

Buildables Week 1

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

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

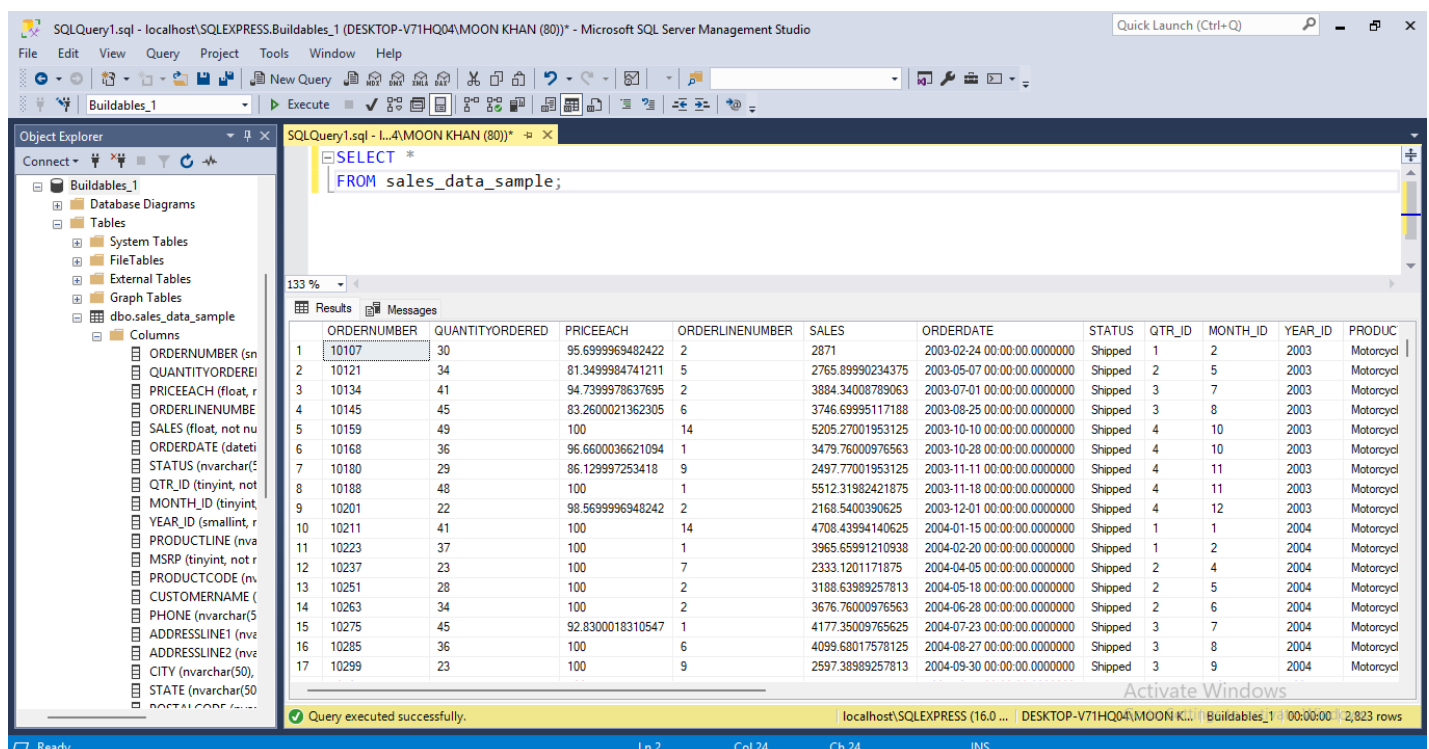
Query 1:

- **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.



SQLQuery1.sql - localhost\SQLEXPRESS\Buildables_1 (DESKTOP-V71HQ04\MOON KHAN (80)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Buildables_1

Object Explorer

- Buildables_1
 - Database Diagrams
 - Tables
 - System Tables
 - FileTables
 - External Tables
 - Graph Tables
 - dbo.sales_data_sample
 - Columns
 - ORDERNUMBER (smallint)
 - QUANTITYORDERED (smallint)
 - PRICEEACH (float, r)
 - ORDERLINENUMBER (smallint)
 - SALES (float, not null)
 - ORDERDATE (datetime)
 - STATUS (nvarchar(50))
 - QTR_ID (tinyint, not null)
 - MONTH_ID (tinyint, not null)
 - YEAR_ID (smallint, r)
 - PRODUCTLINE (nvarchar(50))
 - MSRP (tinyint, not null)
 - PRODUCTCODE (nvarchar(50))
 - CUSTOMERNAME (nvarchar(50))
 - PHONE (nvarchar(50))
 - ADDRESSLINE1 (nvarchar(50))
 - ADDRESSLINE2 (nvarchar(50))
 - CITY (nvarchar(50))
 - STATE (nvarchar(50))
 - POSTALCODE (nvarchar(50))

SQLQuery1.sql - L:\4\MOON KHAN (80))

```
SELECT *  
FROM sales_data_sample;
```

Results

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID	MONTH_ID	YEAR_ID	PRODUCT
1	10107	30	95.6999969482422	2	2871	2003-02-24 00:00:00.0000000	Shipped	1	2	2003	Motorcycl
2	10121	34	81.3499984741211	5	2765.89990234375	2003-05-07 00:00:00.0000000	Shipped	2	5	2003	Motorcycl
3	10134	41	94.7399978637695	2	3884.34008789063	2003-07-01 00:00:00.0000000	Shipped	3	7	2003	Motorcycl
4	10145	45	83.2600021362305	6	3746.69995117188	2003-08-25 00:00:00.0000000	Shipped	3	8	2003	Motorcycl
5	10159	49	100	14	5205.27001953125	2003-10-10 00:00:00.0000000	Shipped	4	10	2003	Motorcycl
6	10168	36	96.6600036621094	1	3479.76000976563	2003-10-28 00:00:00.0000000	Shipped	4	10	2003	Motorcycl
7	10180	29	86.129997253418	9	2497.77001953125	2003-11-11 00:00:00.0000000	Shipped	4	11	2003	Motorcycl
8	10188	48	100	1	5512.31982421875	2003-11-18 00:00:00.0000000	Shipped	4	11	2003	Motorcycl
9	10201	22	98.5699996948242	2	2168.5400390625	2003-12-01 00:00:00.0000000	Shipped	4	12	2003	Motorcycl
10	10211	41	100	14	4708.43994140625	2004-01-15 00:00:00.0000000	Shipped	1	1	2004	Motorcycl
11	10223	37	100	1	3965.65991210938	2004-02-20 00:00:00.0000000	Shipped	1	2	2004	Motorcycl
12	10237	23	100	7	2333.1201171875	2004-04-05 00:00:00.0000000	Shipped	2	4	2004	Motorcycl
13	10251	28	100	2	3188.63989257813	2004-05-18 00:00:00.0000000	Shipped	2	5	2004	Motorcycl
14	10263	34	100	2	3676.76000976563	2004-06-28 00:00:00.0000000	Shipped	2	6	2004	Motorcycl
15	10275	45	92.8300018310547	1	4177.35009765625	2004-07-23 00:00:00.0000000	Shipped	3	7	2004	Motorcycl
16	10285	36	100	6	4099.68017578125	2004-08-27 00:00:00.0000000	Shipped	3	8	2004	Motorcycl
17	10299	23	100	9	2597.38989257813	2004-09-30 00:00:00.0000000	Shipped	3	9	2004	Motorcycl

Query executed successfully.

localhost\SQLEXPRESS (16.0. ... | DESKTOP-V71HQ04\MOON KHAN (80)) | Buildables_1 | 00:00:00 | 2,823 rows

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.

The screenshot displays the Microsoft SQL Server Management Studio interface. The 'Object Explorer' on the left shows the database structure, including the 'sales_data_sample' table. The 'Query Editor' in the center contains the following SQL query:

```
SELECT *  
FROM sales_data_sample;  
  
SELECT CUSTOMERNAME, PRODUCTCODE, SALES  
FROM sales_data_sample;
```

The 'Results' pane at the bottom shows the output of the query, displaying a table with three columns: CUSTOMERNAME, PRODUCTCODE, and SALES. The table contains 17 rows of data, including customer names like 'Land of Toys Inc.', 'Reims Collectables', and 'Lyon Souvenirs', along with their respective product codes and sales values.

	CUSTOMERNAME	PRODUCTCODE	SALES
1	Land of Toys Inc.	S10_1678	2871
2	Reims Collectables	S10_1678	2765.89990234375
3	Lyon Souvenirs	S10_1678	3884.34008789063
4	Toys4GrownUps.com	S10_1678	3746.69995117188
5	Corporate Gift Ideas Co.	S10_1678	5205.27001953125
6	Technics Stores Inc.	S10_1678	3479.76000976563
7	Daedalus Designs Imports	S10_1678	2497.77001953125
8	Herkku Gifts	S10_1678	5512.31982421875
9	Mini Wheels Co.	S10_1678	2168.5400390625
10	Auto Canal Petit	S10_1678	4708.43994140625
11	Australian Collectors. Co.	S10_1678	3965.65991210938
12	Vitachrome Inc.	S10_1678	2333.1201171875
13	Tekni Collectables Inc.	S10_1678	3188.63989257813
14	Gift Depot Inc.	S10_1678	3676.76000976563
15	La Rochelle Gifts	S10_1678	4177.35009765625
16	Marta's Replicas Co.	S10_1678	4099.68017578125
17	Toys of Finland, Co.	S10_1678	2597.38989257813

The status bar at the bottom indicates that the query was executed successfully, returning 2,823 rows.

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.

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

```
FROM sales_data_sample;  
  
SELECT *  
FROM sales_data_sample  
WHERE COUNTRY = 'France';
```

The Results pane shows the output of the query, which is a table with 11 columns: ORDERNUMBER, QUANTITYORDERED, PRICEEACH, ORDERLINENUMBER, SALES, ORDERDATE, STATUS, QTR_ID, MONTH_ID, YEAR_ID, and PRODUCT. The table contains 17 rows of data, all of which are orders from France.

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID	MONTH_ID	YEAR_ID	PRODUCT
2	10134	41	94.7399978637695	2	3884.34008789063	2003-07-01 00:00:00.0000000	Shipped	3	7	2003	Motorcycl
3	10180	29	86.129997253418	9	2497.77001953125	2003-11-11 00:00:00.0000000	Shipped	4	11	2003	Motorcycl
4	10211	41	100	14	4708.43994140625	2004-01-15 00:00:00.0000000	Shipped	1	1	2004	Motorcycl
5	10275	45	92.8300018310547	1	4177.35009765625	2004-07-23 00:00:00.0000000	Shipped	3	7	2004	Motorcycl
6	10375	21	34.9099998474121	12	733.109985351563	2005-02-03 00:00:00.0000000	Shipped	1	2	2005	Motorcycl
7	10194	42	100	11	7290.35986328125	2003-11-25 00:00:00.0000000	Shipped	4	11	2003	Classic C
8	10304	47	100	6	10172.7001953125	2004-10-11 00:00:00.0000000	Shipped	4	10	2004	Classic C
9	10134	27	100	5	3307.77001953125	2003-07-01 00:00:00.0000000	Shipped	3	7	2003	Motorcycl
10	10180	42	100	12	4695.60009765625	2003-11-11 00:00:00.0000000	Shipped	4	11	2003	Motorcycl
11	10275	22	100	4	2904.43994140625	2004-07-23 00:00:00.0000000	Shipped	3	7	2004	Motorcycl
12	10298	39	96.3399963378906	1	3757.26000976563	2004-09-27 00:00:00.0000000	Shipped	3	9	2004	Motorcycl
13	10402	45	100	1	5833.7998046875	2005-04-07 00:00:00.0000000	Shipped	2	4	2005	Motorcycl
14	10134	31	100	4	7023.97998046875	2003-07-01 00:00:00.0000000	Shipped	3	7	2003	Motorcycl
15	10180	41	100	11	8892.900390625	2003-11-11 00:00:00.0000000	Shipped	4	11	2003	Motorcycl
16	10275	36	100	3	6901.919921875	2004-07-23 00:00:00.0000000	Shipped	3	7	2004	Motorcycl
17	10178	24	100	12	3492.47998046875	2003-11-08 00:00:00.0000000	Shipped	4	11	2003	Classic C

The status bar at the bottom indicates that the query was executed successfully and returned 314 rows.

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.

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

```
WHERE COUNTRY = 'France';  
  
SELECT *  
FROM sales_data_sample  
WHERE COUNTRY = 'USA' AND SALES > 500;
```

The Results pane shows the output of the query, displaying columns: ORDERNUMBER, QUANTITYORDERED, PRICEEACH, ORDERLINENUMBER, SALES, ORDERDATE, STATUS, QTR_ID, MONTH_ID, YEAR_ID, and PRODUCT. The first 17 rows are visible, showing data for various orders and products.

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID	MONTH_ID	YEAR_ID	PRODUCT
1	10107	30	95.6999969482422	2	2871	2003-02-24 00:00:00.0000000	Shipped	1	2	2003	Motorcycl
2	10145	45	83.2600021362305	6	3746.69995117188	2003-08-25 00:00:00.0000000	Shipped	3	8	2003	Motorcycl
3	10159	49	100	14	5205.27001953125	2003-10-10 00:00:00.0000000	Shipped	4	10	2003	Motorcycl
4	10168	36	96.6600036621094	1	3479.76000976563	2003-10-28 00:00:00.0000000	Shipped	4	10	2003	Motorcycl
5	10201	22	98.5699996948242	2	2168.5400390625	2003-12-01 00:00:00.0000000	Shipped	4	12	2003	Motorcycl
6	10237	23	100	7	2333.1201171875	2004-04-05 00:00:00.0000000	Shipped	2	4	2004	Motorcycl
7	10251	28	100	2	3188.63989257813	2004-05-18 00:00:00.0000000	Shipped	2	5	2004	Motorcycl
8	10263	34	100	2	3676.76000976563	2004-06-28 00:00:00.0000000	Shipped	2	6	2004	Motorcycl
9	10285	36	100	6	4099.68017578125	2004-08-27 00:00:00.0000000	Shipped	3	8	2004	Motorcycl
10	10318	46	94.7399978637695	1	4358.0400390625	2004-11-02 00:00:00.0000000	Shipped	4	11	2004	Motorcycl
11	10329	42	100	1	4396.14013671875	2004-11-15 00:00:00.0000000	Shipped	4	11	2004	Motorcycl
12	10388	42	76.3600006103516	4	3207.1201171875	2005-03-03 00:00:00.0000000	Shipped	1	3	2005	Motorcycl
13	10140	37	100	11	7374.10009765625	2003-07-24 00:00:00.0000000	Shipped	3	7	2003	Classic C
14	10163	21	100	1	4860.240234375	2003-10-20 00:00:00.0000000	Shipped	4	10	2003	Classic C
15	10183	23	100	8	5372.56982421875	2003-11-13 00:00:00.0000000	Shipped	4	11	2003	Classic C
16	10215	35	100	3	6075.2998046875	2004-01-29 00:00:00.0000000	Shipped	1	1	2004	Classic C
17	10222	22	100	2	6123.22000000000	2004-02-18 00:00:00.0000000	Shipped	1	2	2004	Classic C

The status bar at the bottom indicates: Query executed successfully. | localhost\SQLEXPRESS (16.0 ... | DESKTOP-V71HQ04\MOON K... | Buildables_1 | 00:00:00 | 1,004 rows

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.

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

```
WHERE COUNTRY = 'USA' AND SALES > 500;  
  
SELECT CUSTOMERNAME, SALES  
FROM sales_data_sample  
ORDER BY SALES DESC;
```

The Object Explorer on the left shows the database structure, including the `sales_data_sample` table. The Results pane shows the output of the query, displaying a list of customers and their sales values, sorted in descending order of sales.

	CUSTOMERNAME	SALES
1	The Sharp Gifts Warehouse	14082.7998046875
2	Online Diecast Creations Co.	12536.5
3	Euro Shopping Channel	12001
4	Euro Shopping Channel	11887.7998046875
5	UK Collectables, Ltd.	11886.599609375
6	Mini Caravay	11739.7001953125
7	Mini Gifts Distributors Ltd.	11623.7001953125
8	Mini Wheels Co.	11336.7001953125
9	Muscle Machine Inc	11279.2001953125
10	Dragon Souvenirs, Ltd.	10993.5
11	Tokyo Collectables, Ltd	10758
12	Suominen Souvenirs	10606.2001953125
13	Danish Wholesale Imports	10468.900390625
14	Auto Assoc. & Cie.	10172.7001953125
15	FunGiftIdeas.com	10066.599609375
16	La Rochelle Gifts	10039.599609375
17	Australian Collectors, Co.	9774.0302734375

The status bar at the bottom indicates that the query was executed successfully, returning 2,823 rows.

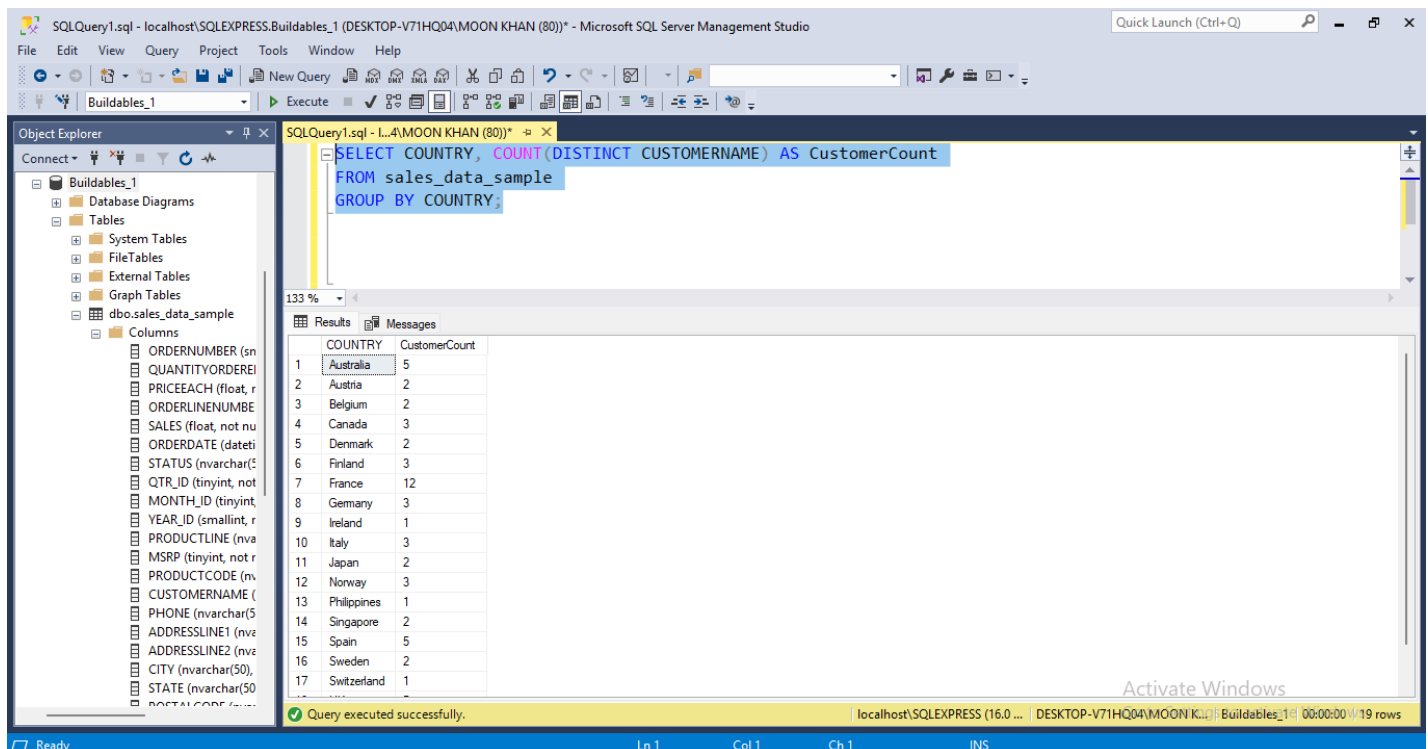
Day 2 – Data Aggregation (GROUP BY, HAVING, Aggregates)

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.



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

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

The results are shown in the Results pane, displaying a table with 17 rows and 2 columns: COUNTRY and CustomerCount.

COUNTRY	CustomerCount
Australia	5
Austria	2
Belgium	2
Canada	3
Denmark	2
Finland	3
France	12
Germany	3
Ireland	1
Italy	3
Japan	2
Norway	3
Philippines	1
Singapore	2
Spain	5
Sweden	2
Switzerland	1

The status bar at the bottom indicates "Query executed successfully." and "19 rows".

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.

The screenshot displays the Microsoft SQL Server Management Studio interface. The 'Object Explorer' on the left shows the database structure, including the 'sales_data_sample' table. The 'SQLQuery1.sql' window in the center contains the following query:

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

The 'Results' pane at the bottom shows the output of the query, which is a table with two columns: 'PRODUCTLINE' and 'TotalSales'. The data is as follows:

PRODUCTLINE	TotalSales
Trains	226243.468994141
Motorcycles	1166388.3392334
Ships	714437.130126953
Trucks and Buses	1127789.84326172
Vintage Cars	1903150.83557129
Classic Cars	3919615.6607666
Planes	975003.571350098

The status bar at the bottom indicates 'Query executed successfully.' and shows the execution time as 00:00:00 with 7 rows returned.

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.

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

```
GROUP BY PRODUCTLINE;  
  
SELECT COUNTRY, AVG(SALES) AS AvgOrderValue  
FROM sales_data_sample  
GROUP BY COUNTRY;
```

The Results pane shows the output of the query, displaying a table with two columns: COUNTRY and AvgOrderValue. The table contains 17 rows of data, representing the average sales for each country.

COUNTRY	AvgOrderValue
Finland	3582.41206890604
USA	3613.52871090102
Italy	3315.70186702762
Philippines	3615.98963341346
Germany	3556.00144712387
Switzerland	3797.21156754032
Australia	3408.77350678315
Sweden	3684.45982815927
Austria	3673.8641878995
UK	3325.55874252319
Canada	3201.12228480748
Ireland	3609.77689361572
Norway	3617.21999655331
France	3537.95070633615
Belgium	3285.23089784564
Japan	3618.61174245981
Spain	3554.64012386924

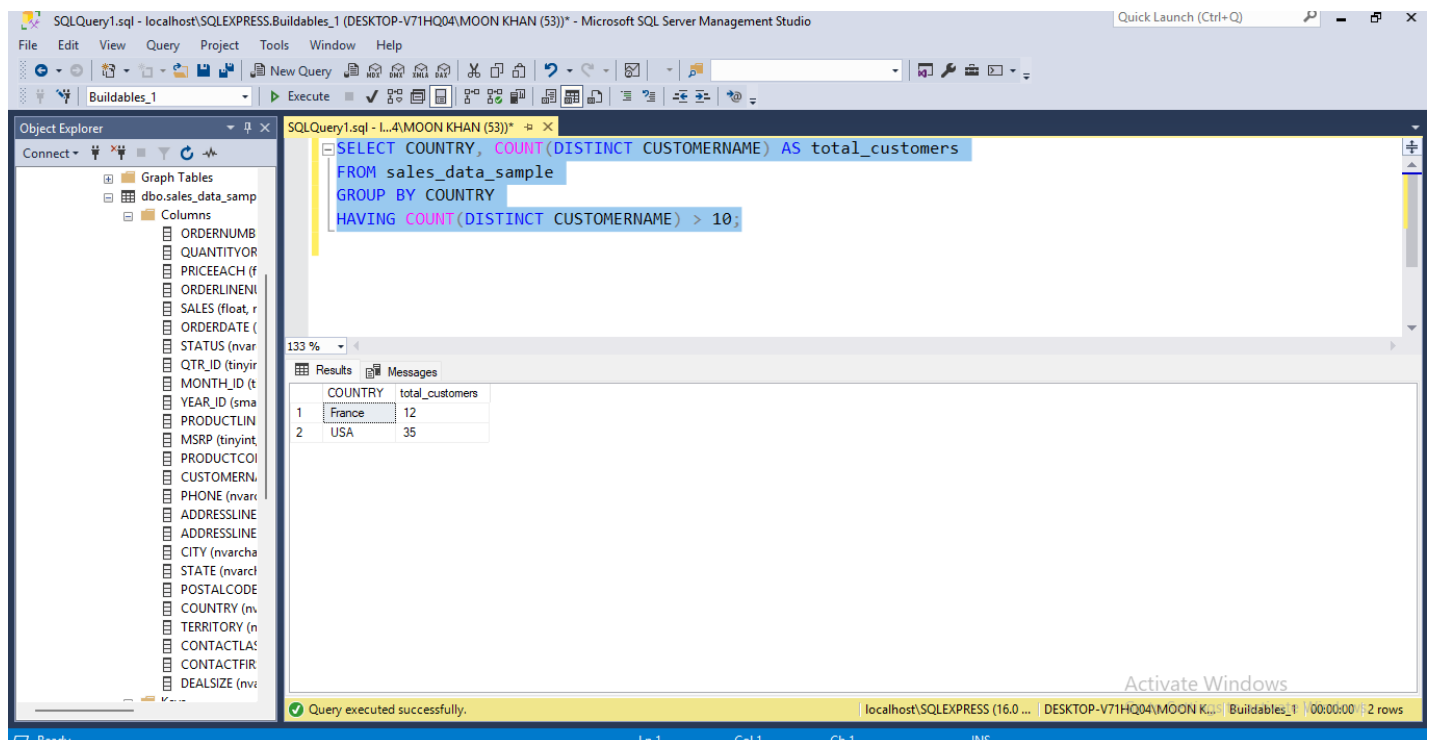
The status bar at the bottom indicates that the query was executed successfully, returning 19 rows.

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.



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

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

The Results pane shows the output of the query:

	COUNTRY	total_customers
1	France	12
2	USA	35

The status bar at the bottom indicates "Query executed successfully." and "2 rows".

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.

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

```
HAVING COUNT(DISTINCT CUSTOMERNAME) > 10;  
  
SELECT YEAR_ID, MAX(SALES) AS HighestSale, MIN(SALES) AS LowestSale  
FROM sales_data_sample  
GROUP BY YEAR_ID;
```

The Results pane shows the output of the query:

	YEAR_ID	HighestSale	LowestSale
1	2004	12536.5	577.599975585938
2	2005	14082.7998046875	482.130004882813
3	2003	11279.2001953125	710.200012207031

The status bar at the bottom indicates "Query executed successfully." and "localhost\SQLEXPRESS (16.0 ... | DESKTOP-V71HQ04\MOON KHAN | Buildables_1 | 00:00:00 | 3 rows".

Day 3: Multi-Table Operations

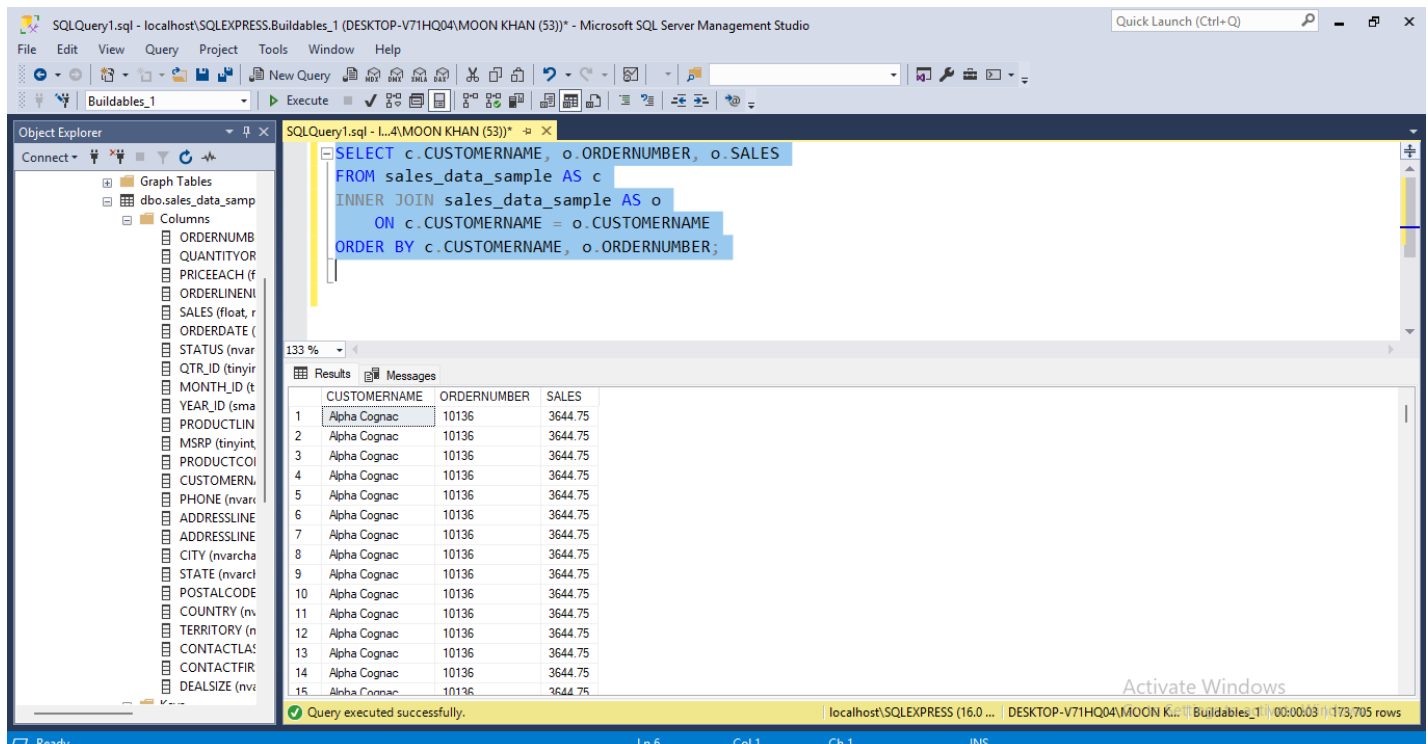
Query 11:

- **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.



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

```
SELECT c.CUSTOMERNAME, o.ORDERNUMBER, o.SALES  
FROM sales_data_sample AS c  
INNER JOIN sales_data_sample AS o  
ON c.CUSTOMERNAME = o.CUSTOMERNAME  
ORDER BY c.CUSTOMERNAME, o.ORDERNUMBER;
```

The Results pane shows the output of the query, displaying 15 rows of data with columns CUSTOMERNAME, ORDERNUMBER, and SALES. The data is as follows:

	CUSTOMERNAME	ORDERNUMBER	SALES
1	Alpha Cognac	10136	3644.75
2	Alpha Cognac	10136	3644.75
3	Alpha Cognac	10136	3644.75
4	Alpha Cognac	10136	3644.75
5	Alpha Cognac	10136	3644.75
6	Alpha Cognac	10136	3644.75
7	Alpha Cognac	10136	3644.75
8	Alpha Cognac	10136	3644.75
9	Alpha Cognac	10136	3644.75
10	Alpha Cognac	10136	3644.75
11	Alpha Cognac	10136	3644.75
12	Alpha Cognac	10136	3644.75
13	Alpha Cognac	10136	3644.75
14	Alpha Cognac	10136	3644.75
15	Alpha Cognac	10136	3644.75

The status bar at the bottom indicates "Query executed successfully." and "173,705 rows".

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.

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

```
SELECT c.CUSTOMERNAME, o.ORDERNUMBER, o.SALES  
FROM (SELECT DISTINCT CUSTOMERNAME FROM sales_data_sample) AS c  
LEFT JOIN sales_data_sample AS o  
ON c.CUSTOMERNAME = o.CUSTOMERNAME  
ORDER BY c.CUSTOMERNAME;
```

The Results pane shows the output of the query, which is a table with 15 rows and 3 columns: CUSTOMERNAME, ORDERNUMBER, and SALES. The data is as follows:

	CUSTOMERNAME	ORDERNUMBER	SALES
1	Alpha Cognac	10178	3492.47998046875
2	Alpha Cognac	10136	3644.75
3	Alpha Cognac	10178	6490.68017578125
4	Alpha Cognac	10136	5274.72021484375
5	Alpha Cognac	10178	3350.52001953125
6	Alpha Cognac	10136	8331.6103515625
7	Alpha Cognac	10178	5386.56005859375
8	Alpha Cognac	10178	2748.56005859375
9	Alpha Cognac	10178	1988.28002929688
10	Alpha Cognac	10178	1514.52001953125
11	Alpha Cognac	10178	2169.89990234375
12	Alpha Cognac	10397	2577.60009765625
13	Alpha Cognac	10178	3293.23999023438
14	Alpha Cognac	10178	1930.5
15	Alpha Cognac	10397	1463

The status bar at the bottom indicates that the query was executed successfully, returning 2,823 rows.

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.

The screenshot displays the Microsoft SQL Server Management Studio interface. The query editor shows a SQL query using a RIGHT JOIN to list all products and their sales. The results pane shows 15 rows of data for product S10_1678, all categorized as Motorcycles. The status bar at the bottom indicates the query was executed successfully, returning 2,823 rows.

```
SELECT p.PRODUCTCODE, p.PRODUCTLINE, o.ORDERNUMBER, o.SALES  
FROM sales_data_sample AS o  
RIGHT JOIN (SELECT DISTINCT PRODUCTCODE, PRODUCTLINE FROM sales_data_sample) AS p  
ON o.PRODUCTCODE = p.PRODUCTCODE  
ORDER BY p.PRODUCTCODE;
```

	PRODUCTCODE	PRODUCTLINE	ORDERNUMBER	SALES
1	S10_1678	Motorcycles	10107	2871
2	S10_1678	Motorcycles	10121	2765.89990234375
3	S10_1678	Motorcycles	10134	3884.34008789063
4	S10_1678	Motorcycles	10145	3746.69995117188
5	S10_1678	Motorcycles	10159	5205.27001953125
6	S10_1678	Motorcycles	10168	3479.76000976563
7	S10_1678	Motorcycles	10180	2497.77001953125
8	S10_1678	Motorcycles	10188	5512.31982421875
9	S10_1678	Motorcycles	10201	2168.5400390625
10	S10_1678	Motorcycles	10211	4708.43994140625
11	S10_1678	Motorcycles	10223	3965.65991210938
12	S10_1678	Motorcycles	10237	2333.1201171875
13	S10_1678	Motorcycles	10251	3188.63989257813
14	S10_1678	Motorcycles	10263	3676.76000976563
15	S10_1678	Motorcycles	10275	4177.35009765625

Query executed successfully. | localhost\SQLEXPRESS (16.0 ... | DESKTOP-V71HQ04\MOON K... | Buildables_1 | 00:00:00 | 2,823 rows

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.

SQLQuery1.sql - localhost\SQLEXPRESS:Buildables_1 (DESKTOP-V71HQ04\MOON KHAN (53)) - Microsoft SQL Server Management Studio

Object Explorer: Graph Tables, dbo.sales_data_samp, Columns

Columns: ORDERNUMB, QUANTITYOR, PRICEEACH (f, ORDERLINENI, SALES (float, r, ORDERDATE (, STATUS (nvarchar, QTR_ID (tinyint, MONTH_ID (t, YEAR_ID (sma, PRODUCTLIN, MSRP (tinyint, PRODUCTCOI, CUSTOMERN, PHONE (nvarchar, ADDRESSLINE, ADDRESSLINE, CITY (nvarchar, STATE (nvarchar, POSTALCODE, COUNTRY (nv, TERRITORY (n, CONTACTLA, CONTACTFIR, DEALSIZE (nv

SQLQuery1.sql - L:\4\MOON KHAN (53))

```
SELECT
    s.ORDERNUMBER,
    s.CUSTOMERNAME,
    s.CITY,
    s.COUNTRY,
    s.ORDERDATE,
    s.PRODUCTCODE,
    s.PRODUCTLINE,
    s.QUANTITYORDERED
```

Results

	ORDERNUMBER	CUSTOMERNAME	CITY	COUNTRY	ORDERDATE	PRODUCTCODE	PRODUCTLINE	QUANTITYORDERED	PRICEEACH	SALES
1	10100	Online Diecast Creations Co.	Nashua	USA	2003-01-06 00:00:00.0000000	S18_1749	Vintage Cars	30	100	5151
2	10100	Online Diecast Creations Co.	Nashua	USA	2003-01-06 00:00:00.0000000	S18_2248	Vintage Cars	50	67.8000030517578	3390
3	10100	Online Diecast Creations Co.	Nashua	USA	2003-01-06 00:00:00.0000000	S18_4409	Vintage Cars	22	86.51000021362305	1903.2199
4	10100	Online Diecast Creations Co.	Nashua	USA	2003-01-06 00:00:00.0000000	S24_3969	Vintage Cars	49	34.4700012207031	1689.0300
5	10101	Blauer See Auto, Co.	Frankfurt	Germany	2003-01-09 00:00:00.0000000	S24_1937	Vintage Cars	45	31.2000007629395	1404
6	10101	Blauer See Auto, Co.	Frankfurt	Germany	2003-01-09 00:00:00.0000000	S24_2022	Vintage Cars	46	53.7599983215332	2472.9599
7	10101	Blauer See Auto, Co.	Frankfurt	Germany	2003-01-09 00:00:00.0000000	S18_2325	Vintage Cars	25	100	3782
8	10101	Blauer See Auto, Co.	Frankfurt	Germany	2003-01-09 00:00:00.0000000	S18_2795	Vintage Cars	26	100	3773.3798
9	10102	Vitachrome Inc.	NYC	USA	2003-01-10 00:00:00.0000000	S18_1342	Vintage Cars	39	100	4808.3100
10	10102	Vitachrome Inc.	NYC	USA	2003-01-10 00:00:00.0000000	S18_1367	Vintage Cars	41	50.1399993896484	2055.7399
11	10103	Baane Mini Imports	Stavem	Norway	2003-01-29 00:00:00.0000000	S18_1097	Trucks and Buses	35	100	3920
12	10103	Baane Mini Imports	Stavem	Norway	2003-01-29 00:00:00.0000000	S10_1949	Classic Cars	26	100	5404.6201
13	10103	Baane Mini Imports	Stavem	Norway	2003-01-29 00:00:00.0000000	S10_4962	Classic Cars	42	100	5398.2597
14	10103	Baane Mini Imports	Stavem	Norway	2003-01-29 00:00:00.0000000	S12_1666	Trucks and Buses	27	100	3394.9799

Query executed successfully. | localhost\SQLEXPRESS (16.0 ... | DESKTOP-V71HQ04\MOON KHAN (53) | Buildables_1 | 00:00:00 | 2,823 rows

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.

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

The Results window displays the following data:

	YEAR_ID	MONTH_ID	DEALSIZE	PRODUCTLINE	TotalSales	UniqueCustomers
1	2003	1	Medium	Classic Cars	34585.6098632813	2
2	2003	1	Medium	Trucks and Buses	33944.7502441406	2
3	2003	1	Medium	Vintage Cars	25696.509765625	4
4	2003	1	Small	Vintage Cars	21130.3298339844	4
5	2003	2	Medium	Planes	29590.0700683594	1
6	2003	2	Medium	Vintage Cars	18399.419921875	2
7	2003	2	Medium	Motorcycles	16153.7797851563	1
8	2003	2	Medium	Ships	16104.0100097656	2
9	2003	2	Large	Classic Cars	15898.3603515625	1
10	2003	2	Small	Ships	10946.3698730469	1
11	2003	3	Medium	Classic Cars	76546.8103027344	3
12	2003	3	Medium	Vintage Cars	27288.8500976563	2
13	2003	3	Small	Vintage Cars	19642.1598510742	4
14	2003	3	Large	Classic Cars	15466.1098632813	2
15	2002	1	Small	Classic Cars	13013.7598876953	2

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

Query 16:

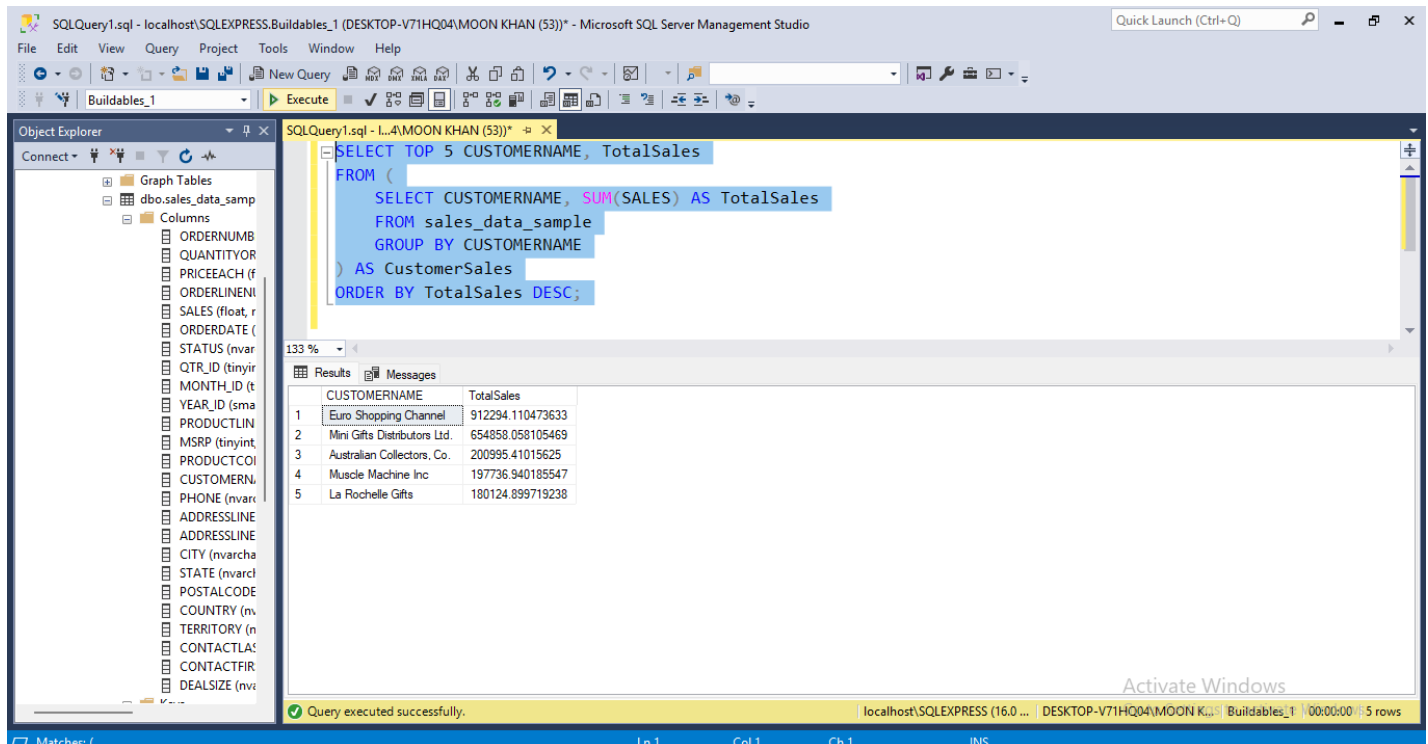
- **Purpose:** Find the top 5 customers by total sales.
- **Concepts Used:** SUM, GROUP BY, Subquery, ORDER BY, TOP (limit)
- **Code:**


```

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.



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

```

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;

```

The query has been executed successfully, and the results are shown in the Results pane. The results table has two columns: CUSTOMERNAME and TotalSales. The data is as follows:

	CUSTOMERNAME	TotalSales
1	Euro Shopping Channel	912294.110473633
2	Mini Gifts Distributors Ltd.	654858.058105469
3	Australian Collectors, Co.	200995.41015625
4	Muscle Machine Inc	197736.940185547
5	La Rochelle Gifts	180124.899719238

The status bar at the bottom indicates that the query was executed successfully on the 'Buildables_1' database, taking 00:00:00.5 rows.

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).

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

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

The Results pane displays the output of the query, showing 15 rows of data. The columns are ORDERNUMBER, CUSTOMERNAME, and SALES. The data is sorted in descending order of SALES.

	ORDERNUMBER	CUSTOMERNAME	SALES
1	10407	The Sharp Gifts Warehouse	14082.7998046875
2	10322	Online Diecast Creations Co.	12536.5
3	10424	Euro Shopping Channel	12001
4	10412	Euro Shopping Channel	11887.7998046875
5	10403	UK Collectables, Ltd.	11886.599609375
6	10405	Mini Caravay	11739.7001953125
7	10312	Mini Gifts Distributors Ltd.	11623.7001953125
8	10333	Mini Wheels Co.	11336.7001953125
9	10127	Muscle Machine Inc	11279.2001953125
10	10150	Dragon Souvenirs, Ltd.	10993.5
11	10339	Tokyo Collectables, Ltd	10758
12	10247	Suominen Souvenirs	10606.2001953125
13	10406	Danish Wholesale Imports	10468.900390625
14	10304	Auto Assoc. & Cie.	10172.7001953125
15	10388	FunGiftIdeas.com	10066.599609375

The status bar at the bottom indicates that the query was executed successfully, returning 1,168 rows.

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.

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

```

The Results pane displays the following data:

	YEAR_ID	MONTH_ID	TotalSales
1	2003	1	129753.599731445
2	2003	2	140836.189941406
3	2003	3	174504.900085449
4	2003	4	201609.551086426
5	2003	5	192673.109985352
6	2003	6	168082.56072998
7	2003	7	187731.880493164
8	2003	8	197809.300048828
9	2003	9	263973.360961914
10	2003	10	568290.971557617
11	2003	11	1029837.66271973
12	2003	12	261876.459899902
13	2004	1	316577.418762207
14	2004	2	311419.530029297
15	2004	3	205733.729858398

The status bar at the bottom indicates "Query executed successfully." and "29 rows".

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

```

)

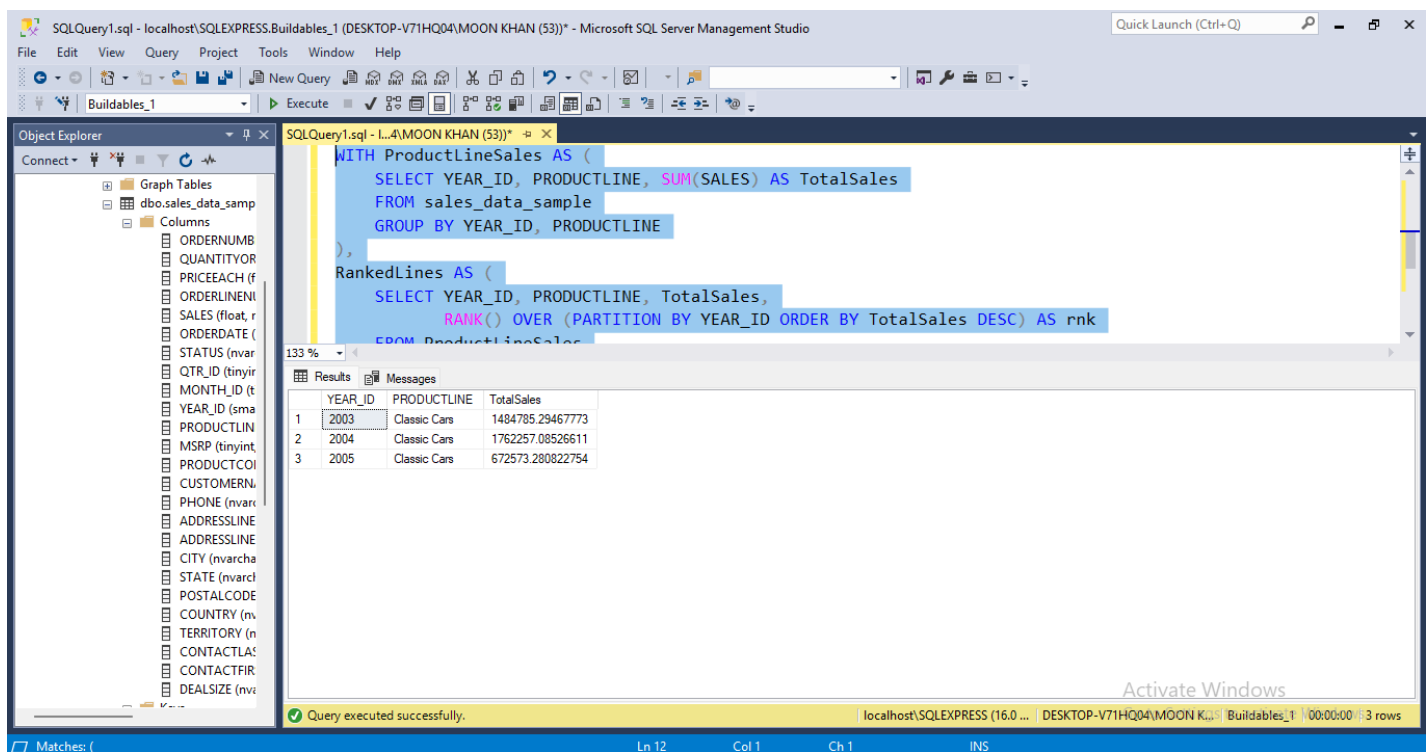
SELECT YEAR_ID, PRODUCTLINE, TotalSales

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.

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

```

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;

```

The Results pane displays the following data:

	CUSTOMERNAME	TotalSales	YEAR_ID	MONTH_ID	MonthlySales	TopProductLine
1	Euro Shopping Channel	912294.110473633	2003	1	129753.599731445	Classic Cars
2	Mini Gifts Distributors Ltd.	654858.058105469	2003	1	129753.599731445	Classic Cars
3	Australian Collectors. Co.	200995.41015625	2003	1	129753.599731445	Classic Cars
4	Muscle Machine Inc	197736.940185547	2003	1	129753.599731445	Classic Cars
5	La Rochelle Gifts	180124.899719238	2003	1	129753.599731445	Classic Cars
6	Dragon Souvenirs. Ltd.	172989.680541992	2003	1	129753.599731445	Classic Cars
7	Land of Toys Inc.	164069.439331055	2003	1	129753.599731445	Classic Cars
8	The Sharp Gifts Warehouse	160010.270263672	2003	1	129753.599731445	Classic Cars
9	AV Stores. Co.	157807.809631348	2003	1	129753.599731445	Classic Cars
10	Anna's Decorations. Ltd	153996.129150391	2003	1	129753.599731445	Classic Cars
11	Souvenirs And Things Co.	151570.979858398	2003	1	129753.599731445	Classic Cars
12	Corporate Gift Ideas Co.	149882.500244141	2003	1	129753.599731445	Classic Cars
13	Salzburg Collectables	149798.630187988	2003	1	129753.599731445	Classic Cars
14	Danish Wholesale Imports	145041.600830078	2003	1	129753.599731445	Classic Cars
15	Savaleu & Harriot. Co	142874.25	2003	1	129753.599731445	Classic Cars

The status bar at the bottom indicates: "Query executed successfully. | localhost\SQLEXPRESS (16.0 ... | DESKTOP-V71HQ04\MOON KHAN | Buildables_1 | 00:00:00 | 2,208 rows