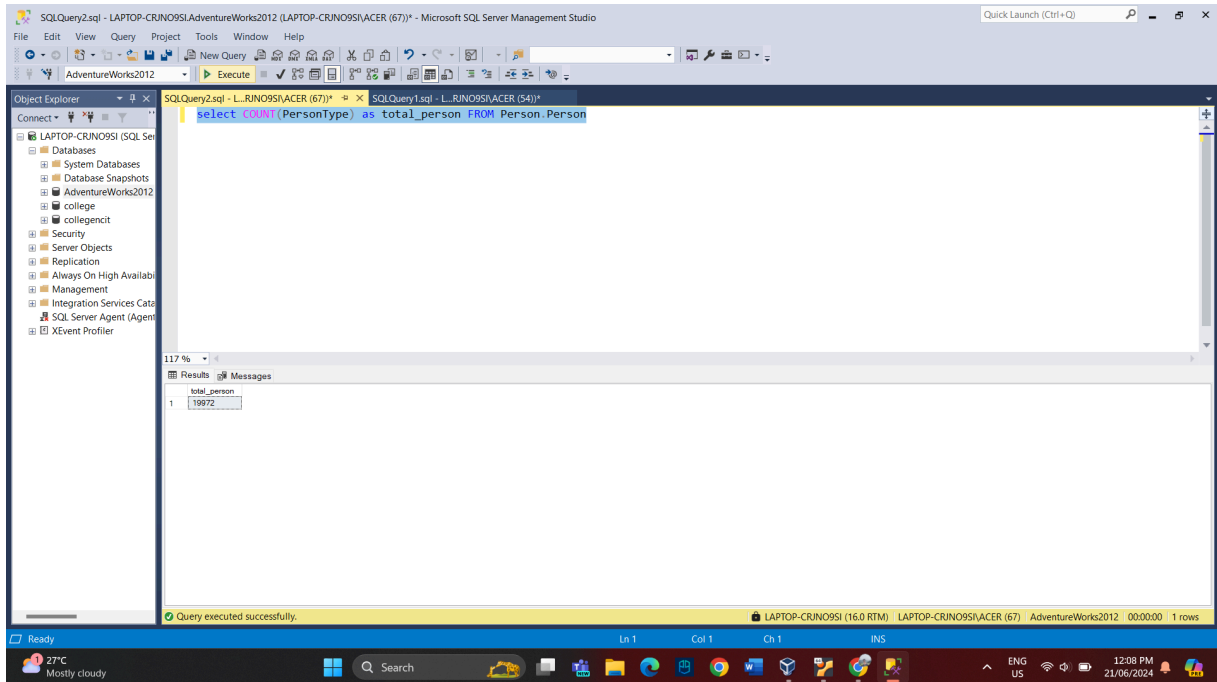
**GROUP BY Clause Practice Problems**

1) In the Person. Person table, how many people are associated with each PersonType?

**Source code:**

```
select COUNT(PersonType) as total_person FROM Person.Person
```

**Output:**

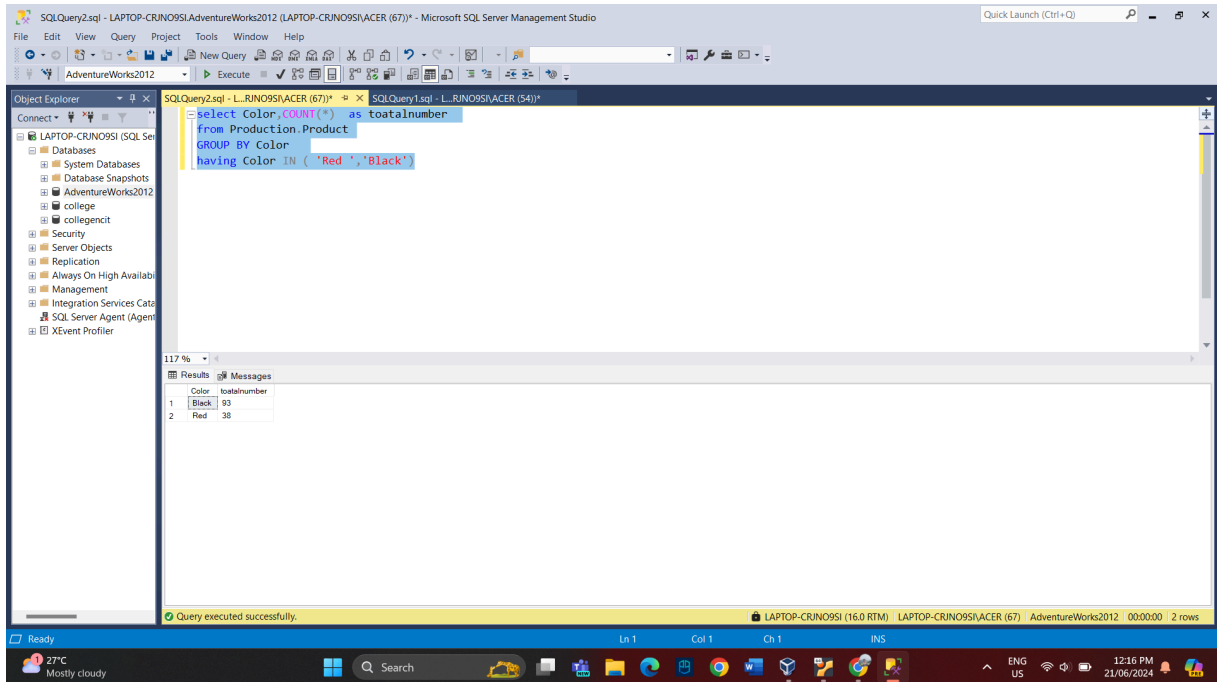


2) Using only one query, find out how many products in Production.Product are the color "red" and how many are "black".

**Source code:**

```
select Color,COUNT(*) as toatalnumber  
from Production.Product  
GROUP BY Color  
having Color IN ( 'Red ','Black')
```

**Output:**



3) Using Sales. Sales OrderHeader, how many sales occurred in each territory between July 1, 2005 and December 31, 2006? Order the results by the sale count in descending order.

**Source code:**

**SELECT**

**TerritoryID,**

**COUNT(\*) AS SaleCount**

**FROM**

**Sales.SalesOrderHeader**

**WHERE**

**ShipDate BETWEEN '2011-07-01' AND '2012-12-31'**

**GROUP BY**

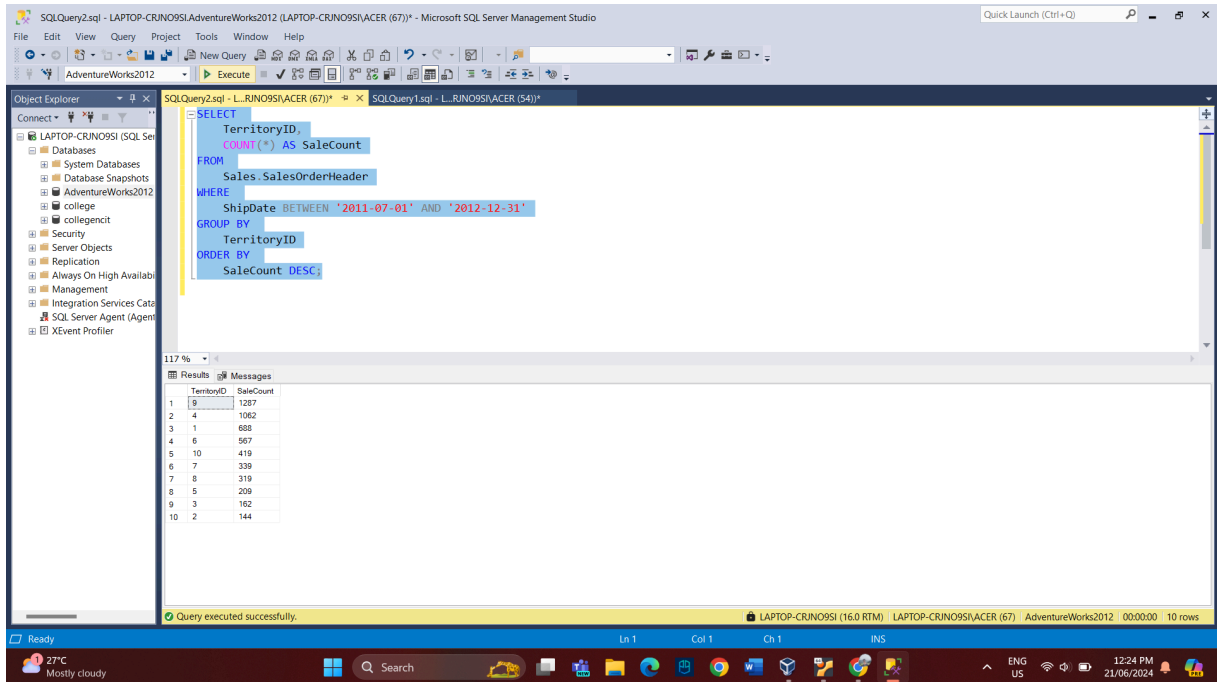
**TerritoryID**

**ORDER BY**

**SaleCount DESC;**

**Output:**





4) Expanding on the previous example, group the results not by the TerritoryID but by the name of the territory (found in the Sales.Sales Territory table).

**Source code:**

**SELECT**

**st.Name AS territory\_name,**

**COUNT(\*) AS SaleCount**

**FROM**

**Sales.SalesOrderHeader so**

**INNER JOIN**

**Sales.SalesTerritory st**

**ON**

**so.TerritoryID=st.TerritoryID**

**WHERE**

**ShipDate BETWEEN '2011-07-01' AND '2012-12-31'**

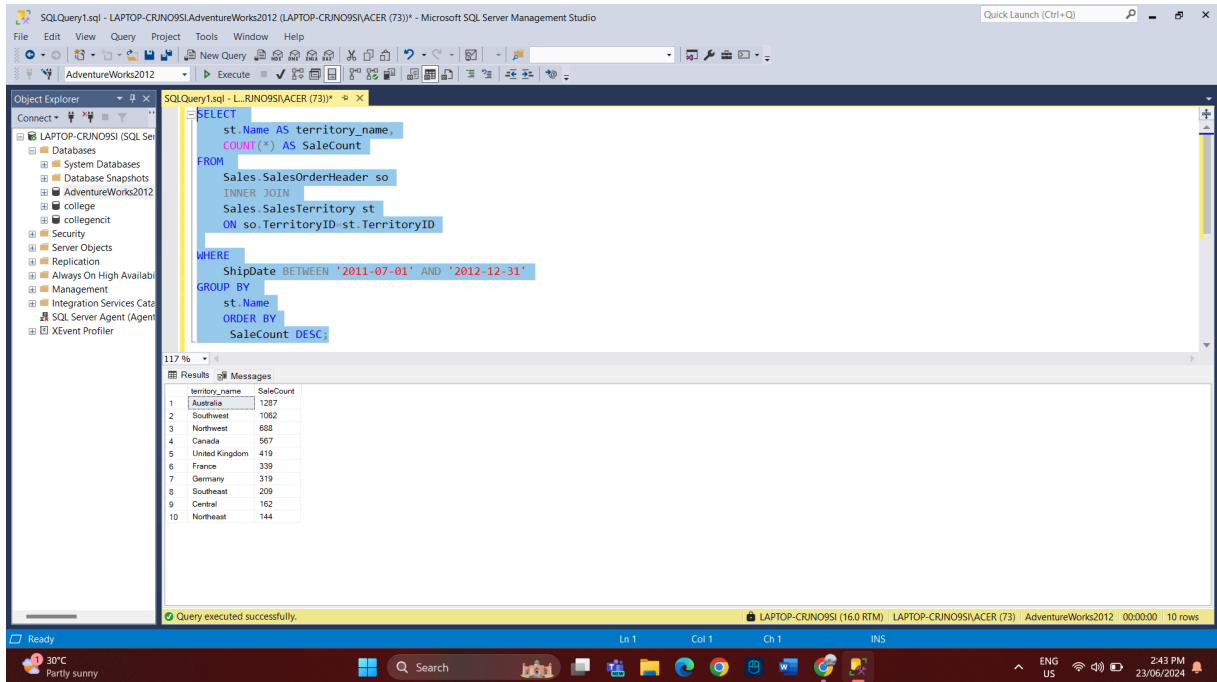
**GROUP BY**

**st.Name**

**ORDER BY**

**SaleCount DESC;**

**Output:**



5) Using the Book, BookAuthor, Author and/or Publisher tables, identify how many books each author either wrote or co-authored.

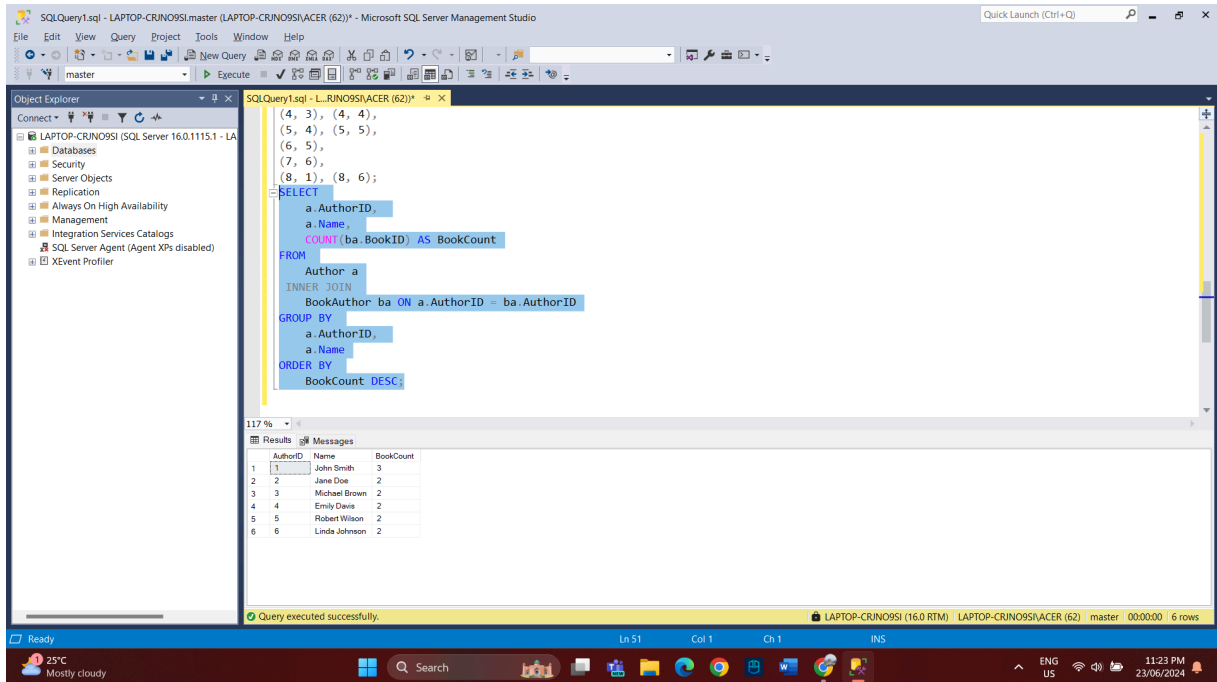
**Source code:**

```

SELECT
    a.AuthorID,
    a.Name,
    COUNT(ba.BookID) AS BookCount
FROM
    Author a
    INNER JOIN
    BookAuthor ba ON a.AuthorID = ba.AuthorID
GROUP BY
    a.AuthorID,
    a.Name
ORDER BY
    BookCount DESC;

```

**Output:**



## HAVING Clause Practice Problems

1

) Find the total sales by territory for all rows in the Sales.Sales OrderHeader table. Return only those territories that have exceeded \$10 million in historical sales. Return the total sales and the TerritoryID column.

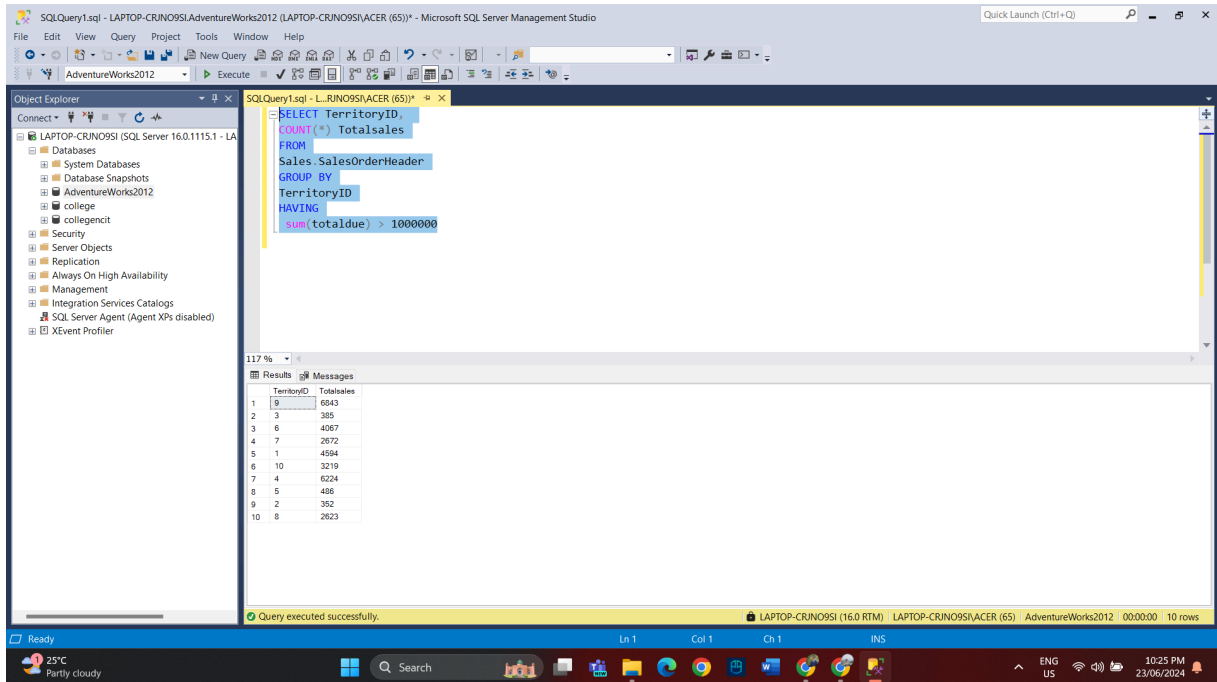
**Source code:**

```

SELECT TerritoryID,
COUNT(*) Totalsales
FROM
Sales.SalesOrderHeader
GROUP BY
TerritoryID
HAVING
sum(totaldue) > 1000000

```

**Output:**



2) Using the query from the previous question, join to the Sales. Sales Territory table and replace the TerritoryID column with the territory's name.

**Source code:** `SELECT st.Name as TerritoryName, COUNT(*) Totalsales`  
**FROM**  
**Sales.SalesOrderHeader soh**  
**inner join**  
**Sales.SalesTerritory st**  
**ON**  
**soh.TerritoryID = st.TerritoryID**  
**GROUP BY**  
**st.Name**  
**HAVING**  
**sum(TotalDue) > 1000000**

**Output:**

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL query:

```

SELECT st.Name as TerritoryName, COUNT(*) Totalsales
FROM
Sales.SalesOrderHeader soh
INNER JOIN
Sales.SalesTerritory st
ON
soh.TerritoryID = st.TerritoryID
GROUP BY
st.Name
HAVING
SUM(TotalDue) > 1000000

```

The Results pane shows the following data:

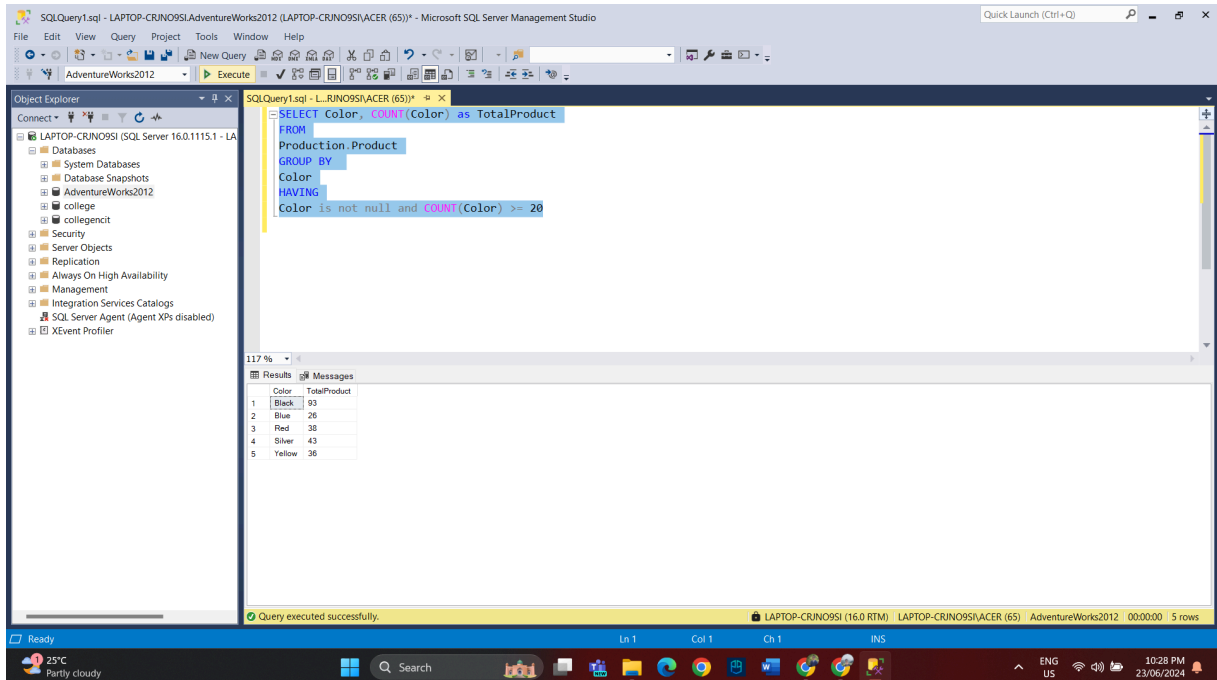
	TerritoryName	Totalsales
1	Australia	6843
2	Central	385
3	Canada	4067
4	France	2672
5	Northwest	4594
6	United Kingdom	3219
7	Southwest	6224
8	Southeast	486
9	Northeast	352
10	Germany	2623

The status bar at the bottom indicates "Query executed successfully." and "10 rows".

3) Using the Production.Product table, find how many products are associated with each color. Ignore all rows where the color has a NULL value. Once grouped, return to the results only those colors that had at least 20 products with that color.

**Source code:** **SELECT Color, COUNT(Color) as TotalProduct**  
**FROM**  
**Production.Product**  
**GROUP BY**  
**Color**  
**HAVING**  
**Color is not null and COUNT(Color) >= 20**

**Output:**



4) Starting with the Sales.SalesOrderHeader table, join to the Sales.SalesOrderDetail table. This table contains the line item details associated with each sale. From Sales.SalesOrderDetail, join to the Production.Product table. Return the Name column from Production.Product and assign it the column alias "Product Name". For each product, find out how many of each product was ordered for all orders that occurred in 2006. Only output those products where at least 200 were ordered.

**Source code:**

```

SELECT
p.Name 'Product Name', COUNT(sod.OrderQty) TotalOrder
FROM
Sales.SalesOrderHeader soh
inner join
Sales.SalesOrderDetail sod
ON
soh.SalesOrderID = sod.SalesOrderID
inner join
Production.Product p
ON
sod.ProductID = p.ProductID
WHERE
soh.OrderDate between '2012-01-01' and '2012-12-31'
GROUP BY
p.Name
HAVING
COUNT(sod.OrderQty) >= 200

```

## Output:

The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Object Explorer' with the 'AdventureWorks2012' database selected. The right pane shows a SQL query in the 'SQL Query1.sql' window. The query is as follows:

```
SELECT soh.SalesOrderHeader soh
inner join
Sales.SalesOrderDetail sod
ON
soh.SalesOrderID = sod.SalesOrderID
inner join
Production.Product p
ON
sod.ProductID = p.ProductID
WHERE
soh.OrderDate between '2012-01-01' and '2012-12-31'
GROUP BY
p.Name
HAVING
COUNT(sod.OrderQty) >= 200
```

Below the query, the 'Results' pane shows the output of the query. The results are as follows:

Color	TotalProduct
1 Black	93
2 Blue	26
3 Red	38
4 Silver	43
5 Yellow	36

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

5) Find the first and last name of each customer who has placed at least 6 orders between July 1, 2005 and December 31, 2006. Order your results by the number of orders placed in descending order. (Hint: You will need to join to three tables Sales.Sales OrderHeader, Sales.Customer, and Person.Person. You will use every clause to complete this query).

**Source code:**

```
SELECT
p.FirstName,
p.LastName,
COUNT(soh.SalesOrderID) TotalOrder
FROM
Sales.SalesOrderHeader soh
inner join
Sales.Customer
ON
soh.CustomerID = c.CustomerID
inner join
Person.Person p
ON
c.PersonID = p.BusinessEntityID
WHERE
soh.OrderDate between '2012-07-01' and '2013-12-31'
GROUP BY
p.FirstName,
p.LastName
HAVING
COUNT(soh.SalesOrderID) >= 6
ORDER BY
```

# TotalOrder DESC

## Output:

