Buildables Week 1

Github Repo: https://github.com/afnank070/Data-Engineering-Buildables-Fellowship

Day 1 – Foundation (SELECT, WHERE, ORDER BY)

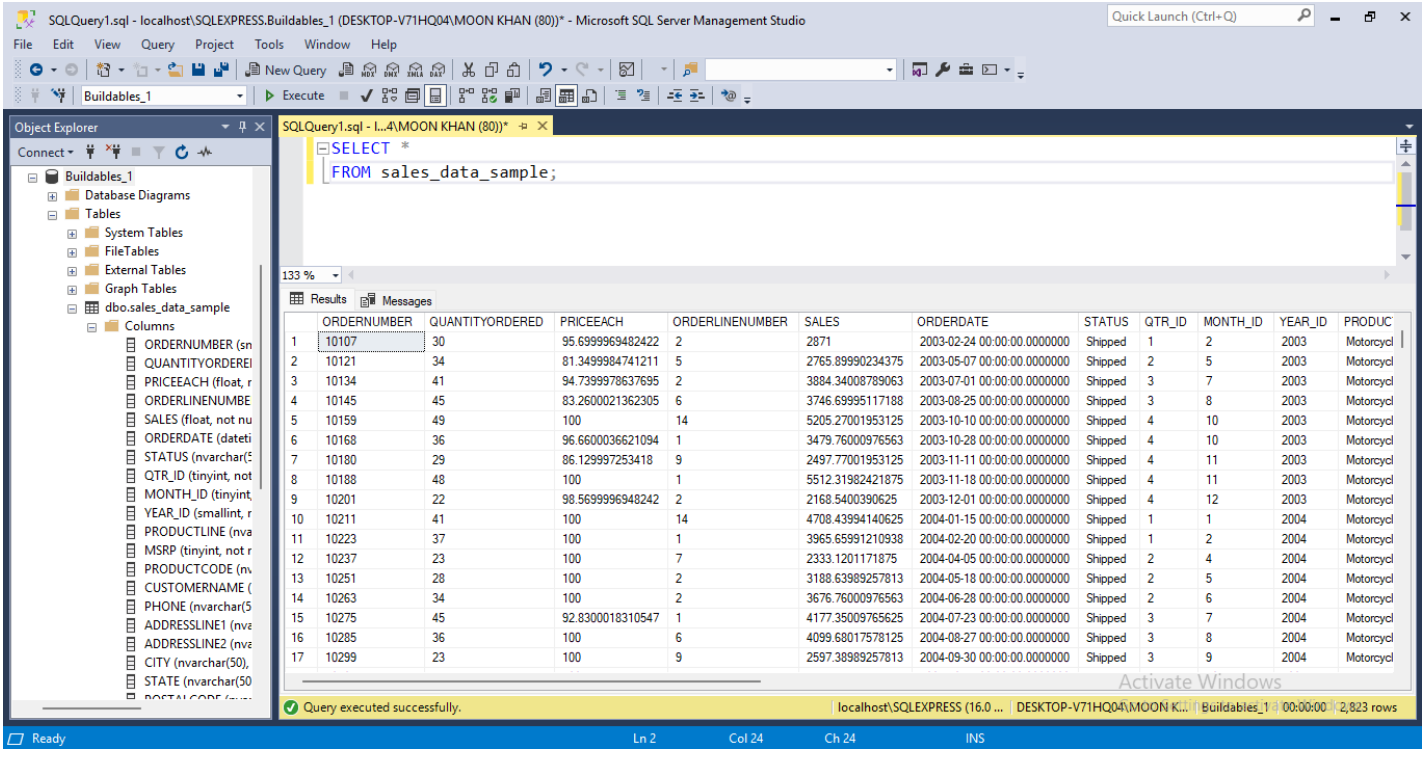
Query 1:

- **Purpose:** See full sales dataset (raw form).
- **Concepts Used:** SELECT
- **Code:**

```
SELECT *

FROM sales_data_sample;
```

- **Expected Output:** Entire table with all columns.
- **Business Insight:** Gives complete picture of sales records.

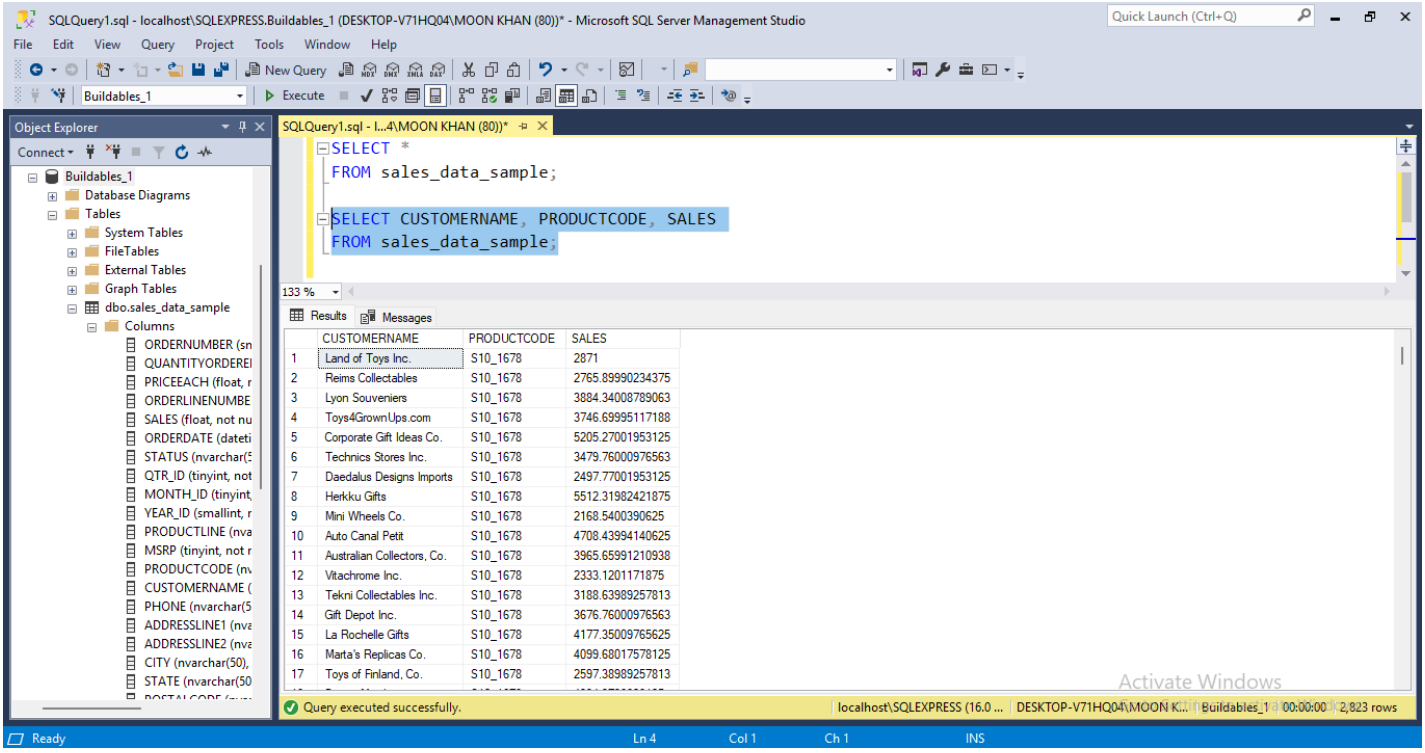


Query 2:

- **Purpose:** View only customer details (name, city, country).
- **Concepts Used:** SELECT
- **Code:**

```
SELECT CUSTOMERNAME, CITY, COUNTRY
FROM sales_data_sample;
```

- **Expected Output:** Customer name + city + country list.
- **Business Insight:** Quick way to focus only on customer information.

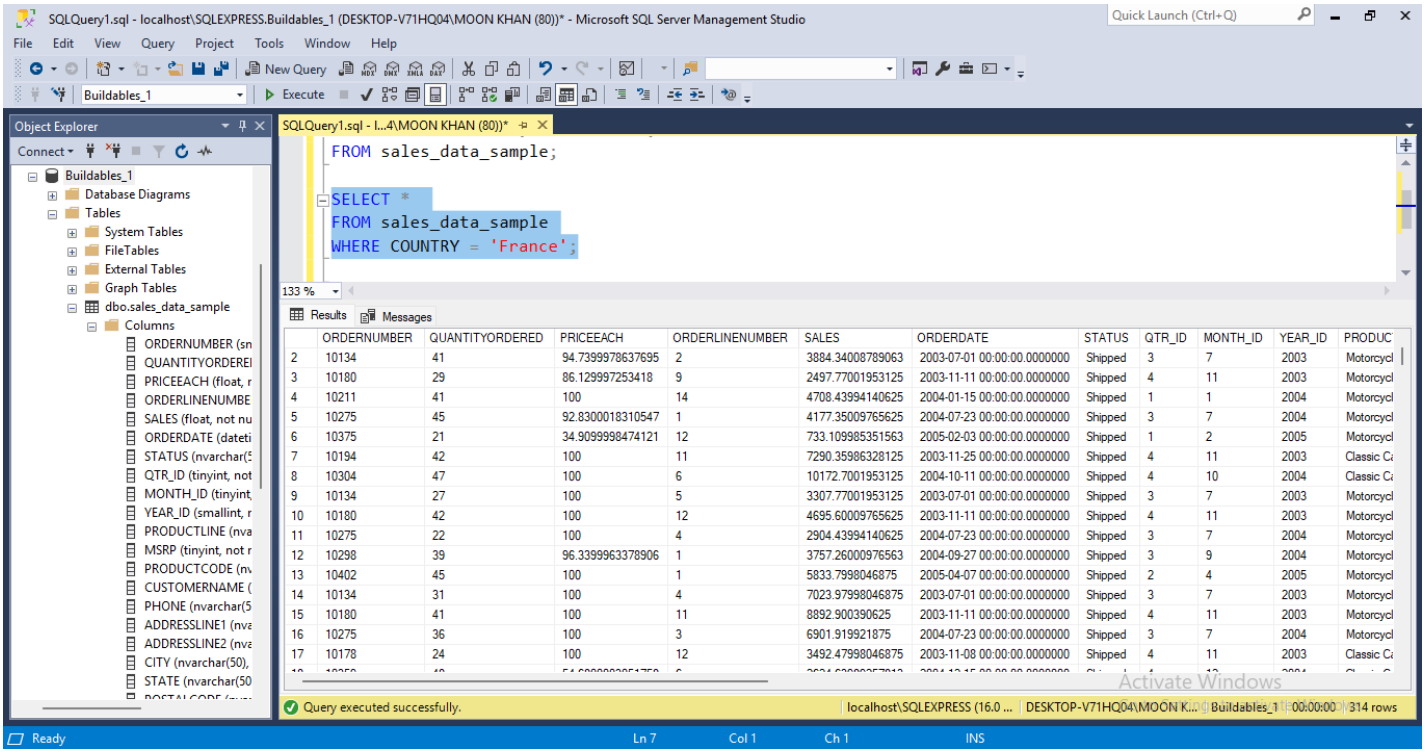


Query 3:

- **Purpose:** Find all orders from France.
- **Concepts Used:** SELECT, WHERE
- **Code:**

```
SELECT ORDERNUMBER, CUSTOMERNAME, COUNTRY
FROM sales_data_sample
WHERE COUNTRY = 'France';
```

- **Expected Output:** Orders placed by France customers.
- **Business Insight:** Identifies region-specific sales.

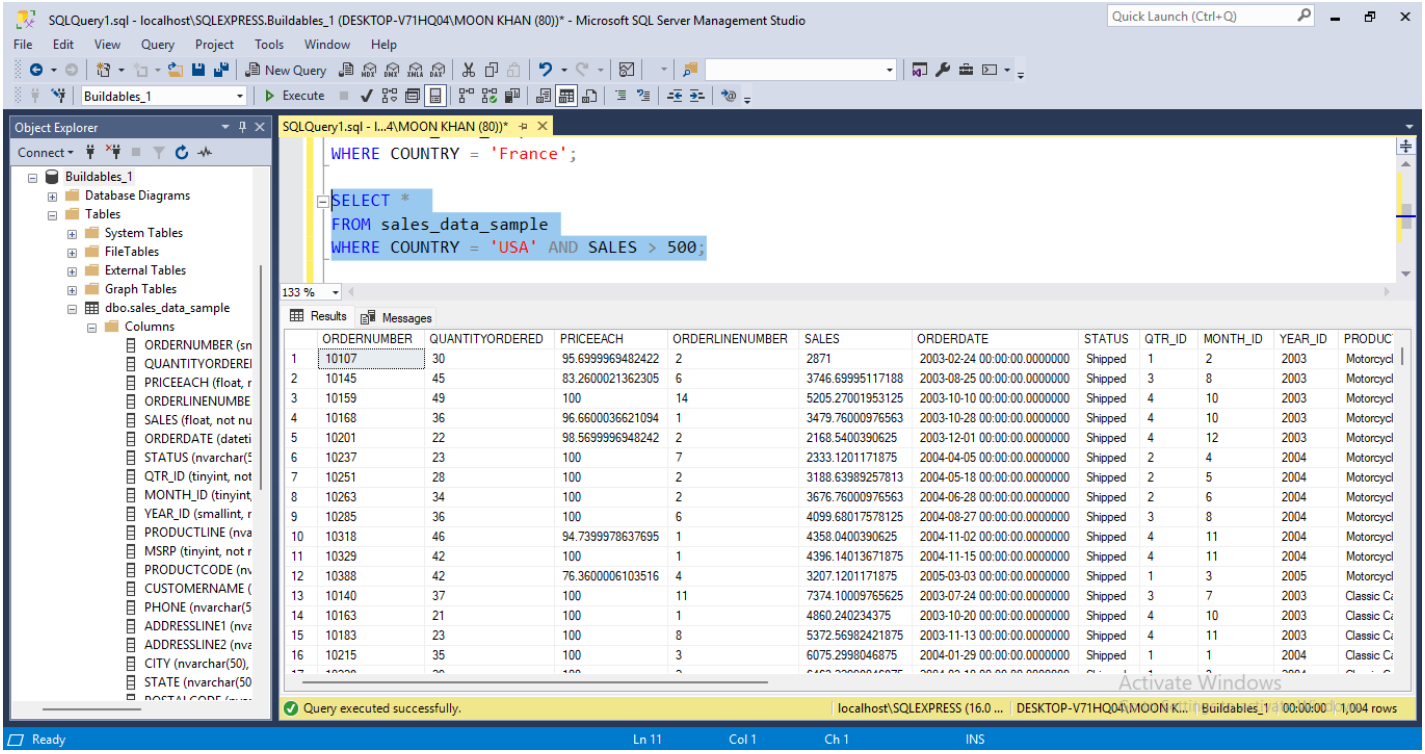


Query 4:

- **Purpose:** Customers from France with sales > 5000.
- **Concepts Used:** SELECT, WHERE, AND
- **Code:**

```
SELECT CUSTOMERNAME, COUNTRY, SALES
FROM sales_data_sample
WHERE COUNTRY = 'France' AND SALES > 5000;
```

- **Expected Output:** High-value French customers.
- **Business Insight:** Useful for targeted marketing.



Query 5:

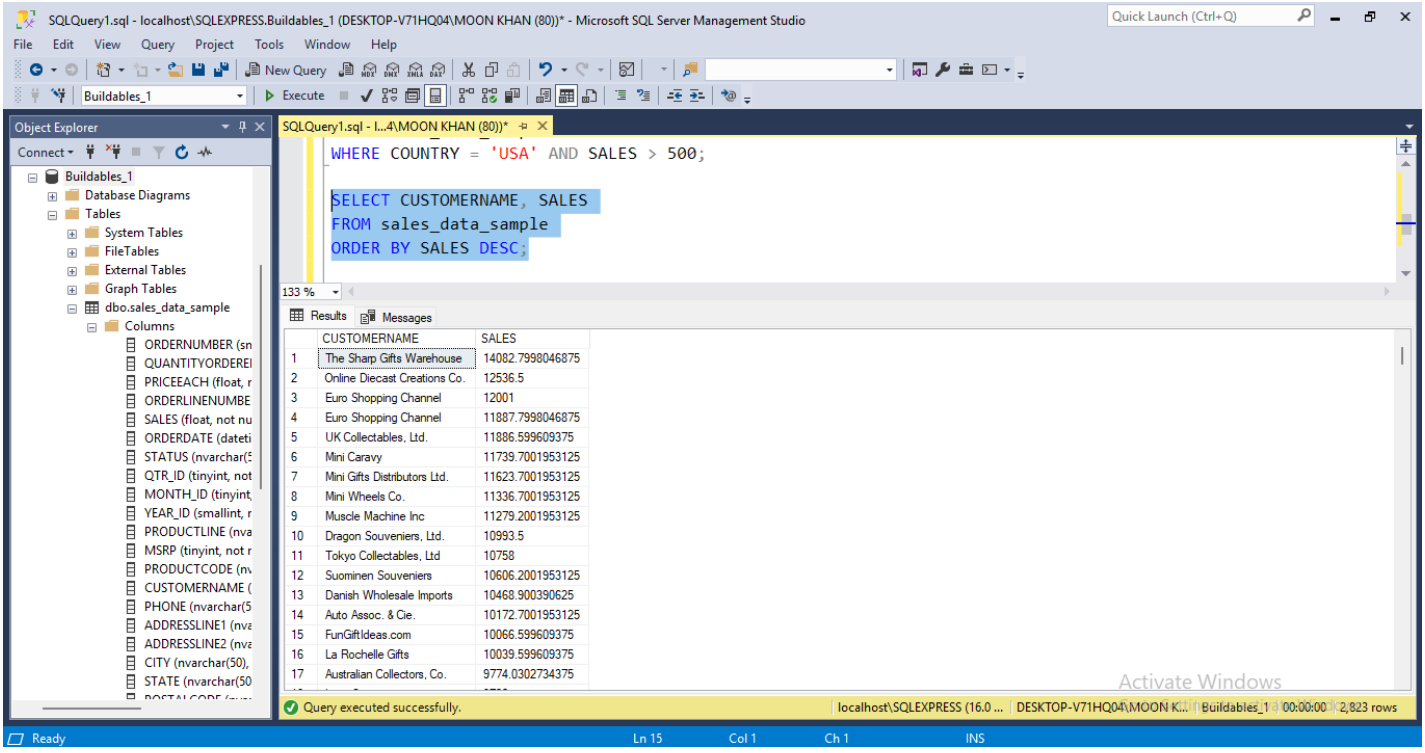
- **Purpose:** Sort orders by most recent first.
- **Concepts Used:** ORDER BY
- **Code:**

SELECT ORDERNUMBER, ORDERDATE, STATUS, SALES

FROM sales_data_sample

ORDER BY ORDERDATE DESC;

- **Expected Output:** Orders arranged newest → oldest.
- **Business Insight:** Helps monitor latest transactions.



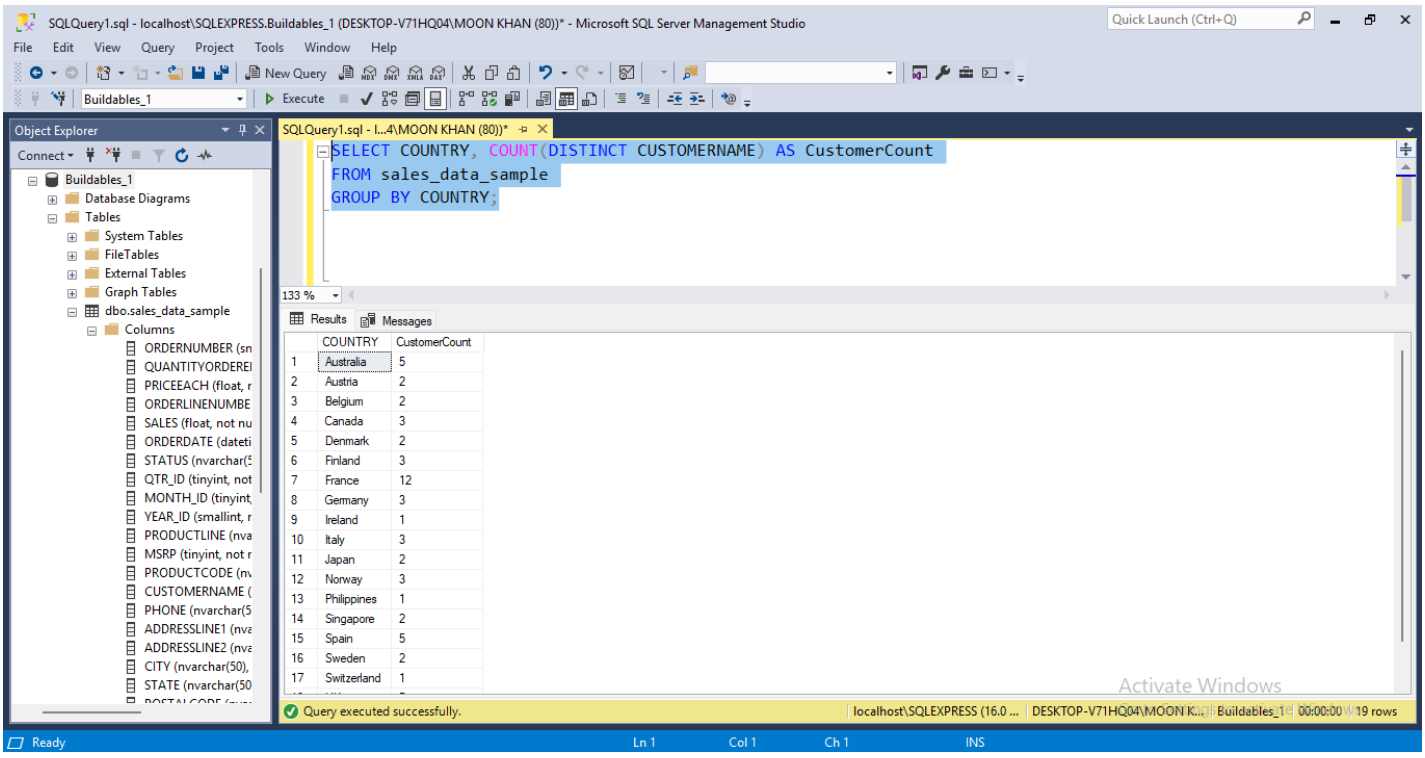
Day 2 – Data Aggregation (GROUP BY, HAVING, Aggregates)

Query 6:

- **Purpose:** Count customers by country.
- **Concepts Used:** GROUP BY, COUNT
- **Code:**

```
SELECT COUNTRY, COUNT(DISTINCT CUSTOMERNAME) AS total_customers
FROM sales_data_sample
GROUP BY COUNTRY;
```

- **Expected Output:** Number of unique customers per country.
- **Business Insight:** Identifies strong vs. weak markets.

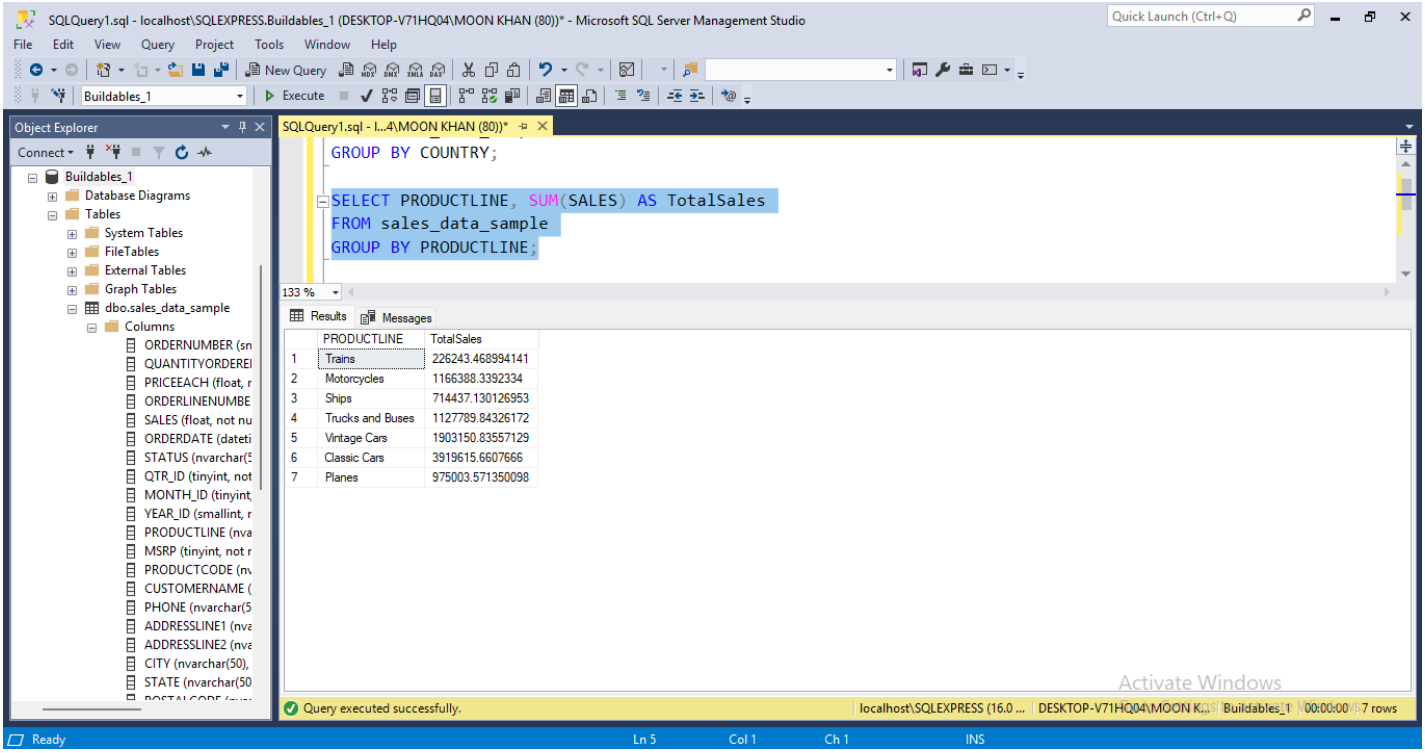


Query 7:

- **Purpose:** Total sales by product line.
- **Concepts Used:** SUM, GROUP BY
- **Code:**

```
SELECT PRODUCTLINE, SUM(SALES) AS total_sales
FROM sales_data_sample
GROUP BY PRODUCTLINE;
```

- **Expected Output:** Sales value grouped by product category.
- **Business Insight:** Shows which product line earns most revenue.

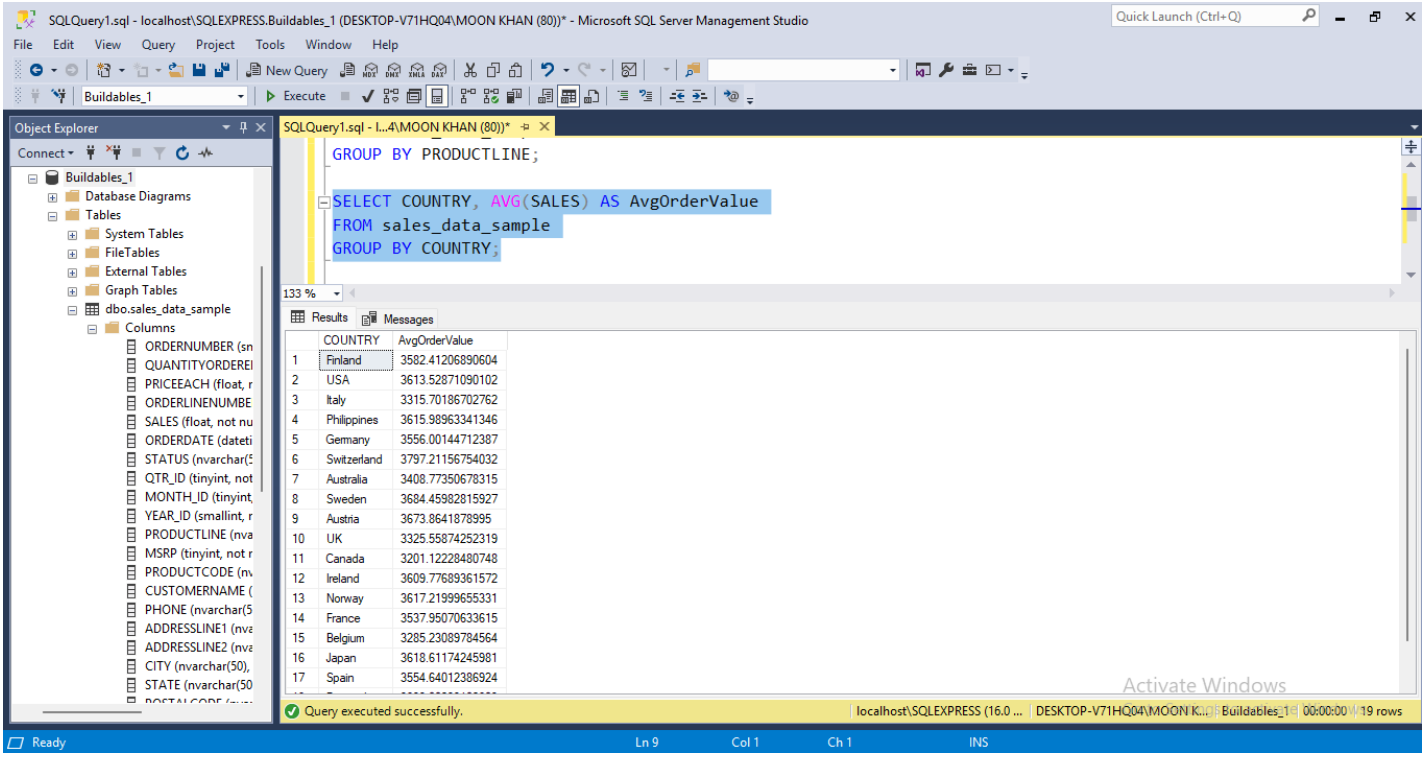


Query 8:

- **Purpose:** Average sales per customer.
- **Concepts Used:** AVG, GROUP BY
- **Code:**

```
SELECT CUSTOMERNAME, AVG(SALES) AS avg_order_value
FROM sales_data_sample
GROUP BY CUSTOMERNAME;
```

- **Expected Output:** Customer-wise average sales.
- **Business Insight:** Detects loyal vs. low-value customers.

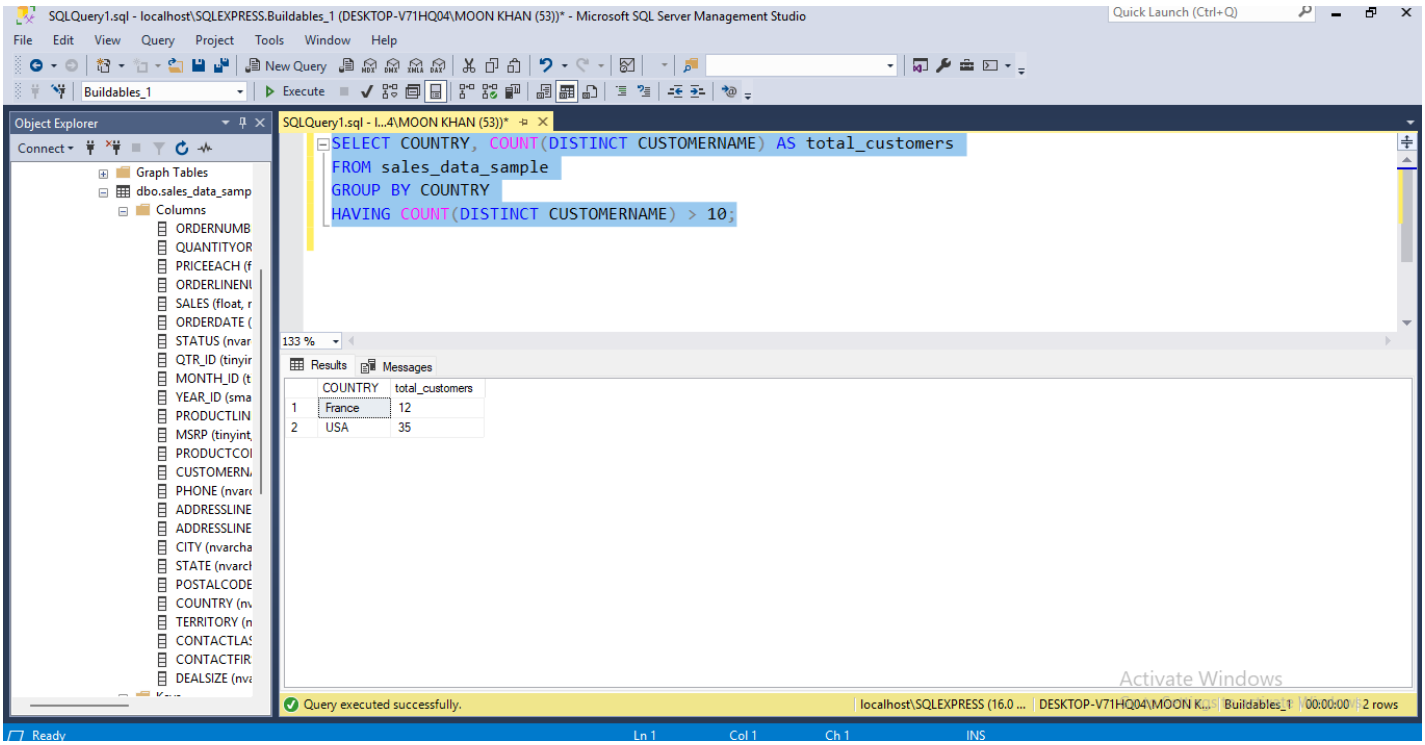


Query 9:

- **Purpose:** Show only countries with > 10 customers.
- **Concepts Used:** GROUP BY, HAVING
- **Code:**

```
SELECT COUNTRY, COUNT(DISTINCT CUSTOMERNAME) AS total_customers
FROM sales_data_sample
GROUP BY COUNTRY
HAVING COUNT(DISTINCT CUSTOMERNAME) > 10;
```

- **Expected Output:** Countries with customer base > 10.
- **Business Insight:** Identifies strong regions for expansion.

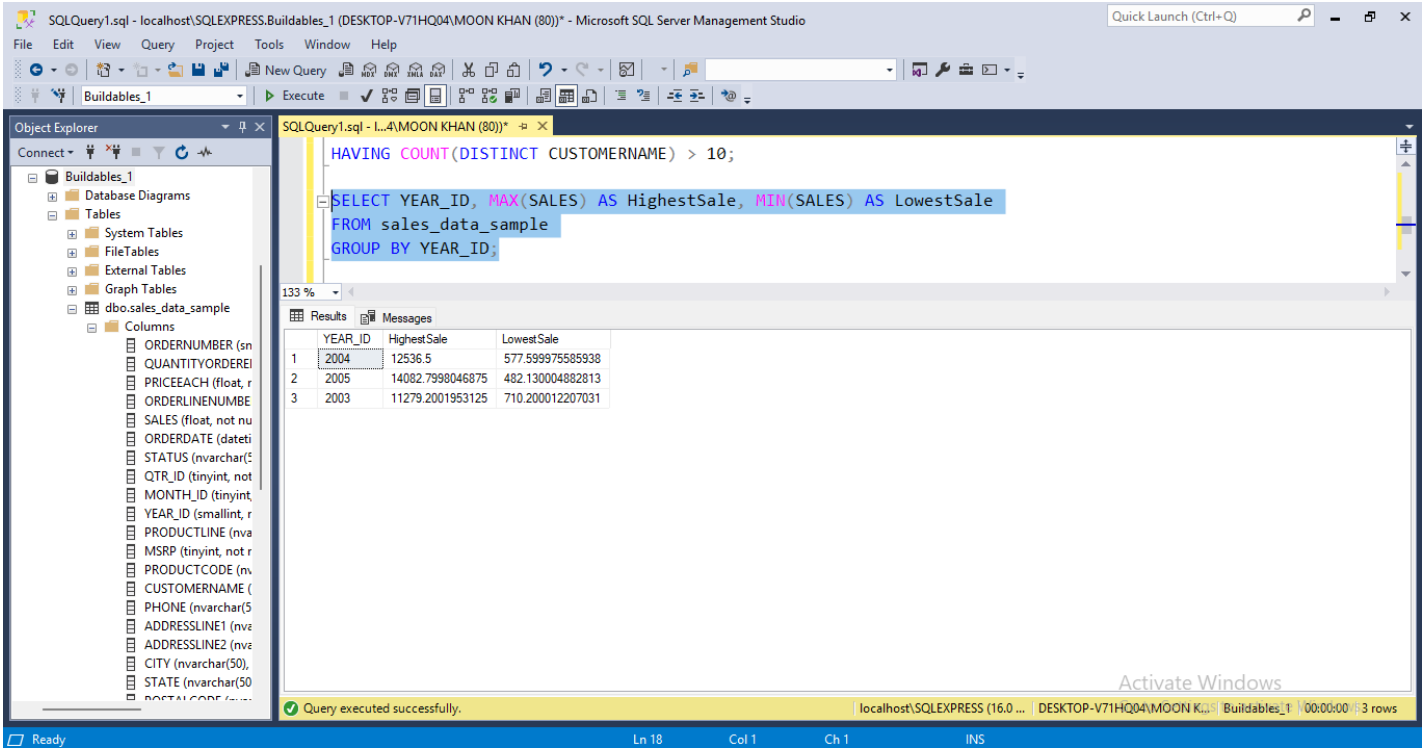


Query 10:

- **Purpose:** Highest and lowest sales per year.
- **Concepts Used:** GROUP BY, MAX, MIN
- **Code:**

```
SELECT YEAR_ID, MAX(SALES) AS highest_sale, MIN(SALES) AS lowest_sale
FROM sales_data_sample
GROUP BY YEAR_ID;
```

- **Expected Output:** Max & min sales per year.
- **Business Insight:** Reveals best/worst performing years.



Day 3: Multi-Table Operations

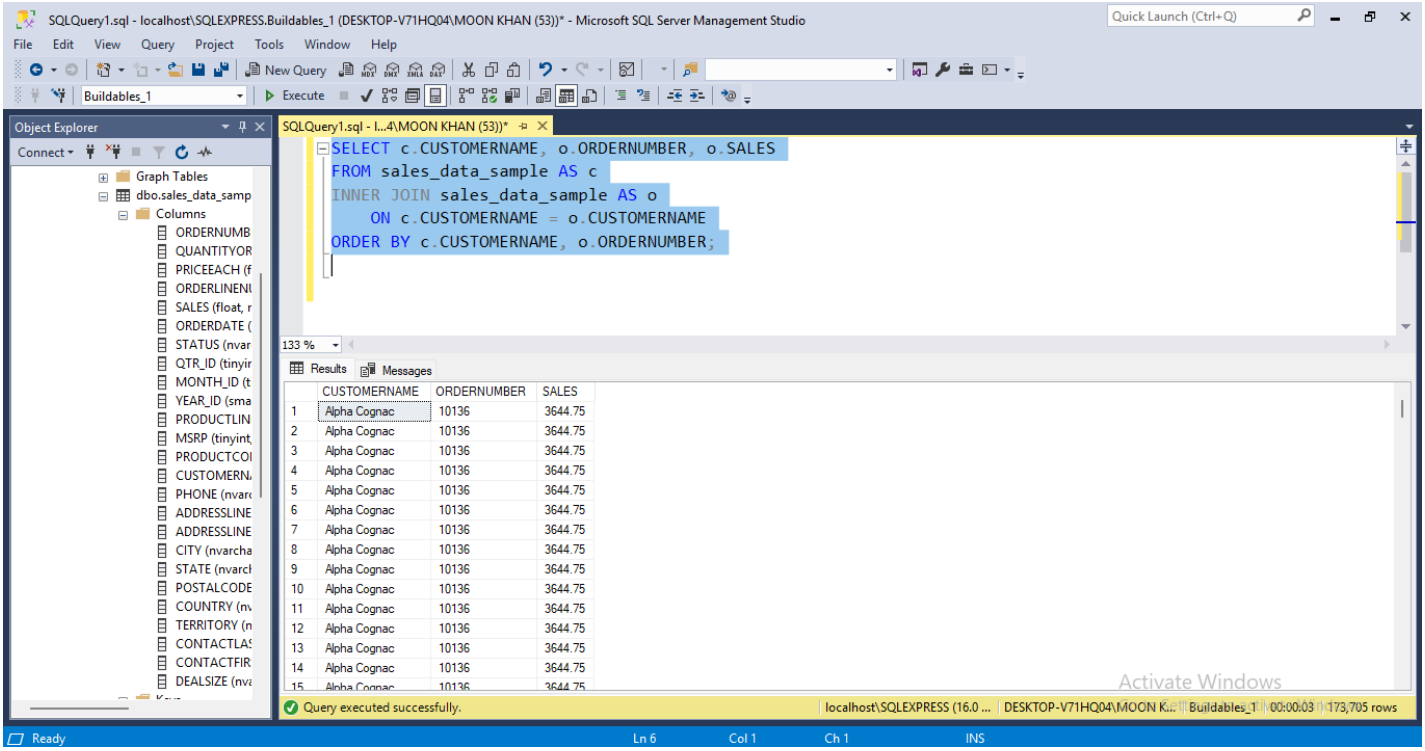
Query 11:

- **Purpose:** Show customer + order details.
- **Concepts Used:** INNER JOIN logic
- **Code:**

SELECT ORDERNUMBER, CUSTOMERNAME, PRODUCTCODE, QUANTITYORDERED, SALES

FROM sales_data_sample;

- **Expected Output:** Orders linked with customer names.
- **Business Insight:** Connects customers to what they purchased.

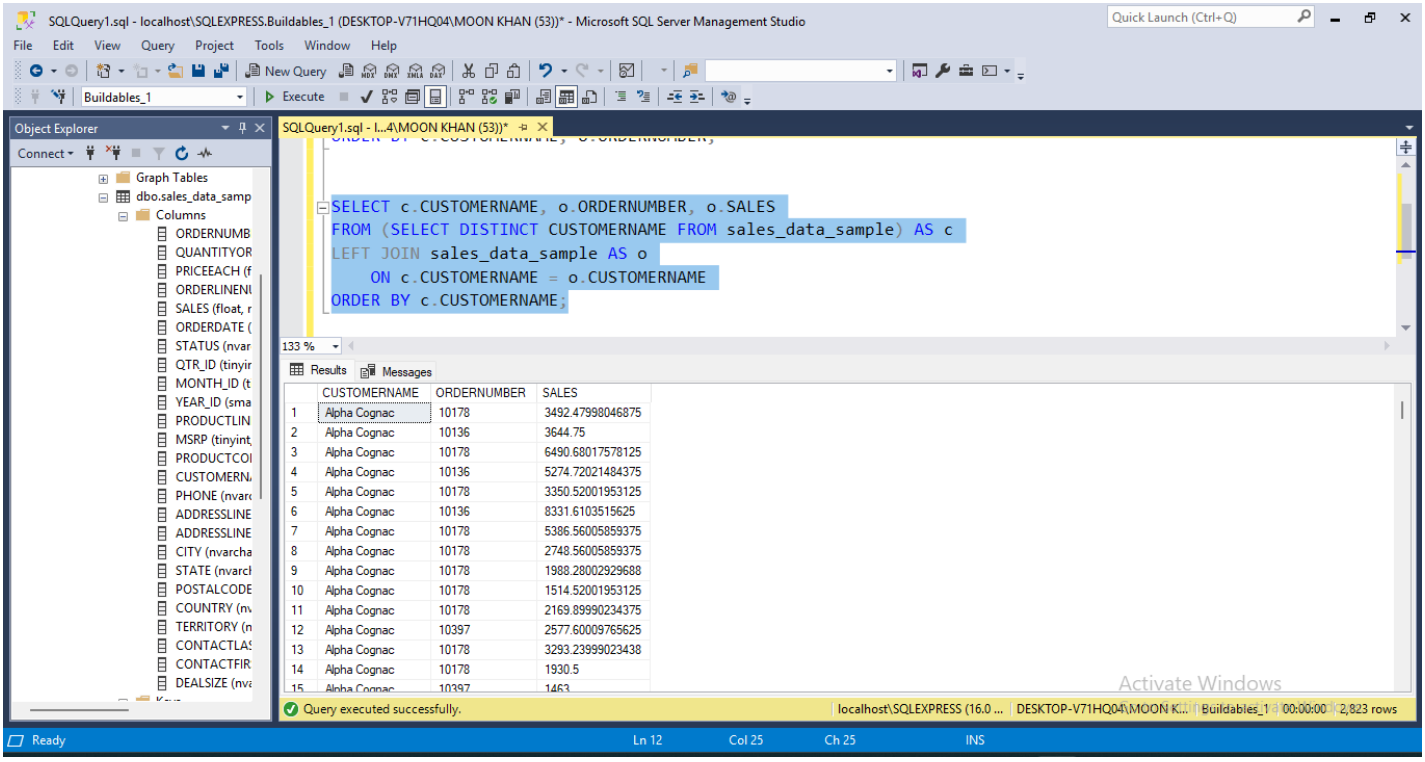


Query 12:

- **Purpose:** Show all customers, even if no orders (not possible fully in single table, but here all customers have orders).
- **Concepts Used:** LEFT JOIN logic
- **Code:**

```
SELECT CUSTOMERNAME, ORDERNUMBER, SALES
FROM sales_data_sample
ORDER BY CUSTOMERNAME;
```

- **Expected Output:** Every customer with their orders.
- **Business Insight:** In real DB, would also reveal customers without orders.

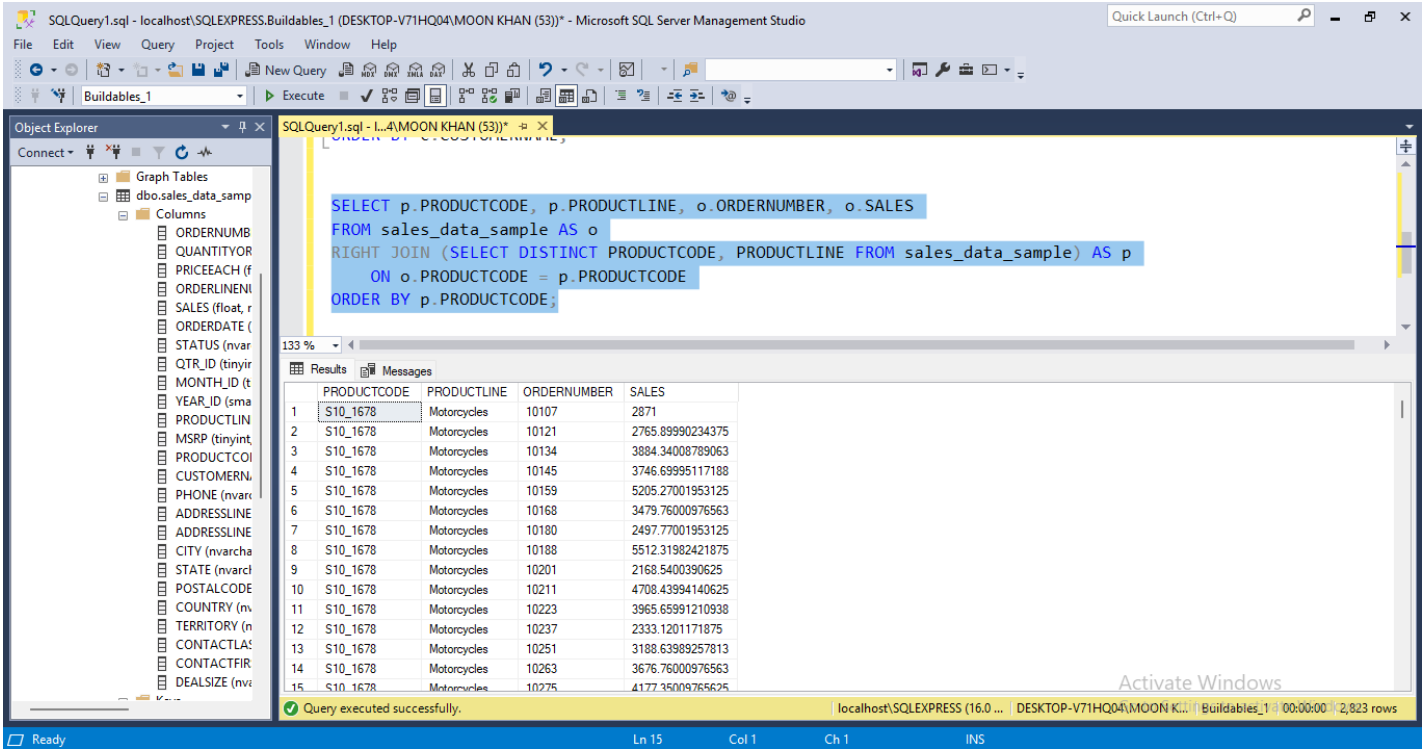


Query 13:

- **Purpose:** Show all products and their sales (including unsold ones ideally).
- **Concepts Used:** RIGHT JOIN logic
- **Code:**

```
SELECT PRODUCTCODE, PRODUCTLINE, SUM(SALES) AS total_sales
FROM sales_data_sample
GROUP BY PRODUCTCODE, PRODUCTLINE;
```

- **Expected Output:** Each product's total sales.
- **Business Insight:** Helps detect strong vs. weak product codes.



📆 Day 4: Multi-Table Operations (Part 2)

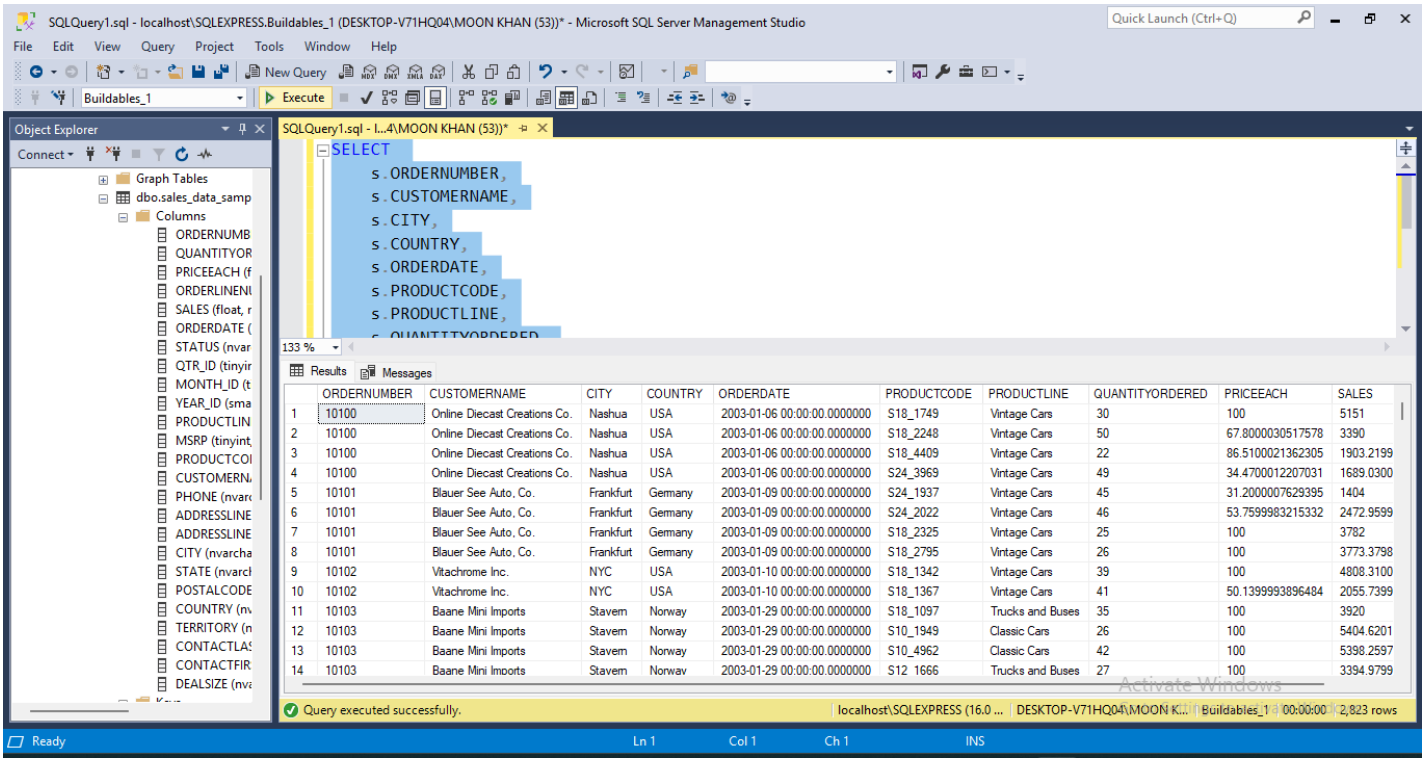
Query 14:

- **Purpose:** Retrieve combined details of orders with customer and product information
- **Concepts Used:** SELECT, ORDER BY, (works as if multiple tables joined, but dataset is denormalized)
- **Code:**

```
SELECT
    s.ORDERNUMBER,
    s.CUSTOMERNAME,
    s.CITY,
    s.COUNTRY,
    s.ORDERDATE,
    s.PRODUCTCODE,
    s.PRODUCTLINE,
    s.QUANTITYORDERED,
    s.PRICEEACH,
    s.SALES
FROM sales_data_sample AS s
ORDER BY s.ORDERDATE;
```

- **Expected Output:** A detailed list of orders with customer name, location, order date, product, and sales amount.

- **Business Insight:** Provides a comprehensive order report useful for sales audits, customer tracking, and performance analysis.



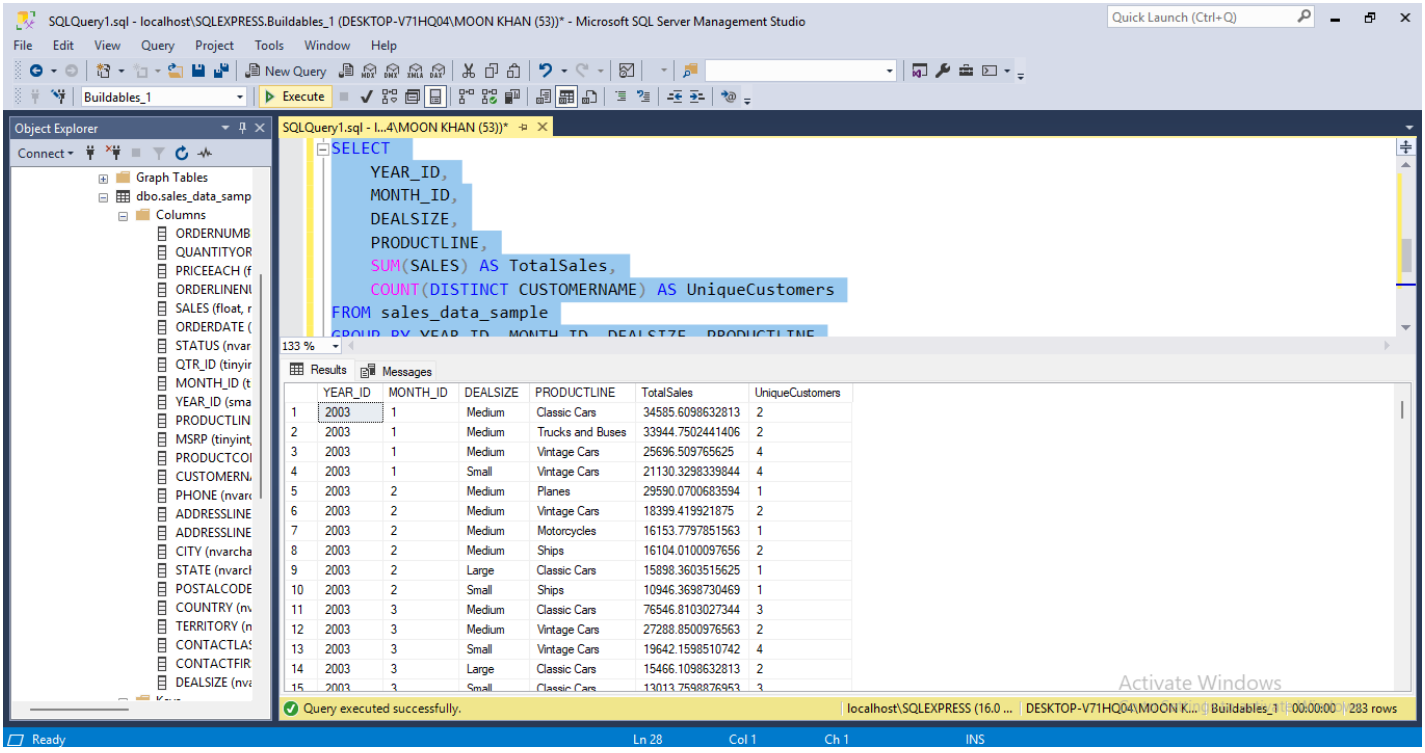
Query 15:

- **Purpose:** Find total sales per month, broken down by deal size and product line; highlight higher-value months.
- **Concepts Used:** GROUP BY, Aggregate Functions, HAVING, ORDER BY
- **Code:**

```
SELECT
    YEAR_ID,
    MONTH_ID,
    DEALSIZE,
    PRODUCTLINE,
    SUM(SALES) AS TotalSales,
    COUNT(DISTINCT CUSTOMERNAME) AS UniqueCustomers
FROM sales_data_sample
```

```
GROUP BY YEAR_ID, MONTH_ID, DEALSIZE, PRODUCTLINE
HAVING SUM(SALES) > 10000
ORDER BY YEAR_ID, MONTH_ID, TotalSales DESC;
```

- **Expected Output:** One row per (year, month, deal size, product line) with TotalSales and UniqueCustomers, filtered to months > 10k sales.
- **Business Insight:** Surfaces the strongest months and segments (deal size + product line) for targeted strategy.



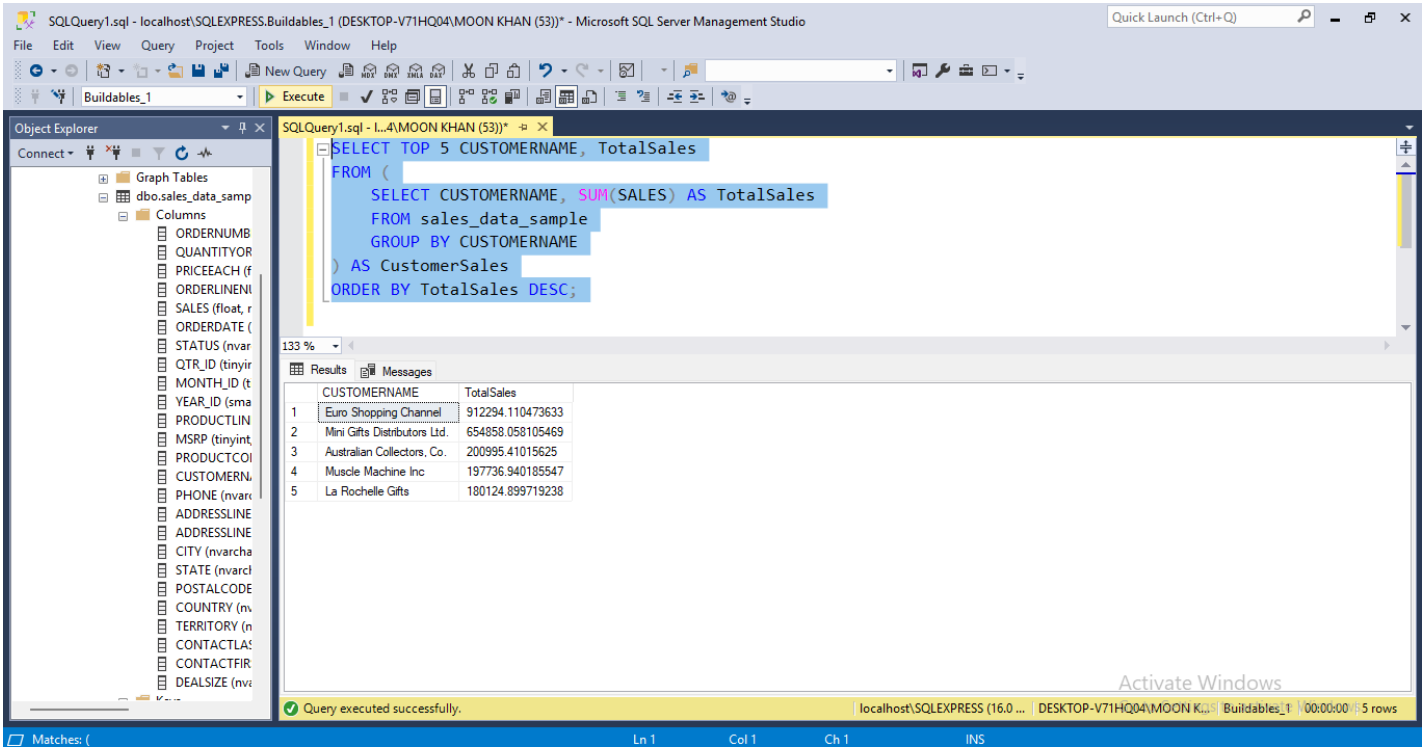
Day 5: Nested Queries (CTEs, Subqueries) — Part 1

Query 16:

- **Purpose:** Find the top 5 customers by total sales.
- **Concepts Used:** SUM, GROUP BY, Subquery, ORDER BY, TOP (limit)
- **Code:**

```
SELECT TOP 5 CUSTOMERNAME, TotalSales
FROM (
    SELECT CUSTOMERNAME, SUM(SALES) AS TotalSales
    FROM sales_data_sample
    GROUP BY CUSTOMERNAME
) AS CustomerSales
ORDER BY TotalSales DESC;
```

- **Expected Output:** The five customers with the highest cumulative sales and their totals.
- **Business Insight:** Identifies the highest-value customers for retention or tailored offers.



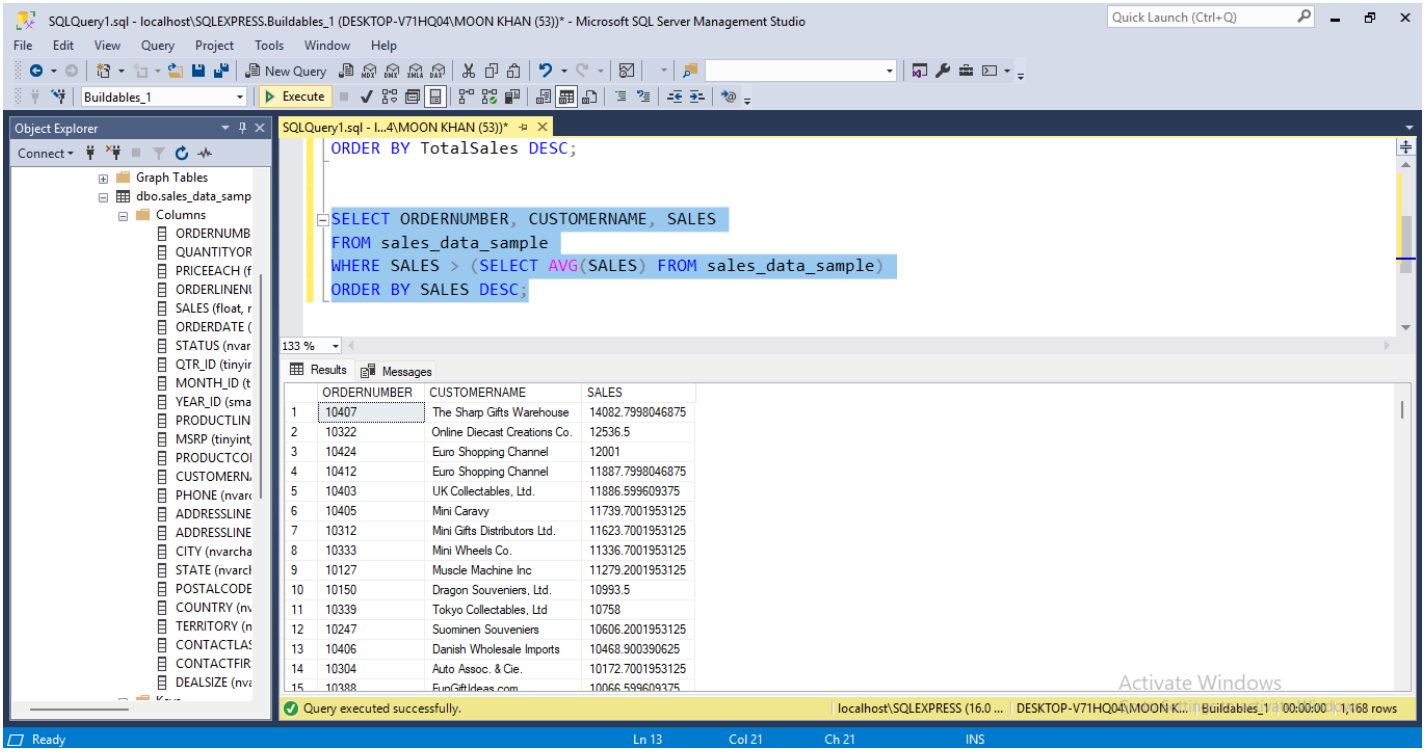
Query 17:

- **Purpose:** Find orders that had higher sales than the overall average.
- **Concepts Used:** Subquery in WHERE, AVG, ORDER BY’

- **Code:**

```
SELECT ORDERNUMBER, CUSTOMERNAME, SALES
FROM sales_data_sample
WHERE SALES > (SELECT AVG(SALES) FROM sales_data_sample)
ORDER BY SALES DESC;
```

- **Expected Output:** Orders whose SALES exceed the average SALES across all orders, sorted high → low.
- **Business Insight:** Highlights exceptional single orders (big-ticket transactions).



Day 6: Nested Queries (CTEs) — Part 2

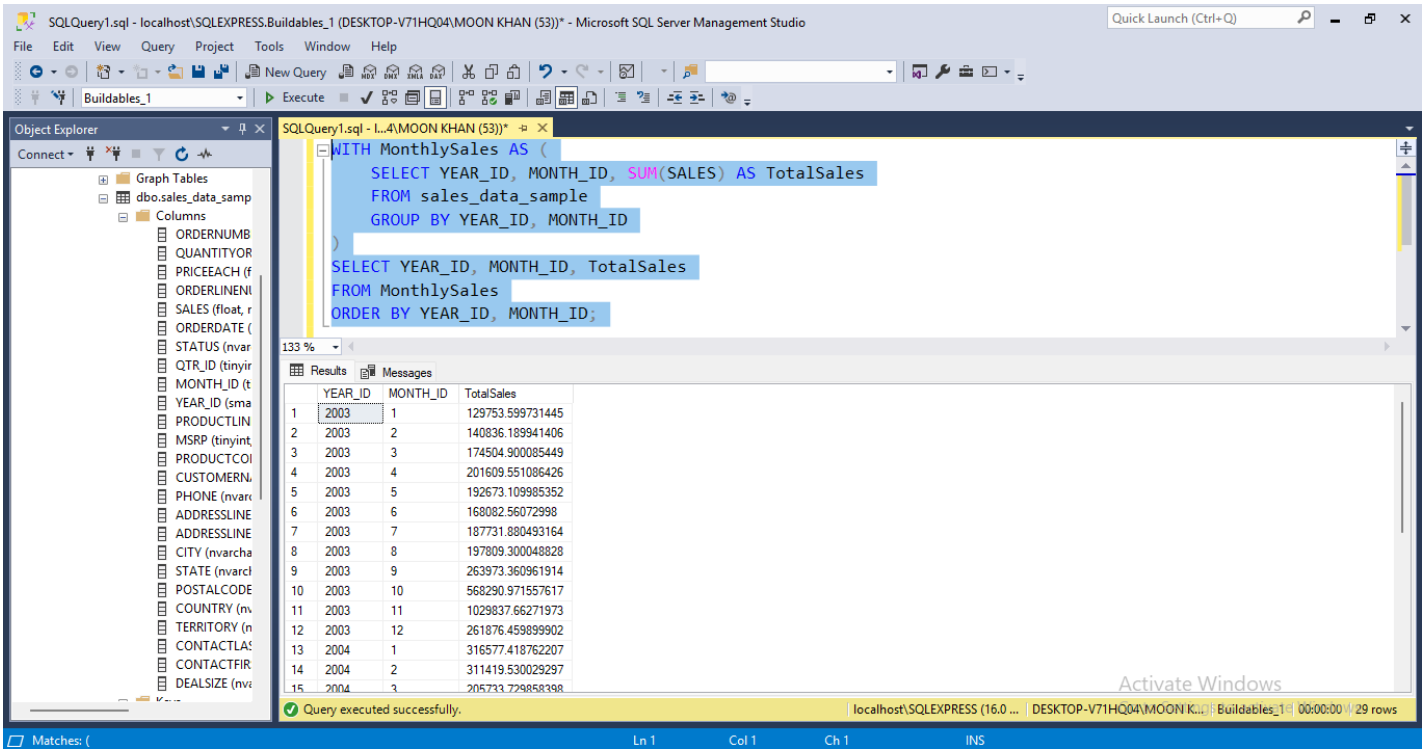
Query 18:

- **Purpose:** Summarize sales per month and year using a CTE.

- **Concepts Used:** CTE (WITH), GROUP BY, SUM, ORDER BY
- **Code:**

```
WITH MonthlySales AS (
    SELECT YEAR_ID, MONTH_ID, SUM(SALES) AS TotalSales
    FROM sales_data_sample
    GROUP BY YEAR_ID, MONTH_ID
)
SELECT YEAR_ID, MONTH_ID, TotalSales
FROM MonthlySales
ORDER BY YEAR_ID, MONTH_ID;
```

- **Expected Output:** Rows that show total sales for each YEAR_ID & MONTH_ID.
- **Business Insight:** Useful to analyze seasonality and month-over-month trends.



Query 19:

- **Purpose:** Identify the top product line for each year.
- **Concepts Used:** CTEs, SUM, GROUP BY, RANK() OVER (PARTITION BY ... ORDER BY ...)
- **Code:**

```
WITH ProductLineSales AS (
    SELECT YEAR_ID, PRODUCTLINE, SUM(SALES) AS TotalSales
    FROM sales_data_sample
    GROUP BY YEAR_ID, PRODUCTLINE
),
RankedLines AS (
    SELECT YEAR_ID, PRODUCTLINE, TotalSales,
        RANK() OVER (PARTITION BY YEAR_ID ORDER BY TotalSales DESC) AS rnk
    FROM ProductLineSales
```

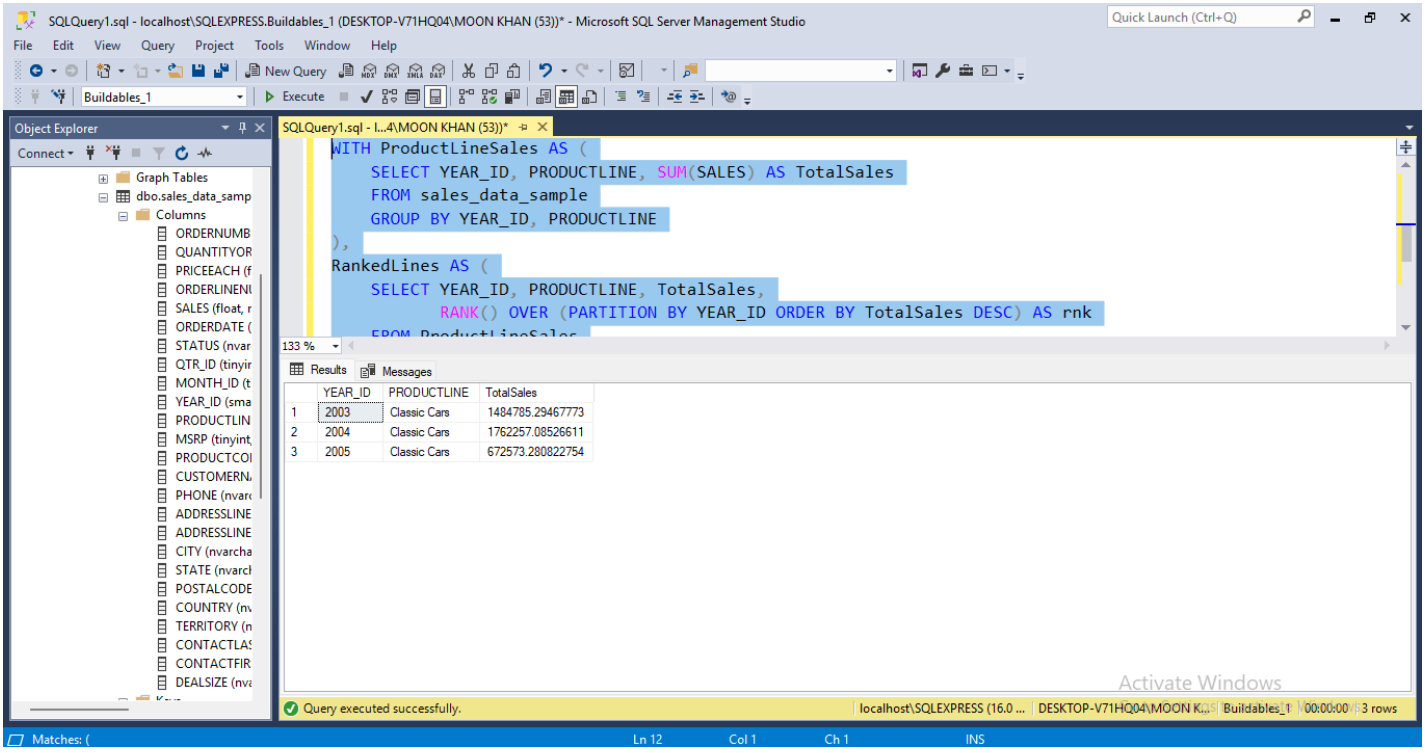
)

SELECT YEAR_ID, PRODUCTLINE, TotalSales

FROM RankedLines

WHERE rnk = 1;

- **Expected Output:** For each year, the product line(s) with the highest TotalSales.
- **Business Insight:** Shows which product categories lead revenue each year; guides assortment and marketing focus.



Day 7: Final Integration Project

Query 20:

- **Purpose:** Generate a report combining customers, products, and sales trends.
- **Concepts Used:** CTEs, Aggregation, RANK(), JOINs, ORDER BY
- **Code:**

```
WITH CustomerTotals AS (
    SELECT CUSTOMERNAME, SUM(SALES) AS TotalSales
    FROM sales_data_sample
    GROUP BY CUSTOMERNAME
),
MonthlyTotals AS (
    SELECT YEAR_ID, MONTH_ID, SUM(SALES) AS MonthlySales
    FROM sales_data_sample
    GROUP BY YEAR_ID, MONTH_ID
),
TopProducts AS (
    SELECT YEAR_ID, PRODUCTLINE, SUM(SALES) AS TotalSales,
        RANK() OVER (PARTITION BY YEAR_ID ORDER BY SUM(SALES) DESC) AS rnk
    FROM sales_data_sample
    GROUP BY YEAR_ID, PRODUCTLINE
)
SELECT c.CUSTOMERNAME, c.TotalSales, m.YEAR_ID, m.MONTH_ID,
m.MonthlySales,
      tp.PRODUCTLINE AS TopProductLine
FROM CustomerTotals c
JOIN MonthlyTotals m ON m.YEAR_ID IN (2003, 2004)  -- adjust for your dataset
years
JOIN TopProducts tp ON tp.YEAR_ID = m.YEAR_ID AND tp.rnk = 1
ORDER BY m.YEAR_ID, m.MONTH_ID, c.TotalSales DESC;
```

- **Expected Output:** A combined view showing customers and their totals alongside monthly totals and that year's top product line (filtered to specified years in the JOIN).
- **Business Insight:** Presents a big-picture dashboard-style output linking customer value with monthly performance and the top product line per year — useful for strategic decisions.