KULLIYYAH OF INFORMATION AND COMMUNICATION TECHNOLOGY

DEPARTMENT OF INFORMATION SYSTEMS

INFO 4311 – Data Warehousing

Semester 2 2021/2022

Data Profiling and High-Level Data Warehouse Design Project

Section : 1

Team Name: Group C

Group Members

Member’s Name	Matric No.
Muhammad Aniq Haris	1920041
Rafi Mirza Rubayed Hasan	1837613
Islam Md Rafiqul	1823295
Joyaddar Md Jobayer	1731833

1. Value Distribution		• Haris (All)
1. Statistics		• Haris (All)
1. Length Distribution		• Rafiqul (All)
1. Null Ratio		• Jobayer (All)
1. Custom SQL Statement		• Haris (All)
1. Data Warehouse Bus Matrix		• Haris (StarSchema, BusMatrix) • Jobayer • Rafiqul • Rubayed

1. INTRODUCTION

Large retail & commerce stores have long been in a fierce competition among one another especially with the on-growing demand of household goods, food items, clothing, and a wide variety of other industries. With such tremendous amount of data available within their reach, it's important to analyse it and to develop association methods along with it to further understand which products best suit their location, categories, and customer segments. By doing so, losses could be avoided, and a better comprehension of the customer purchasing behaviour is achieved.

The objective of this project is to comprehend what exactly the data set is about, identify key relationships between each attribute, and to perform complex SQL Queries on-demand. Such process of examining, analysing, and crafting useful summaries of data is known as data profiling. By profiling data for the superstore used in our example here, we'd be able to yield a high-level overview which can eliminate costly errors such as data quality issues, risks, and values outside the normal range. All these factors are made to align with the business's standards & goals, and in turn produces critical insights into data that the superstore can leverage to their advantage.

1. DATA SETS

1. Description
   - The sample data taken is from a superstore giant's data set. The data set include the details of every individual orders.
   - The metadata for the dataset include the following columns:

COLUMNS	METADATA
RowID	The unique ID for each row
OrderID	The unique ID for each order
Order Date	Order Date of the product.
Ship Date	The product's shipping date.
Ship Mode	Customer-specified shipping method
Customer ID	Unique ID to identify each Customer.
Customer Name	Name Of the Customer
Segment	The segment where the Customer belongs.
Country	Customer's country of residence.
City	The Customer's city of residence.
State	Customer's state of residence
Postal Code	Every customer's postal code.
Region	Region where the Customer belong.
Product ID	Unique ID of the Product.
Category	Category of the product ordered.
Sub-Category	Sub-Category of the product ordered.
Product Name	Name of the Product
Sales	Sales of the Product.
Quantity	Quantity of the Product.
Discount	Discount provided.
Profit	Profit/Loss that was made.

1. Reference to Data Set
   - View Data – Orders Table
     - ○ Superstore Dataset
   - External Link – Sourced from Kaggle
     - ○ https://www.kaggle.com/datasets/vivek468/superstore-dataset-final

1. Data Set Tables
   -

Data Source	Number of Rows	Natural Key(s)	Each row represents
Superstore Dataset	**9994** rows, 20 columns	OrderID, CustomerID, Product ID	An individual order

1. DATA PROFILING

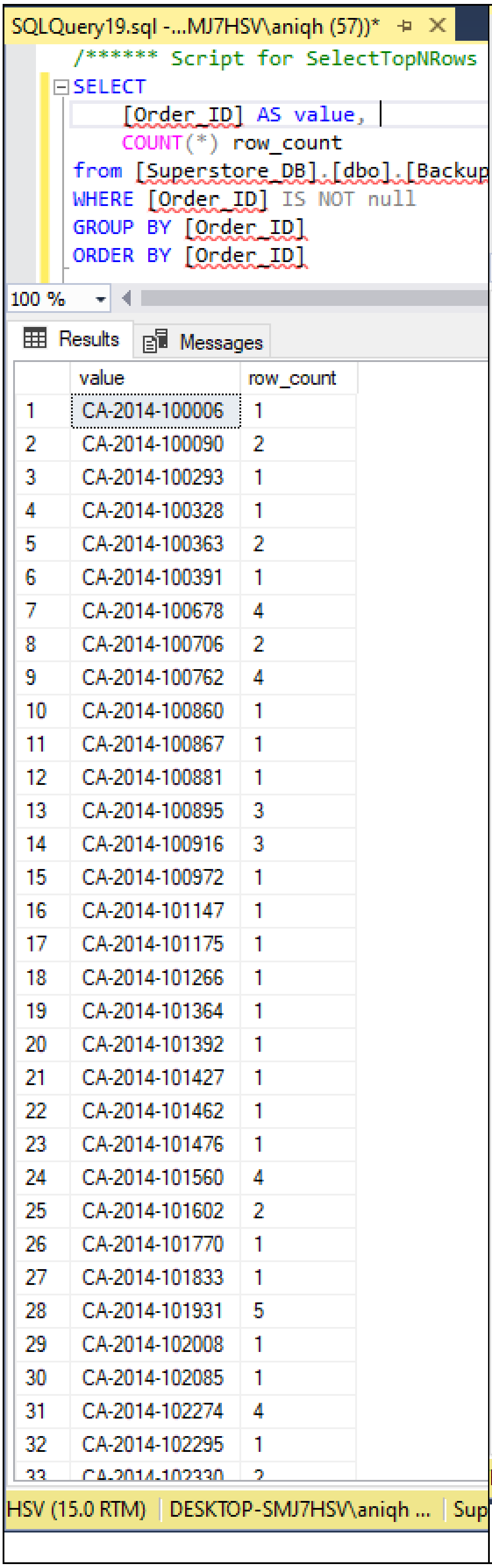
1.

Use SQL queries to run a profile on your data set (for each column) following the statements below. Show the SQL queries and the results. You may compile the results for the columns in the table(s) and present them in the best manner possible.

1. **Value Distribution -** Find the number of times each value in the column occurs (also show the value). This profile helps you identify problems in your data, such as incorrect number of distinct values in a column. For example, you profile a column that is supposed to contain states in Malaysia and discover more than 13 distinct values. (*3 points*)
   - SQL Statement Template:

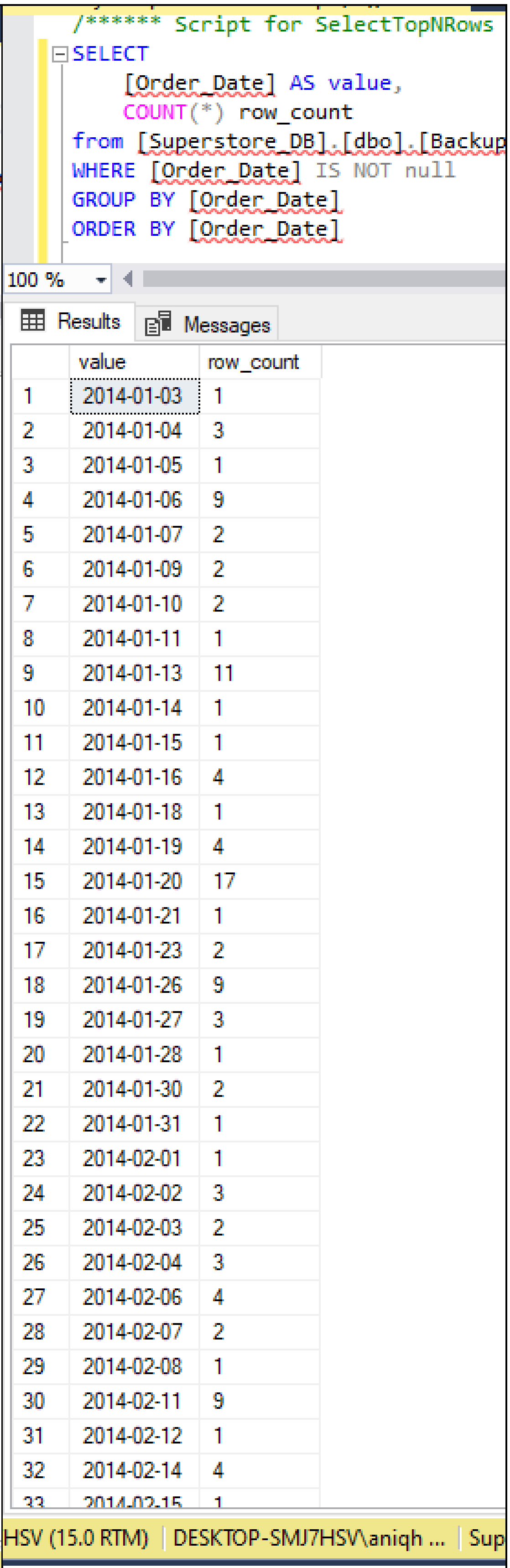
```
SELECT
    column AS value,
    COUNT(*) row_count
from table
WHERE column IS NOT null
GROUP BY column
ORDER BY column;
```

Order_ID	Order_Date
• 5,009 rows selected – Indicates the number of items bought in the same order.	• 1,237 rows selected – Indicates the number of orders/items made during that day(date).


SQLQuery19.sql -...MJ7HSV\aniqh (57))*
100 %
Results
Messages
HSV (15.0 RTM) | DESKTOP-SMJ7HSV\aniqh ... | Sup

```
/****** Script for SelectTopNRows
SELECT
      [Order_ID] AS value,
      COUNT(*) row_count
from [Superstore_DB].[dbo].[Backup
WHERE [Order_ID] IS NOT null
GROUP BY [Order_ID]
ORDER BY [Order_ID]
```

	value	row_count
1	CA-2014-100006	1
2	CA-2014-100090	2
3	CA-2014-100293	1
4	CA-2014-100328	1
5	CA-2014-100363	2
6	CA-2014-100391	1
7	CA-2014-100678	4
8	CA-2014-100706	2
9	CA-2014-100762	4
10	CA-2014-100860	1
11	CA-2014-100867	1
12	CA-2014-100881	1
13	CA-2014-100895	3
14	CA-2014-100916	3
15	CA-2014-100972	1
16	CA-2014-101147	1
17	CA-2014-101175	1
18	CA-2014-101266	1
19	CA-2014-101364	1
20	CA-2014-101392	1
21	CA-2014-101427	1
22	CA-2014-101462	1
23	CA-2014-101476	1
24	CA-2014-101560	4
25	CA-2014-101602	2
26	CA-2014-101770	1
27	CA-2014-101833	1
28	CA-2014-101931	5
29	CA-2014-102008	1
30	CA-2014-102085	1
31	CA-2014-102274	4
32	CA-2014-102295	1
33	CA-2014-102330	2

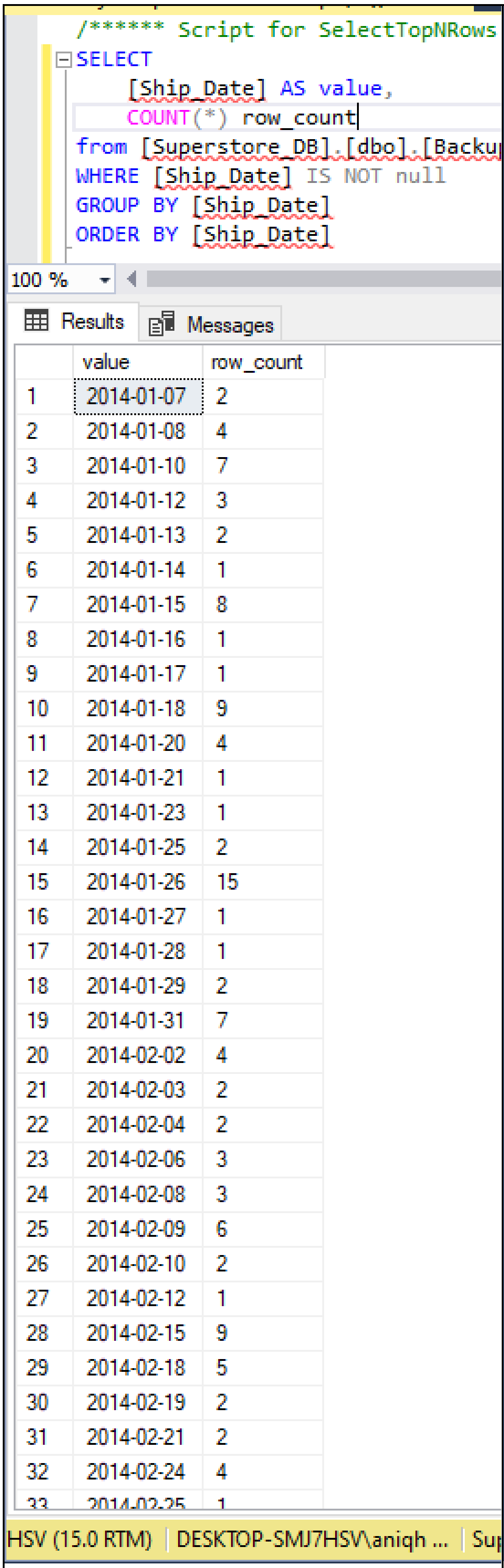


```
/****** Script for SelectTopNRows
SELECT
      [Order_Date] AS value,
      COUNT(*) row_count
from [Superstore_DB].[dbo].[Backup
WHERE [Order_Date] IS NOT null
GROUP BY [Order_Date]
ORDER BY [Order_Date]
```

	value	row_count
1	2014-01-03	1
2	2014-01-04	3
3	2014-01-05	1
4	2014-01-06	9
5	2014-01-07	2
6	2014-01-09	2
7	2014-01-10	2
8	2014-01-11	1
9	2014-01-13	11
10	2014-01-14	1
11	2014-01-15	1
12	2014-01-16	4
13	2014-01-18	1
14	2014-01-19	4
15	2014-01-20	17
16	2014-01-21	1
17	2014-01-23	2
18	2014-01-26	9
19	2014-01-27	3
20	2014-01-28	1
21	2014-01-30	2
22	2014-01-31	1
23	2014-02-01	1
24	2014-02-02	3
25	2014-02-03	2
26	2014-02-04	3
27	2014-02-06	4
28	2014-02-07	2
29	2014-02-08	1
30	2014-02-11	9
31	2014-02-12	1
32	2014-02-14	4
33	2014-02-15	1

Ship_Date	Ship_Mode

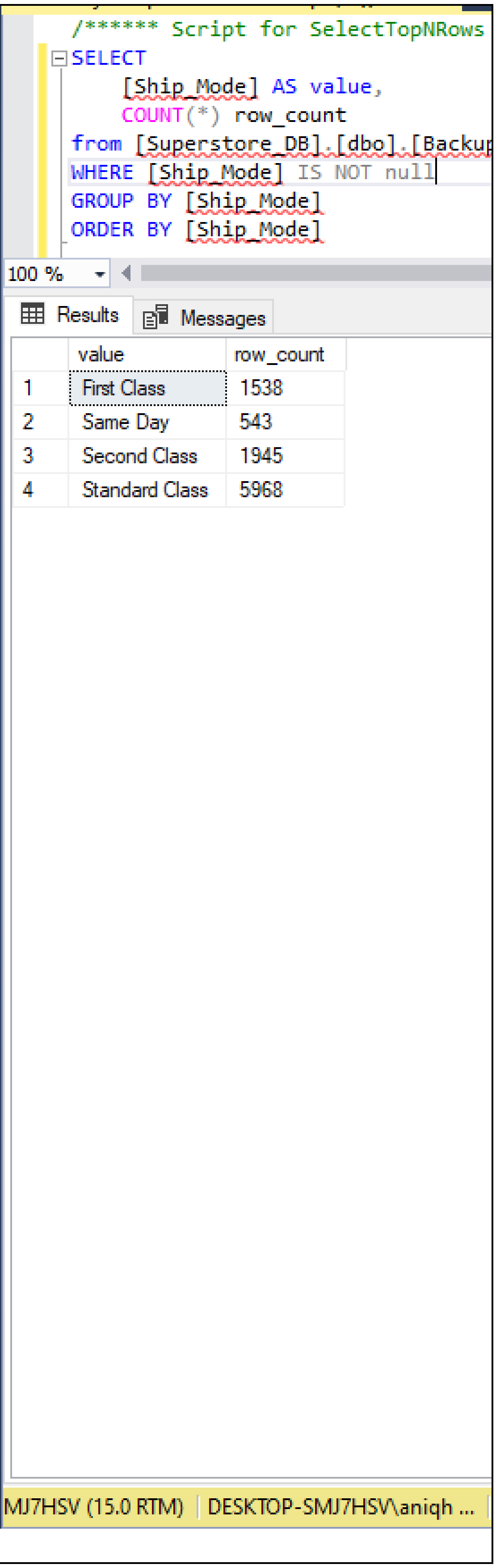
• 1,334 rows selected – Indicates the number of orders/items shipped out on a certain day (date).	• 4 rows selected – Indicates the number of orders/items shipped out using a specific shipping mode.



```
/****** Script for SelectTopNRows
SELECT
      [Ship_Date] AS value,
      COUNT(*) row_count
from [Superstore_DB].[dbo].[Backu
WHERE [Ship_Date] IS NOT null
GROUP BY [Ship_Date]
ORDER BY [Ship_Date]
```

	value	row_count
1	2014-01-07	2
2	2014-01-08	4
3	2014-01-10	7
4	2014-01-12	3
5	2014-01-13	2
6	2014-01-14	1
7	2014-01-15	8
8	2014-01-16	1
9	2014-01-17	1
10	2014-01-18	9
11	2014-01-20	4
12	2014-01-21	1
13	2014-01-23	1
14	2014-01-25	2
15	2014-01-26	15
16	2014-01-27	1
17	2014-01-28	1
18	2014-01-29	2
19	2014-01-31	7
20	2014-02-02	4
21	2014-02-03	2
22	2014-02-04	2
23	2014-02-06	3
24	2014-02-08	3
25	2014-02-09	6
26	2014-02-10	2
27	2014-02-12	1
28	2014-02-15	9
29	2014-02-18	5
30	2014-02-19	2
31	2014-02-21	2
32	2014-02-24	4
33	2014-02-25	1

HSV (15.0 RTM) | DESKTOP-SMJ7HSV\aniqh ... | Su



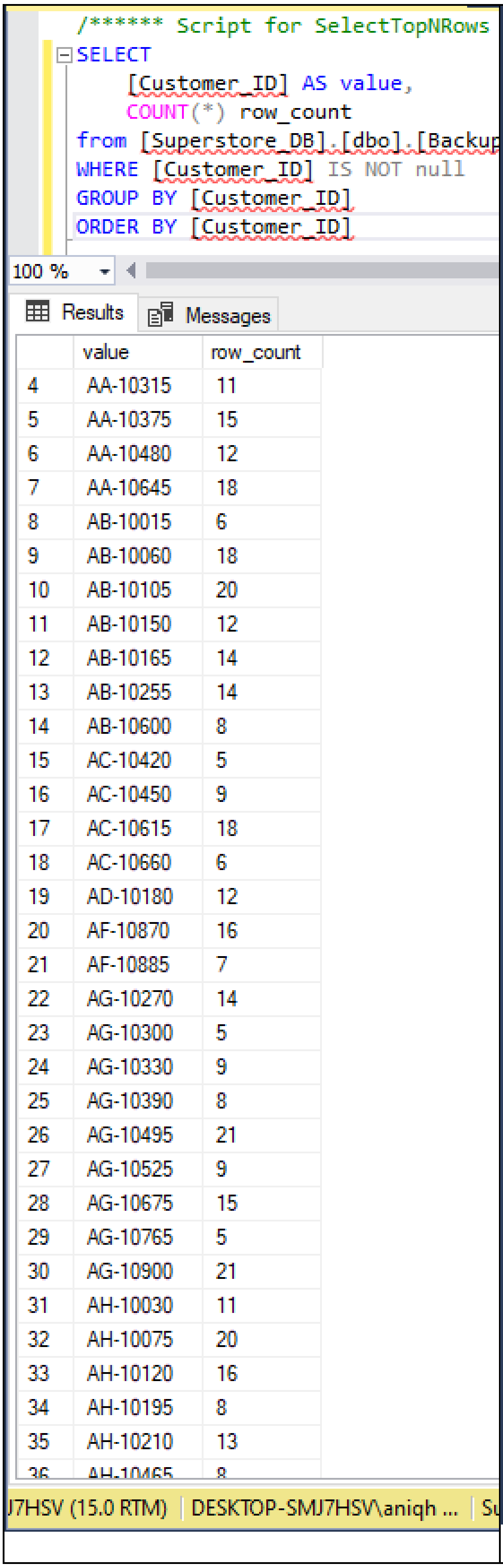
```
/****** Script for SelectTopNRows
SELECT
      [Ship_Mode] AS value,
      COUNT(*) row_count
from [Superstore_DB].[dbo].[Backup
WHERE [Ship_Mode] IS NOT null
GROUP BY [Ship_Mode]
ORDER BY [Ship_Mode]
```

	value	row_count
1	First Class	1538
2	Same Day	543
3	Second Class	1945
4	Standard Class	5968

MJ7HSV (15.0 RTM) | DESKTOP-SMJ7HSV\aniqh ... |

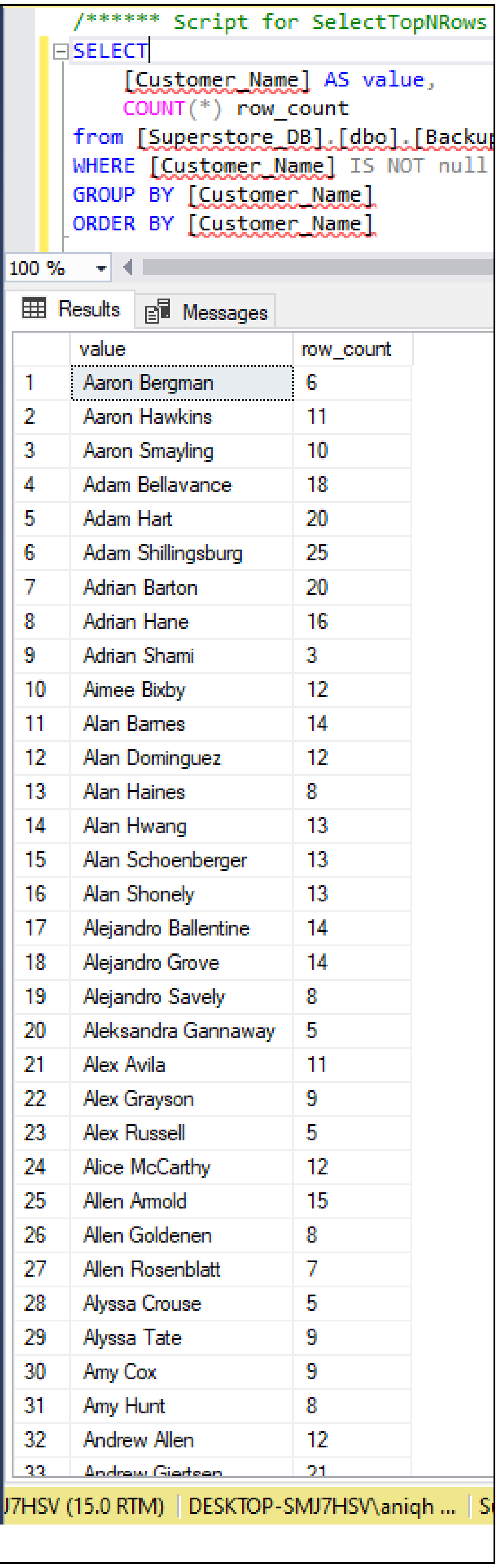
Customer_ID	Customer_Name

• 793 rows selected – Indicates the number of orders/items bought by a customer.	• 793 rows selected – Indicates the number of orders/items bought by a customer specified by his/her name.



```sql
/****** Script for SelectTopNRows
SELECT
      [Customer_ID] AS value,
      COUNT(*) row_count
from [Superstore_DB].[dbo].[Backup
WHERE [Customer_ID] IS NOT null
GROUP BY [Customer_ID]
ORDER BY [Customer_ID]
```

	value	row_count
4	AA-10315	11
5	AA-10375	15
6	AA-10480	12
7	AA-10645	18
8	AB-10015	6
9	AB-10060	18
10	AB-10105	20
11	AB-10150	12
12	AB-10165	14
13	AB-10255	14
14	AB-10600	8
15	AC-10420	5
16	AC-10450	9
17	AC-10615	18
18	AC-10660	6
19	AD-10180	12
20	AF-10870	16
21	AF-10885	7
22	AG-10270	14
23	AG-10300	5
24	AG-10330	9
25	AG-10390	8
26	AG-10495	21
27	AG-10525	9
28	AG-10675	15
29	AG-10765	5
30	AG-10900	21
31	AH-10030	11
32	AH-10075	20
33	AH-10120	16
34	AH-10195	8
35	AH-10210	13
36	AH-10465	8

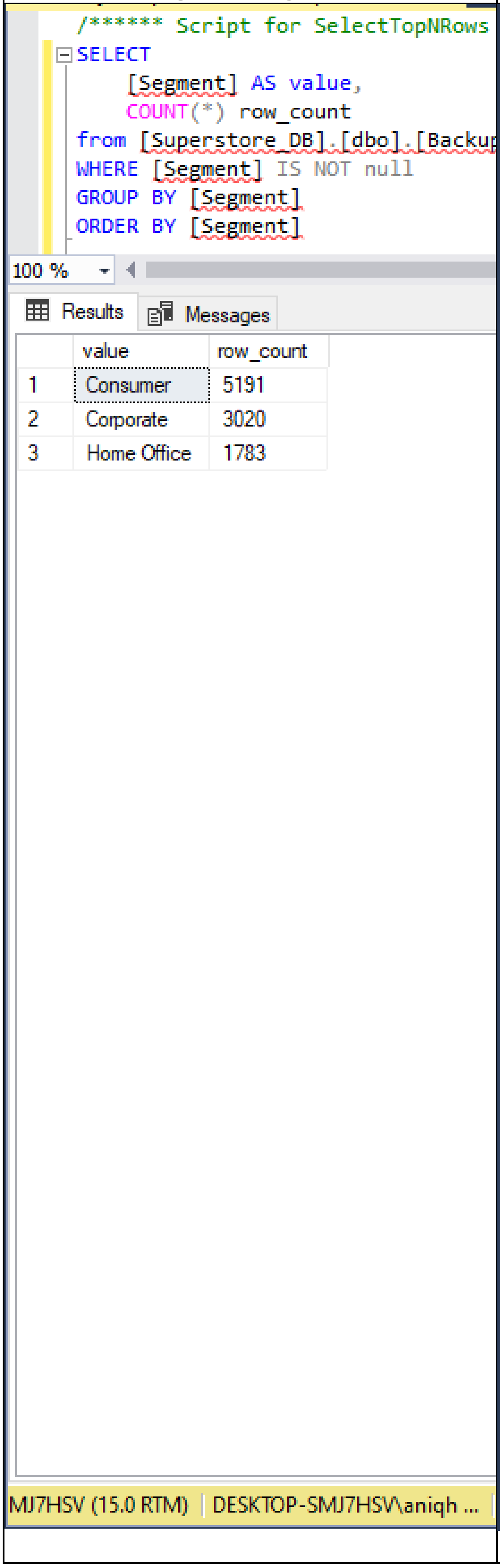


```sql
/****** Script for SelectTopNRows
SELECT
      [Customer_Name] AS value,
      COUNT(*) row_count
from [Superstore_DB].[dbo].[Backu
WHERE [Customer_Name] IS NOT null
GROUP BY [Customer_Name]
ORDER BY [Customer_Name]
```

	value	row_count
1	Aaron Bergman	6
2	Aaron Hawkins	11
3	Aaron Smayling	10
4	Adam Bellavance	18
5	Adam Hart	20
6	Adam Shillingsburg	25
7	Adrian Barton	20
8	Adrian Hane	16
9	Adrian Shami	3
10	Aimee Bixby	12
11	Alan Barnes	14
12	Alan Dominguez	12
13	Alan Haines	8
14	Alan Hwang	13
15	Alan Schoenberger	13
16	Alan Shonely	13
17	Alejandro Ballentine	14
18	Alejandro Grove	14
19	Alejandro Savely	8
20	Aleksandra Gannaway	5
21	Alex Avila	11
22	Alex Grayson	9
23	Alex Russell	5
24	Alice McCarthy	12
25	Allen Armold	15
26	Allen Goldenen	8
27	Allen Rosenblatt	7
28	Alyssa Crouse	5
29	Alyssa Tate	9
30	Amy Cox	9
31	Amy Hunt	8
32	Andrew Allen	12
33	Andrew Gjertsen	21

Segment	Country

- 3 rows selected – Indicates the number of orders/items bought in a segment.

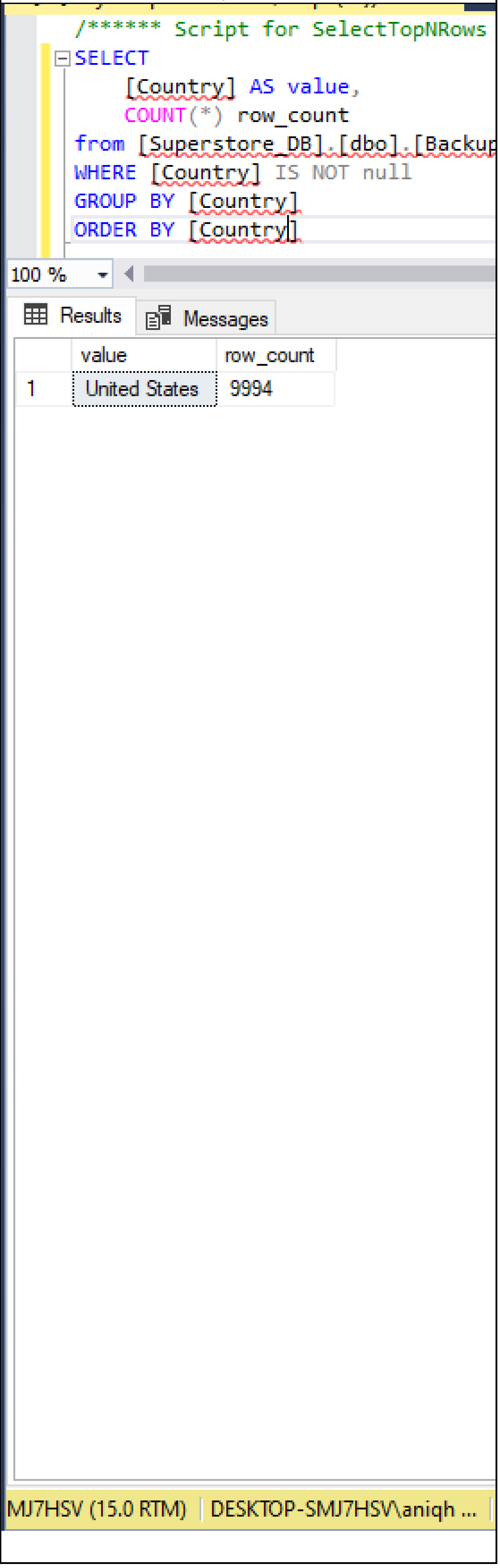


```
/****** Script for SelectTopNRows
SELECT
      [Segment] AS value,
      COUNT(*) row_count
from [Superstore_DB].[dbo].[Backup
WHERE [Segment] IS NOT null
GROUP BY [Segment]
ORDER BY [Segment]
```

	value	row_count
1	Consumer	5191
2	Corporate	3020
3	Home Office	1783

MJ7HSV (15.0 RTM) | DESKTOP-SMJ7HSV\aniqh ...

- 1 row selected – Indicates the number of orders/items bought in a country.

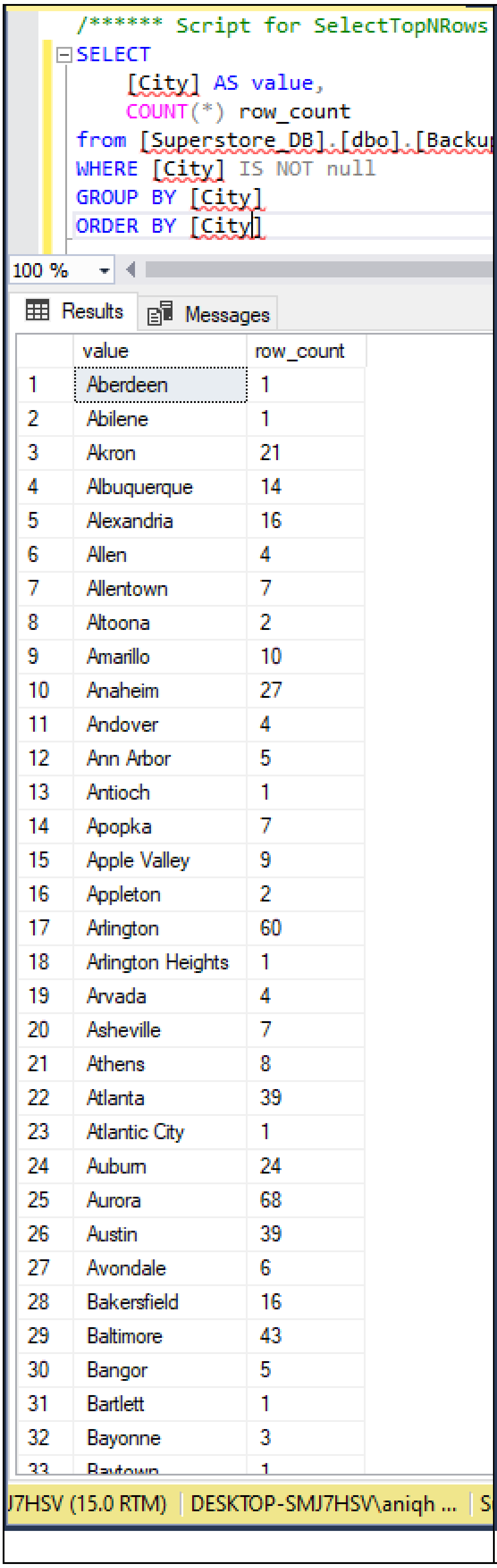


```
/****** Script for SelectTopNRows
SELECT
      [Country] AS value,
      COUNT(*) row_count
from [Superstore_DB].[dbo].[Backup
WHERE [Country] IS NOT null
GROUP BY [Country]
ORDER BY [Country]
```

	value	row_count
1	United States	9994

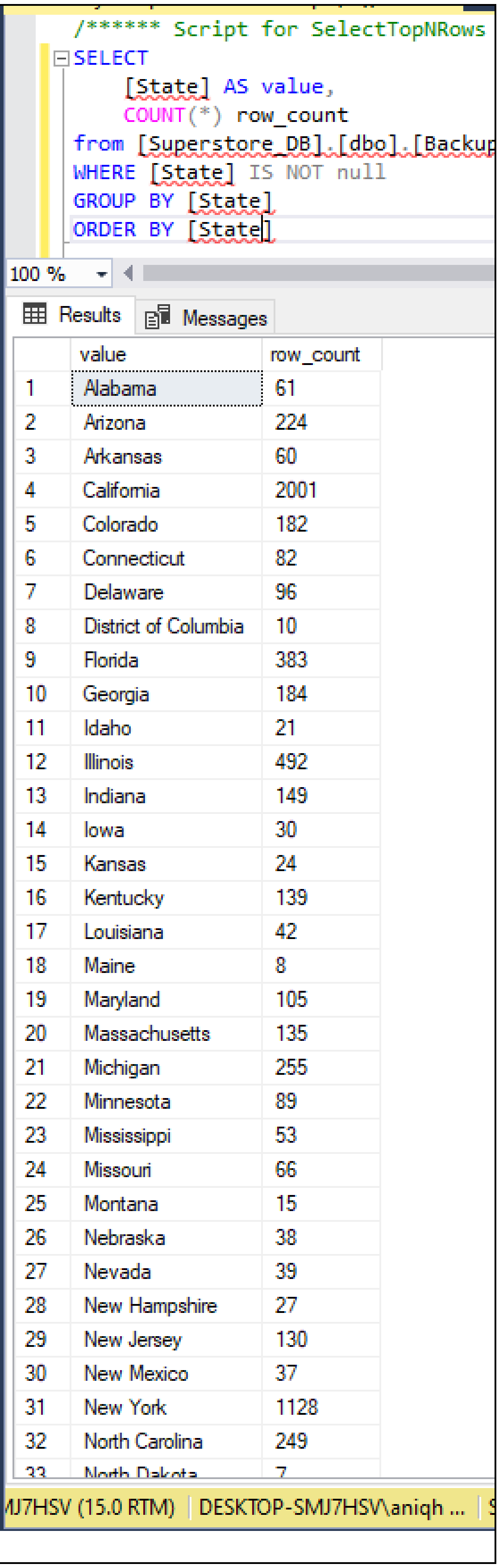
MJ7HSV (15.0 RTM) | DESKTOP-SMJ7HSV\aniqh ...

City	State
• 531 rows selected – Indicates the number of orders/items bought in a city.	• 49 rows selected – Indicates the number of orders/items bought in a state.



```
/****** Script for SelectTopNRows
SELECT
      [City] AS value,
      COUNT(*) row_count
from [Superstore_DB].[dbo].[Backu
WHERE [City] IS NOT null
GROUP BY [City]
ORDER BY [City]
```

	value	row_count
1	Aberdeen	1
2	Abilene	1
3	Akron	21
4	Albuquerque	14
5	Alexandria	16
6	Allen	4
7	Allentown	7
8	Altoona	2
9	Amarillo	10
10	Anaheim	27
11	Andover	4
12	Ann Arbor	5
13	Antioch	1
14	Apopka	7
15	Apple Valley	9
16	Appleton	2
17	Arlington	60
18	Arlington Heights	1
19	Arvada	4
20	Asheville	7
21	Athens	8
22	Atlanta	39
23	Atlantic City	1
24	Auburn	24
25	Aurora	68
26	Austin	39
27	Avondale	6
28	Bakersfield	16
29	Baltimore	43
30	Bangor	5
31	Bartlett	1
32	Bayonne	3
33	Baytown	1

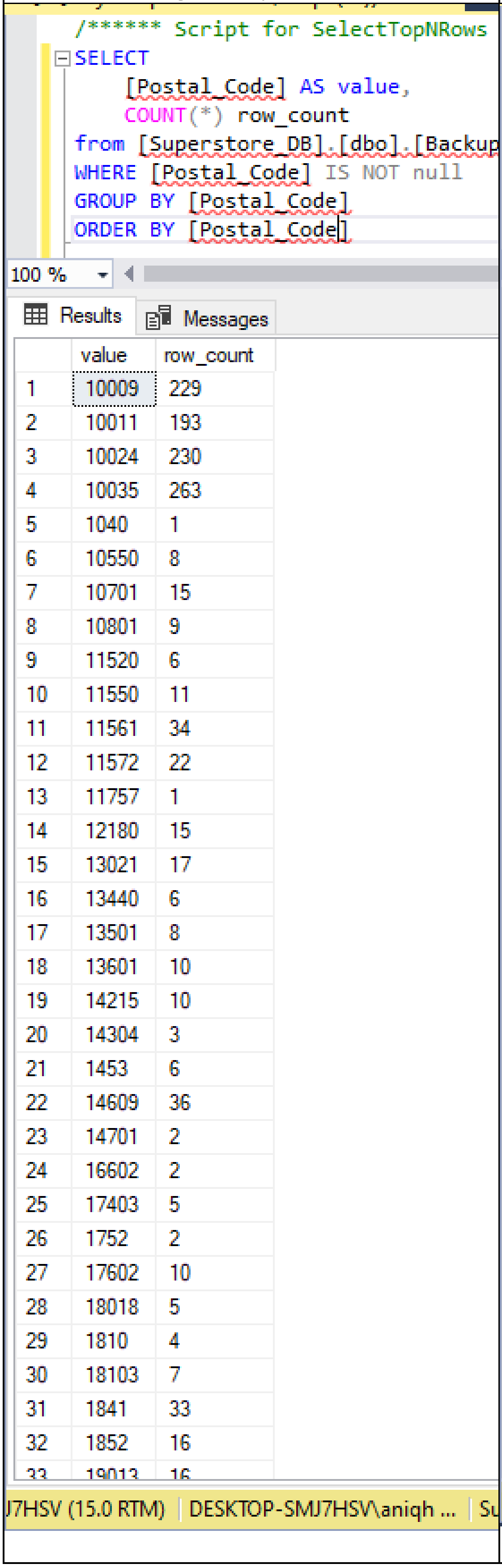


```
/****** Script for SelectTopNRows
SELECT
      [State] AS value,
      COUNT(*) row_count
from [Superstore_DB].[dbo].[Backup
WHERE [State] IS NOT null
GROUP BY [State]
ORDER BY [State]
```

	value	row_count
1	Alabama	61
2	Arizona	224
3	Arkansas	60
4	California	2001
5	Colorado	182
6	Connecticut	82
7	Delaware	96
8	District of Columbia	10
9	Florida	383
10	Georgia	184
11	Idaho	21
12	Illinois	492
13	Indiana	149
14	Iowa	30
15	Kansas	24
16	Kentucky	139
17	Louisiana	42
18	Maine	8
19	Maryland	105
20	Massachusetts	135
21	Michigan	255
22	Minnesota	89
23	Mississippi	53
24	Missouri	66
25	Montana	15
26	Nebraska	38
27	Nevada	39
28	New Hampshire	27
29	New Jersey	130
30	New Mexico	37
31	New York	1128
32	North Carolina	249
33	North Dakota	7

Postal_Code	Region

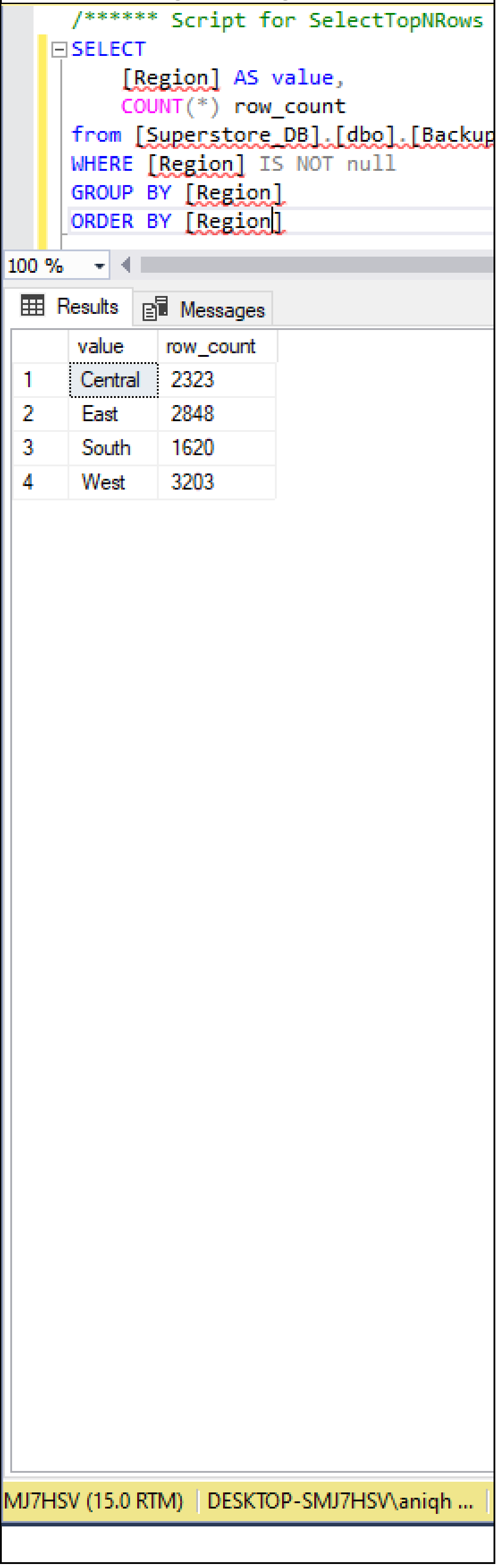
- 631 rows selected – Indicates the number of orders/items bought in a postal code sector.



```
/****** Script for SelectTopNRows
SELECT
      [Postal_Code] AS value,
      COUNT(*) row_count
from [Superstore_DB].[dbo].[Backup
WHERE [Postal_Code] IS NOT null
GROUP BY [Postal_Code]
ORDER BY [Postal_Code]
```

	value	row_count
1	10009	229
2	10011	193
3	10024	230
4	10035	263
5	1040	1
6	10550	8
7	10701	15
8	10801	9
9	11520	6
10	11550	11
11	11561	34
12	11572	22
13	11757	1
14	12180	15
15	13021	17
16	13440	6
17	13501	8
18	13601	10
19	14215	10
20	14304	3
21	1453	6
22	14609	36
23	14701	2
24	16602	2
25	17403	5
26	1752	2
27	17602	10
28	18018	5
29	1810	4
30	18103	7
31	1841	33
32	1852	16
33	19013	16

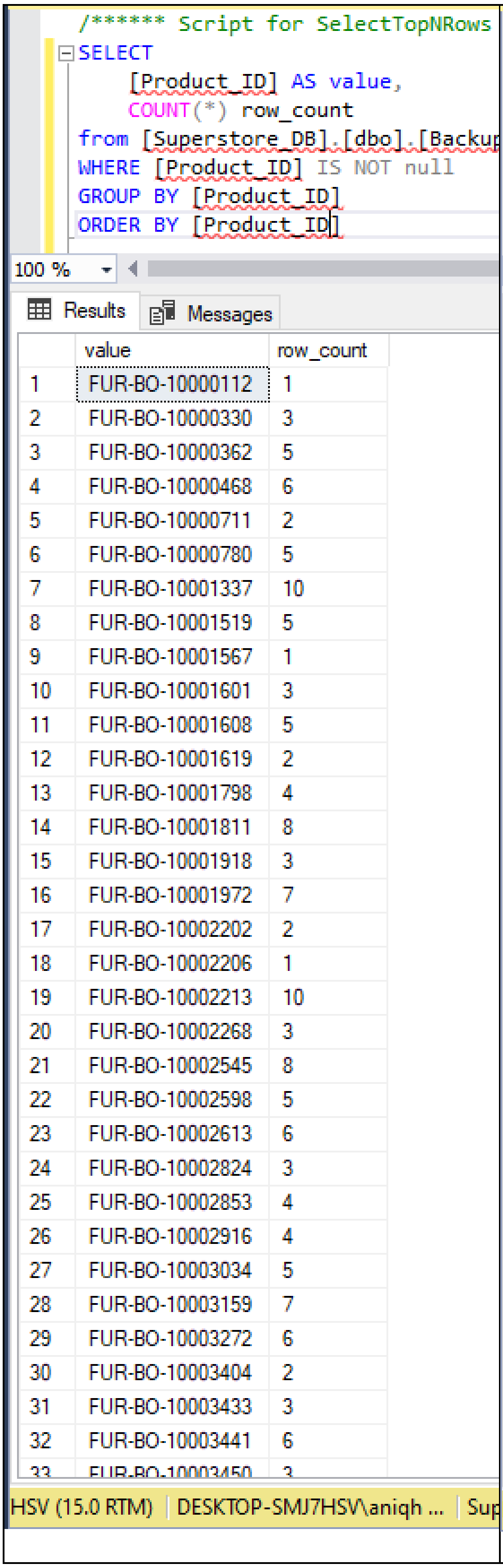
- 4 rows selected – Indicates the number of orders/items bought in a region.



```
/****** Script for SelectTopNRows
SELECT
      [Region] AS value,
      COUNT(*) row_count
from [Superstore_DB].[dbo].[Backup
WHERE [Region] IS NOT null
GROUP BY [Region]
ORDER BY [Region]
```

	value	row_count
1	Central	2323
2	East	2848
3	South	1620
4	West	3203

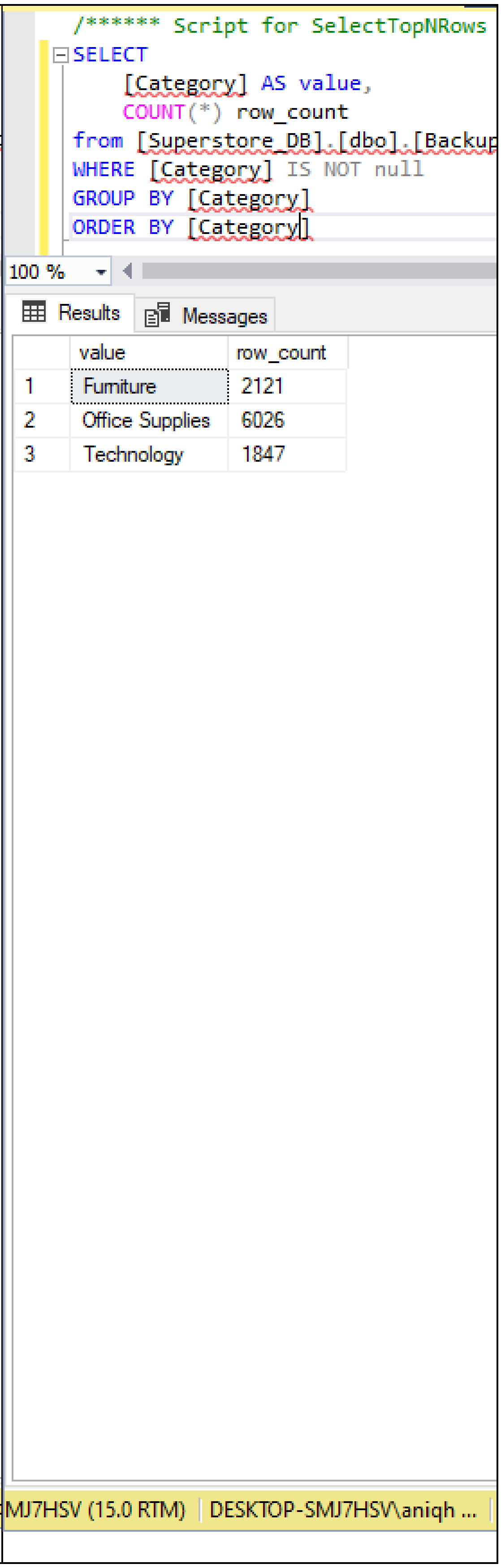
Product_ID	Category
• 1,862 rows selected – Indicates the number of times the item was included in an order.	• 3 rows selected – Indicates the number of orders/items bought in a category.



```sql
/****** Script for SelectTopNRows
SELECT
      [Product_ID] AS value,
      COUNT(*) row_count
from [Superstore_DB].[dbo].[Backup
WHERE [Product_ID] IS NOT null
GROUP BY [Product_ID]
ORDER BY [Product_ID]
```

	value	row_count
1	FUR-BO-10000112	1
2	FUR-BO-10000330	3
3	FUR-BO-10000362	5
4	FUR-BO-10000468	6
5	FUR-BO-10000711	2
6	FUR-BO-10000780	5
7	FUR-BO-10001337	10
8	FUR-BO-10001519	5
9	FUR-BO-10001567	1
10	FUR-BO-10001601	3
11	FUR-BO-10001608	5
12	FUR-BO-10001619	2
13	FUR-BO-10001798	4
14	FUR-BO-10001811	8
15	FUR-BO-10001918	3
16	FUR-BO-10001972	7
17	FUR-BO-10002202	2
18	FUR-BO-10002206	1
19	FUR-BO-10002213	10
20	FUR-BO-10002268	3
21	FUR-BO-10002545	8
22	FUR-BO-10002598	5
23	FUR-BO-10002613	6
24	FUR-BO-10002824	3
25	FUR-BO-10002853	4
26	FUR-BO-10002916	4
27	FUR-BO-10003034	5
28	FUR-BO-10003159	7
29	FUR-BO-10003272	6
30	FUR-BO-10003404	2
31	FUR-BO-10003433	3
32	FUR-BO-10003441	6
33	FUR-BO-10003450	3

HSV (15.0 RTM) | DESKTOP-SMJ7HSV\aniqh ... | Sup



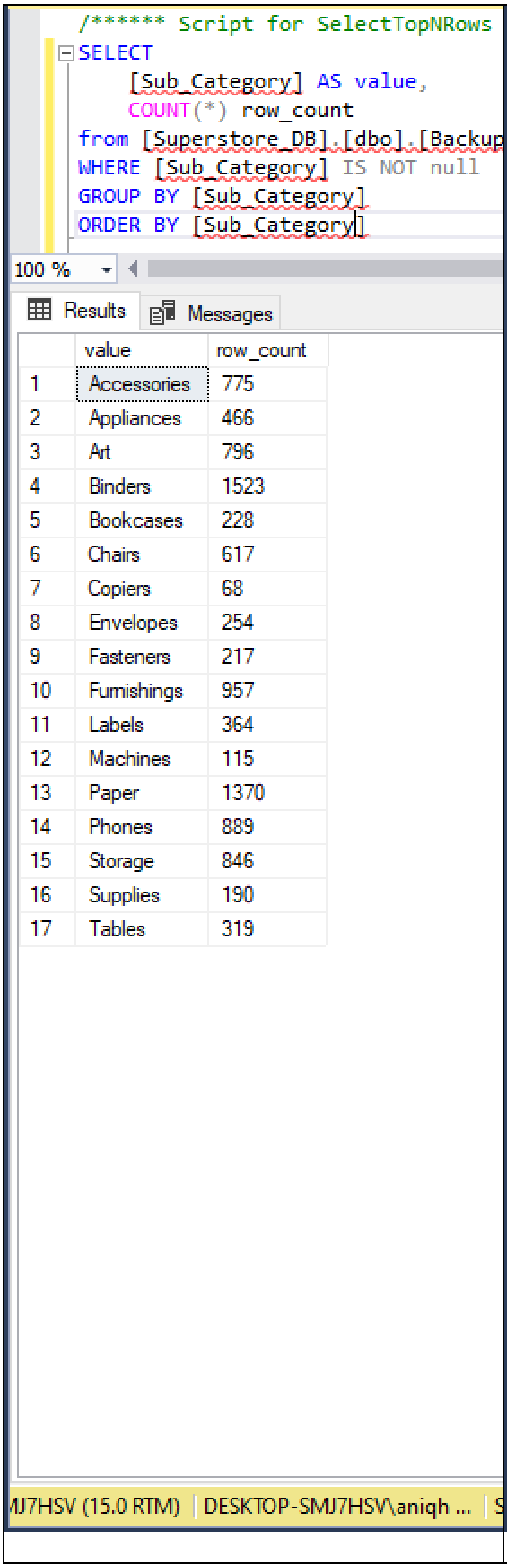
```sql
/****** Script for SelectTopNRows
SELECT
      [Category] AS value,
      COUNT(*) row_count
from [Superstore_DB].[dbo].[Backup
WHERE [Category] IS NOT null
GROUP BY [Category]
ORDER BY [Category]
```

	value	row_count
1	Furniture	2121
2	Office Supplies	6026
3	Technology	1847

MJ7HSV (15.0 RTM) | DESKTOP-SMJ7HSV\aniqh ...

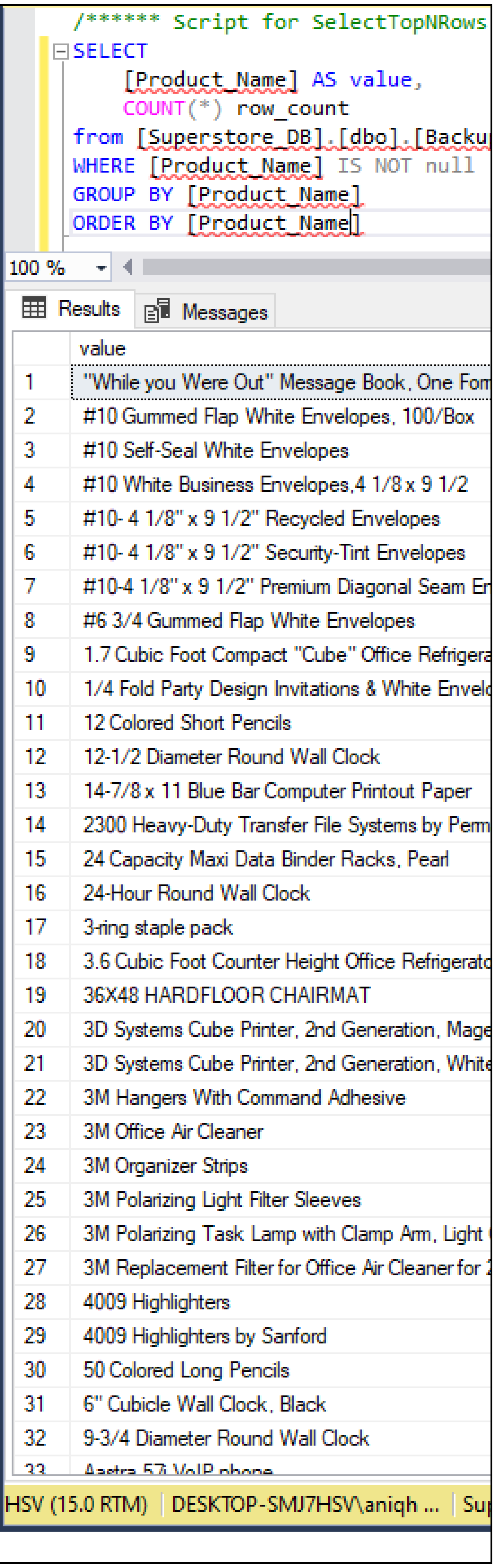
Sub_Category	Product_Name

- 17 rows selected – Indicates the number of orders/items bought in a sub-category.	- 1,850 rows selected – Indicates the number of times the item was included in an order specified by the product name.



```
/****** Script for SelectTopNRows
SELECT
      [Sub_Category] AS value,
      COUNT(*) row_count
from [Superstore_DB].[dbo].[Backup
WHERE [Sub_Category] IS NOT null
GROUP BY [Sub_Category]
ORDER BY [Sub_Category]
```

```
    value        row_count
1   Accessories  775
2   Appliances   466
3   Art          796
4   Binders      1523
5   Bookcases    228
6   Chairs       617
7   Copiers      68
8   Envelopes    254
9   Fasteners    217
10  Furnishings  957
11  Labels       364
12  Machines     115
13  Paper        1370
14  Phones       889
15  Storage      846
16  Supplies     190
17  Tables       319
```

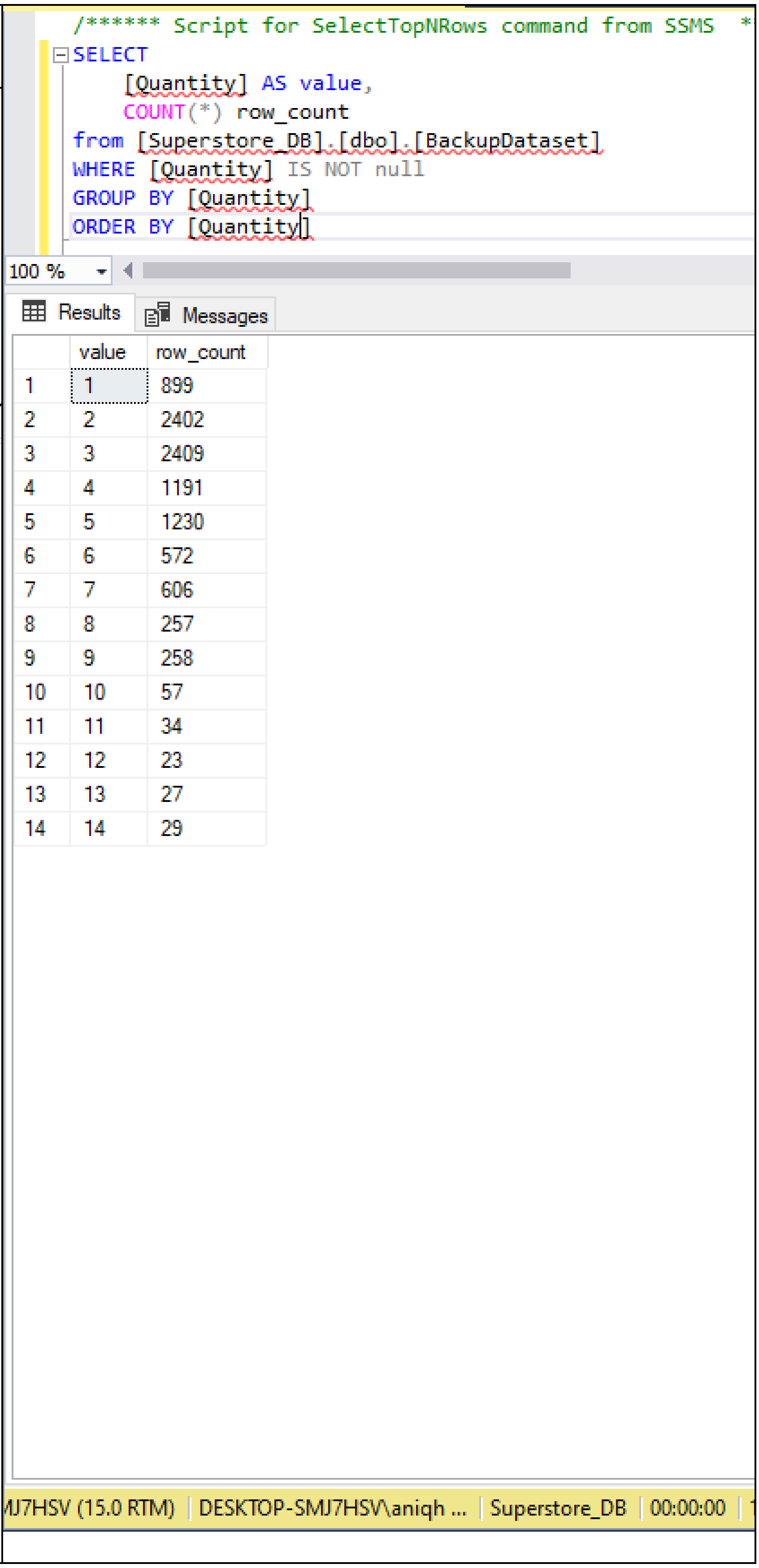


```
/****** Script for SelectTopNRows
SELECT
      [Product_Name] AS value,
      COUNT(*) row_count
from [Superstore_DB].[dbo].[Backu
WHERE [Product_Name] IS NOT null
GROUP BY [Product_Name]
ORDER BY [Product_Name]
```

```
    value
1   "While you Were Out" Message Book, One Forn
2   #10 Gummed Flap White Envelopes, 100/Box
3   #10 Self-Seal White Envelopes
4   #10 White Business Envelopes,4 1/8 x 9 1/2
5   #10- 4 1/8" x 9 1/2" Recycled Envelopes
6   #10- 4 1/8" x 9 1/2" Security-Tint Envelopes
7   #10-4 1/8" x 9 1/2" Premium Diagonal Seam Er
8   #6 3/4 Gummed Flap White Envelopes
9   1.7 Cubic Foot Compact "Cube" Office Refrigera
10  1/4 Fold Party Design Invitations & White Envelo
11  12 Colored Short Pencils
12  12-1/2 Diameter Round Wall Clock
13  14-7/8 x 11 Blue Bar Computer Printout Paper
14  2300 Heavy-Duty Transfer File Systems by Perm
15  24 Capacity Maxi Data Binder Racks, Pearl
16  24-Hour Round Wall Clock
17  3-ring staple pack
18  3.6 Cubic Foot Counter Height Office Refrigerato
19  36X48 HARDFLOOR CHAIRMAT
20  3D Systems Cube Printer, 2nd Generation, Mage
21  3D Systems Cube Printer, 2nd Generation, White
22  3M Hangers With Command Adhesive
23  3M Office Air Cleaner
24  3M Organizer Strips
25  3M Polarizing Light Filter Sleeves
26  3M Polarizing Task Lamp with Clamp Arm, Light
27  3M Replacement Filter for Office Air Cleaner for
28  4009 Highlighters
29  4009 Highlighters by Sanford
30  50 Colored Long Pencils
31  6" Cubicle Wall Clock, Black
32  9-3/4 Diameter Round Wall Clock
33  Aastra 57i VoIP phone
```

Quantity	

- 14 rows selected – Indicates the number of orders/items bought with a quantity of value ***x*** (*x* = quantity).



```
/****** Script for SelectTopNRows command from SSMS  *
SELECT
      [Quantity] AS value,
      COUNT(*) row_count
from [Superstore_DB].[dbo].[BackupDataset]
WHERE [Quantity] IS NOT null
GROUP BY [Quantity]
ORDER BY [Quantity]
```

	value	row_count
1	1	899
2	2	2402
3	3	2409
4	4	1191
5	5	1230
6	6	572
7	7	606
8	8	257
9	9	258
10	10	57
11	11	34
12	12	23
13	13	27
14	14	29

100 %
Results
Messages
IJ7HSV (15.0 RTM) | DESKTOP-SMJ7HSV\aniqh ... | Superstore_DB | 00:00:00

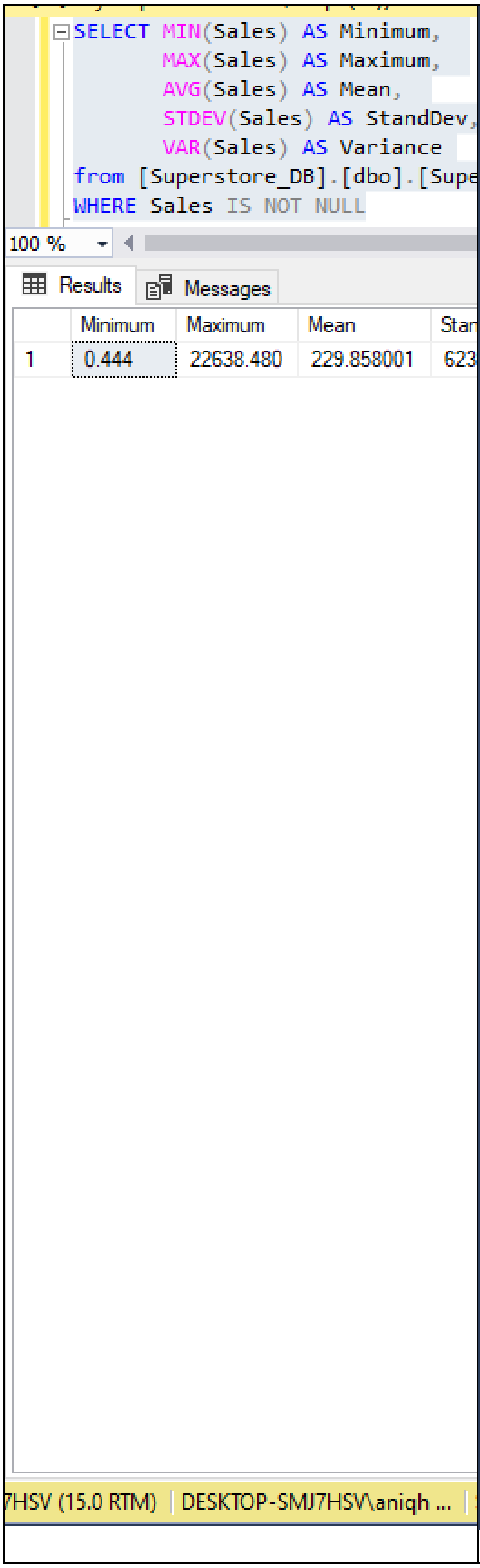
1. **Statistics** – Find the minimum, maximum, average, standard deviation and variance for numeric columns; minimum and maximum for **datetime** columns. This profile helps

you identify problems in your data, such as dates that are not valid. For example, you profile a column of historical dates and discover a maximum date that is in the future. (*3 points*)

- Our number columns:
    - Int/Decimal/Float: *Sales*, *Profit*, *Quantity*, *Discount*
    - Date: *Order_Date*, *Ship_Date*
- SQL Statement Template:

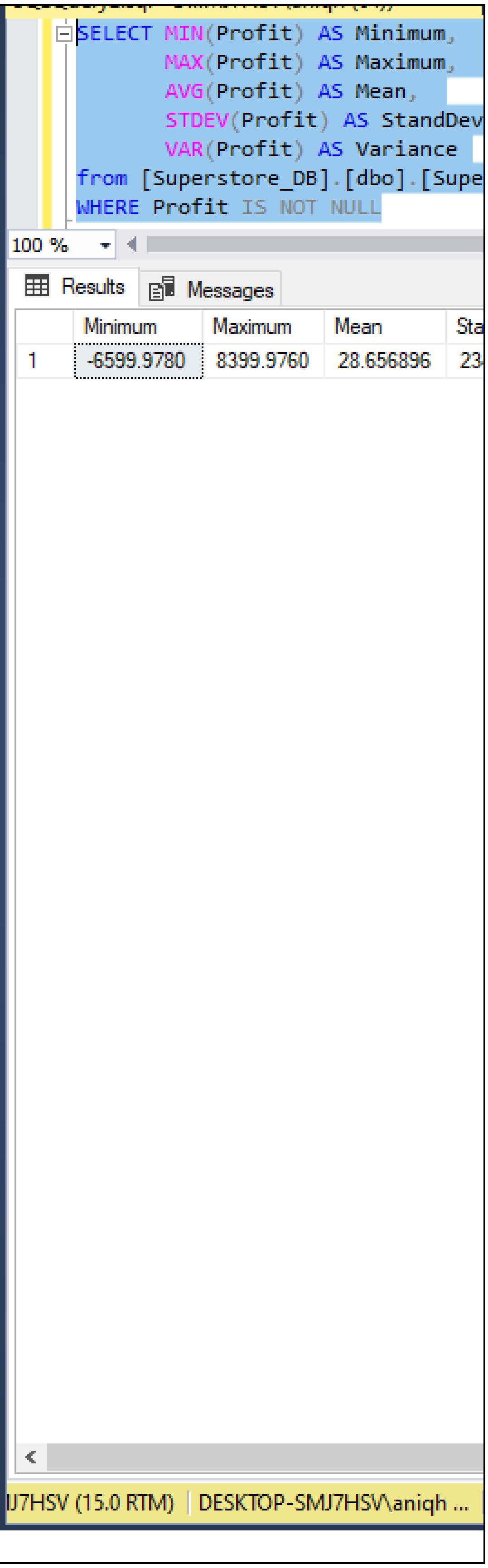
```
SELECT MIN(Column) AS Minimum,
       MAX(Column) AS Maximum,
       AVG(Column) AS Mean,
       STDEV(Column) AS StandDev,
       VAR(Column) AS Variance
from table
WHERE Column IS NOT NULL
```

Sales	Profit



```
SELECT MIN(Sales) AS Minimum,
       MAX(Sales) AS Maximum,
       AVG(Sales) AS Mean,
       STDEV(Sales) AS StandDev,
       VAR(Sales) AS Variance
from [Superstore_DB].[dbo].[Supe
WHERE Sales IS NOT NULL
```

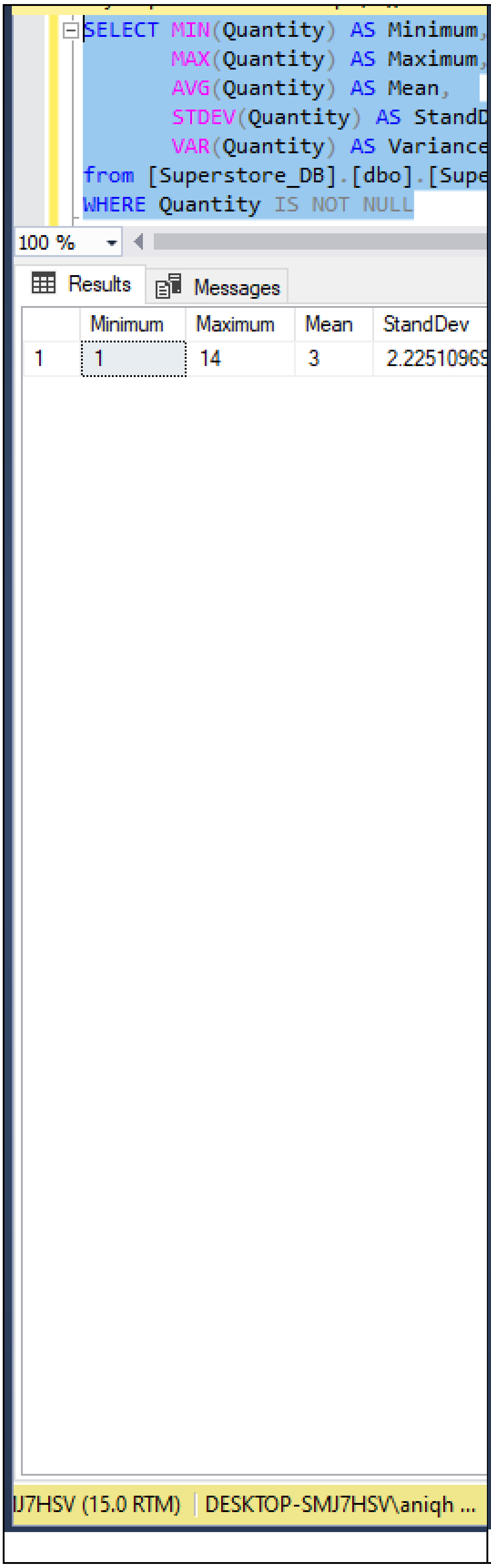
```
   Minimum   Maximum     Mean         Star
1  0.444     22638.480   229.858001   623
```



```
SELECT MIN(Profit) AS Minimum,
       MAX(Profit) AS Maximum,
       AVG(Profit) AS Mean,
       STDEV(Profit) AS StandDev
       VAR(Profit) AS Variance
from [Superstore_DB].[dbo].[Supe
WHERE Profit IS NOT NULL
```

```
   Minimum     Maximum     Mean        Sta
1  -6599.9780  8399.9760   28.656896   23
```

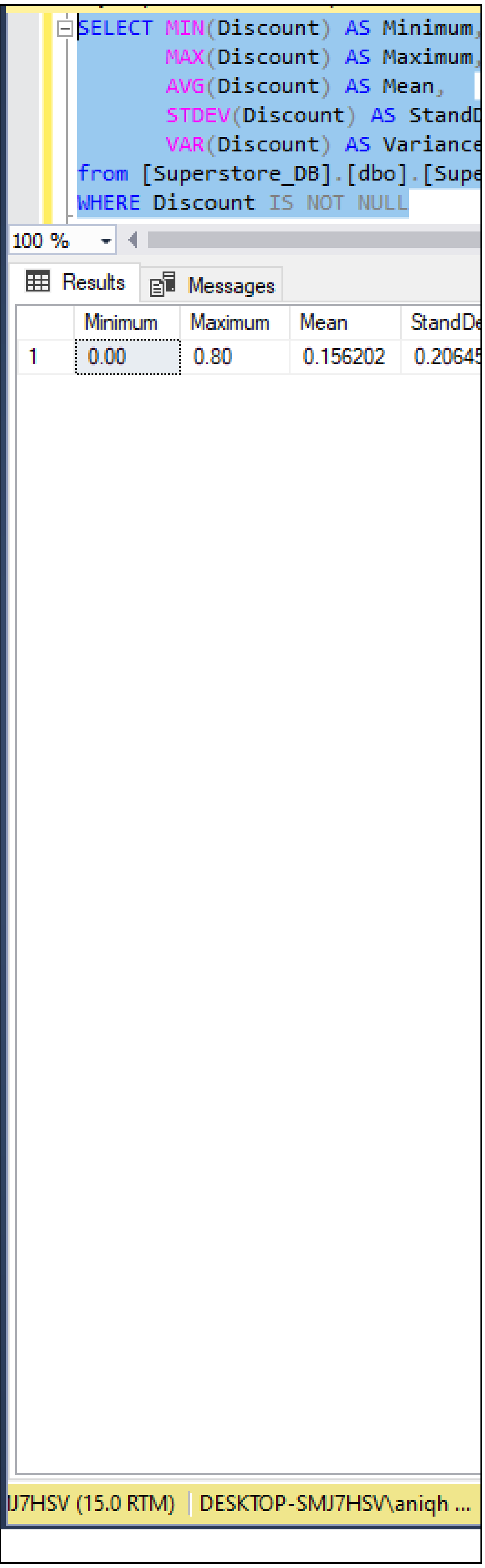
Quantity	Discount



```
SELECT MIN(Quantity) AS Minimum,
       MAX(Quantity) AS Maximum,
       AVG(Quantity) AS Mean,
       STDEV(Quantity) AS StandD
       VAR(Quantity) AS Variance
from [Superstore_DB].[dbo].[Supe
WHERE Quantity IS NOT NULL
```

	Minimum	Maximum	Mean	StandDev
1	1	14	3	2.22510969

J7HSV (15.0 RTM) | DESKTOP-SMJ7HSV\aniqh ...

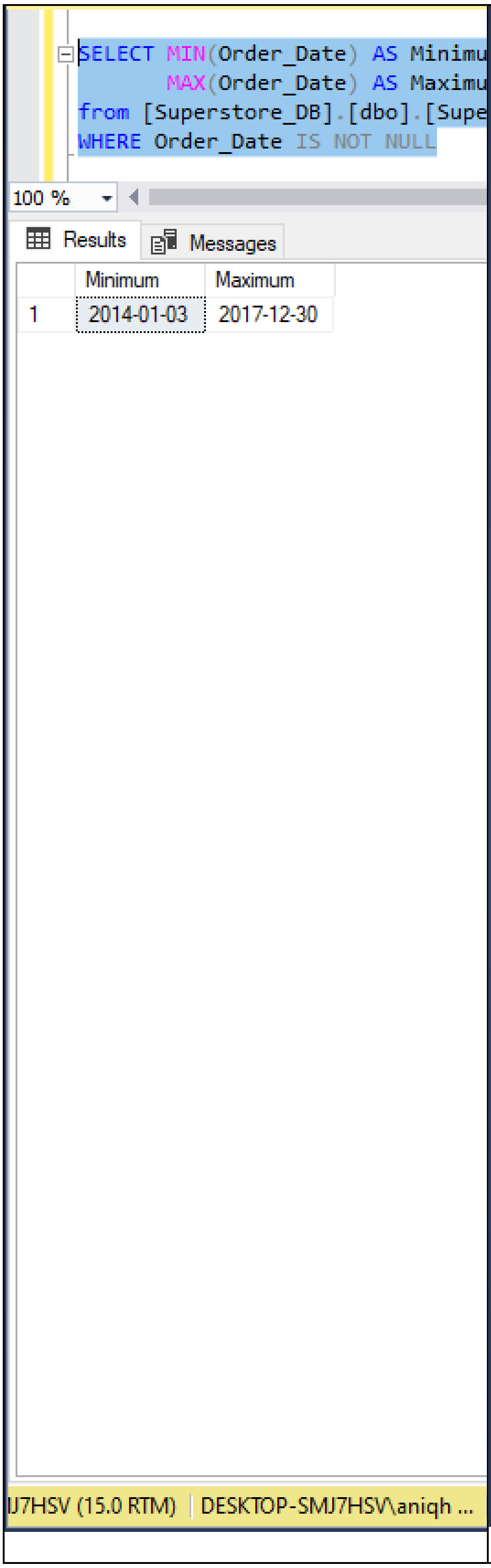


```
SELECT MIN(Discount) AS Minimum,
       MAX(Discount) AS Maximum,
       AVG(Discount) AS Mean,
       STDEV(Discount) AS StandD
       VAR(Discount) AS Variance
from [Superstore_DB].[dbo].[Supe
WHERE Discount IS NOT NULL
```

	Minimum	Maximum	Mean	StandDe
1	0.00	0.80	0.156202	0.20645

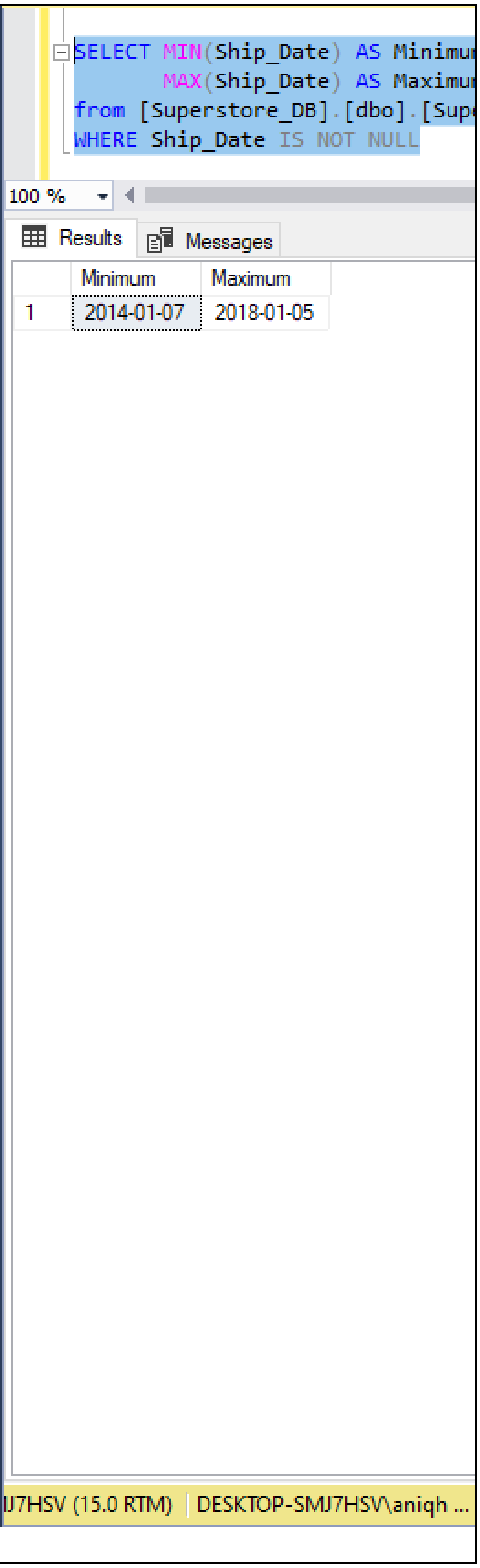
J7HSV (15.0 RTM) | DESKTOP-SMJ7HSV\aniqh ...

Order_Date	Ship_Date



```
SELECT MIN(Order_Date) AS Minimu
       MAX(Order_Date) AS Maximu
from [Superstore_DB].[dbo].[Supe
WHERE Order_Date IS NOT NULL
```

	Minimum	Maximum
1	2014-01-03	2017-12-30



```
SELECT MIN(Ship_Date) AS Minimu
       MAX(Ship_Date) AS Maximu
from [Superstore_DB].[dbo].[Sup
WHERE Ship_Date IS NOT NULL
```

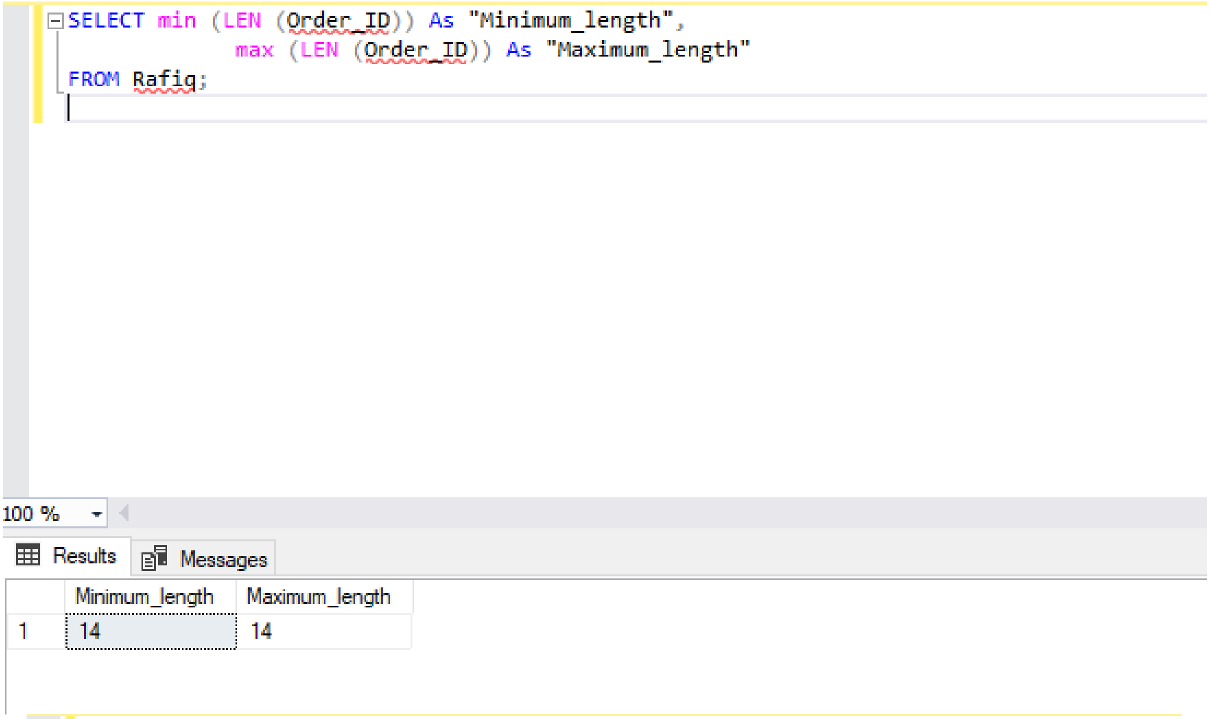
	Minimum	Maximum
1	2014-01-07	2018-01-05

1. **Length Distribution –**

Length Distribution – Find the minimum and maximum lengths of string values in the selected columns. Show, also, the minimum and maximum string values. This profile helps you identify problems in your data, such as values that are valid. For example, you profile a column of Malaysia states codes that should be two characters and discover values longer than two characters. (3 points)

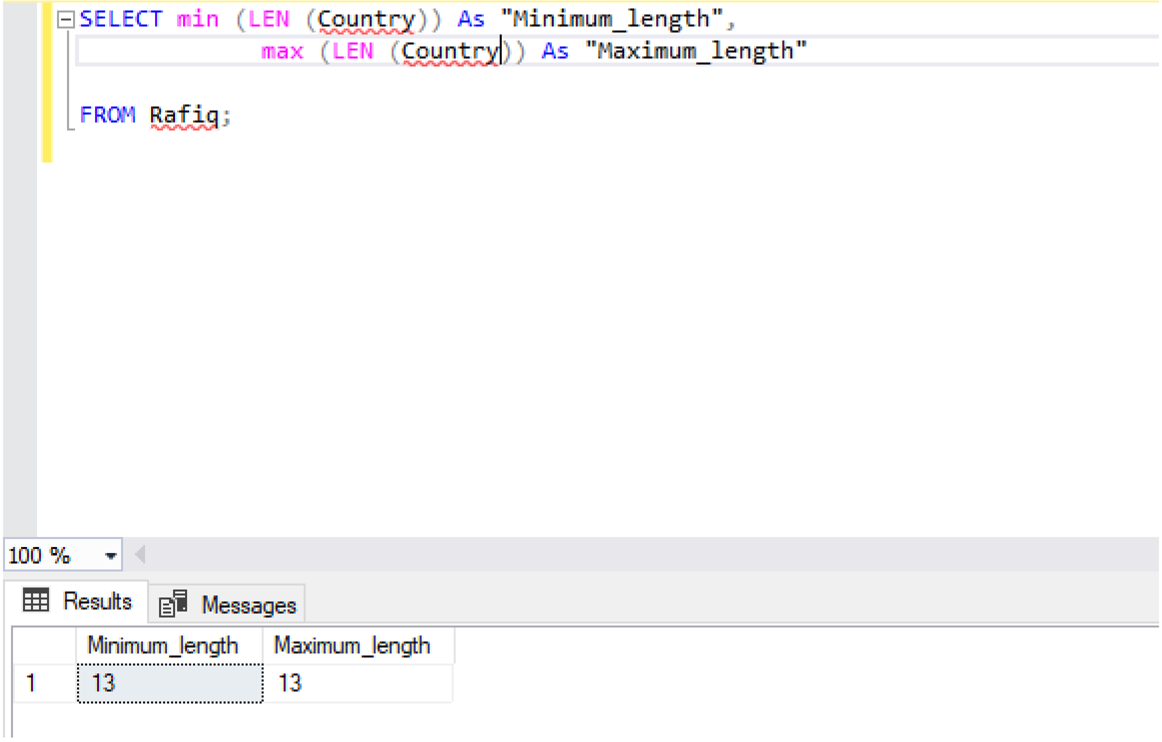
*This method is similar to value distribution, except that it only reports the lengths of string values in selected columns and the percentage of rows in the table corresponding to each value. This can assist us in filtering out invalid data.

```
SELECT  min (LEN (Product_Name)) As "Minimum_length",
            max (LEN (Product_Name)) As "Maximum_length"
        min (LEN (Category)) As "Minimum_length",
            max (LEN (Product_Name)) As "Maximum_length"
                  FROM Rafiq;
```



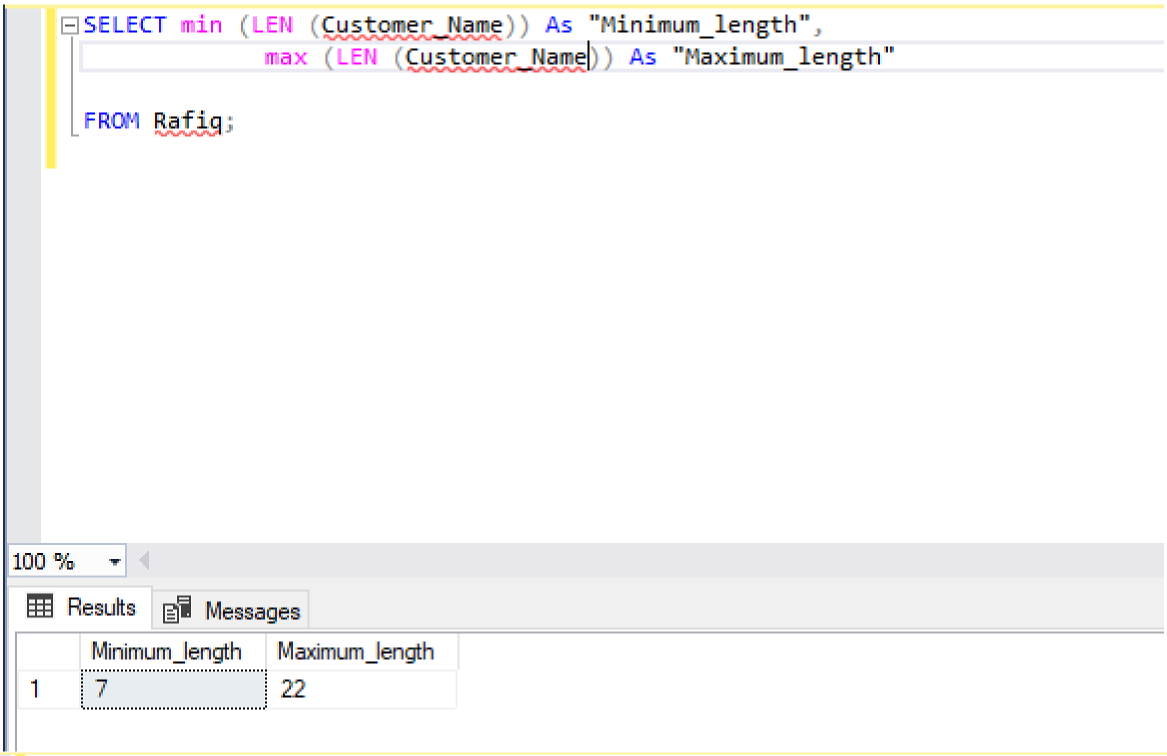
```
SELECT min (LEN (Order_ID)) As "Minimum_length",
                max (LEN (Order_ID)) As "Maximum_length"
FROM Rafiq;
```

	Minimum_length	Maximum_length
1	14	14



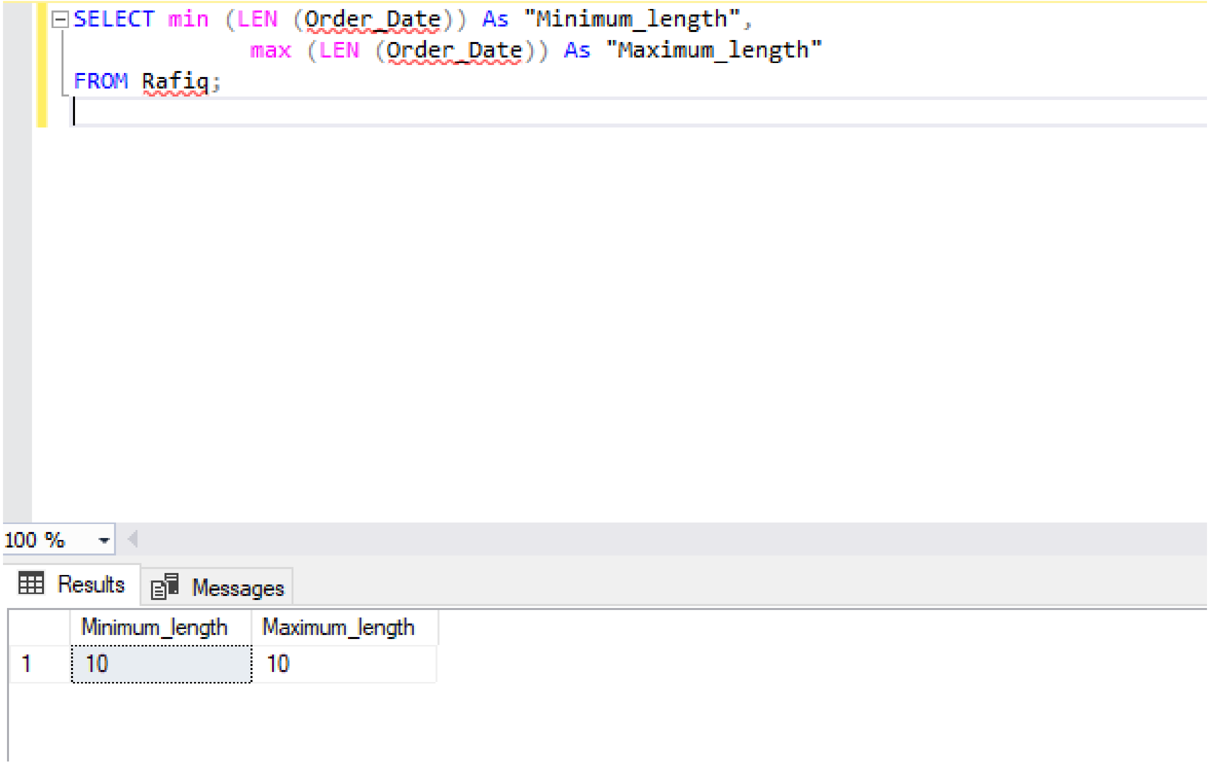
```
SELECT min (LEN (Country)) As "Minimum_length",
                max (LEN (Country)) As "Maximum_length"

FROM Rafiq;
```

	Minimum_length	Maximum_length
1	13	13



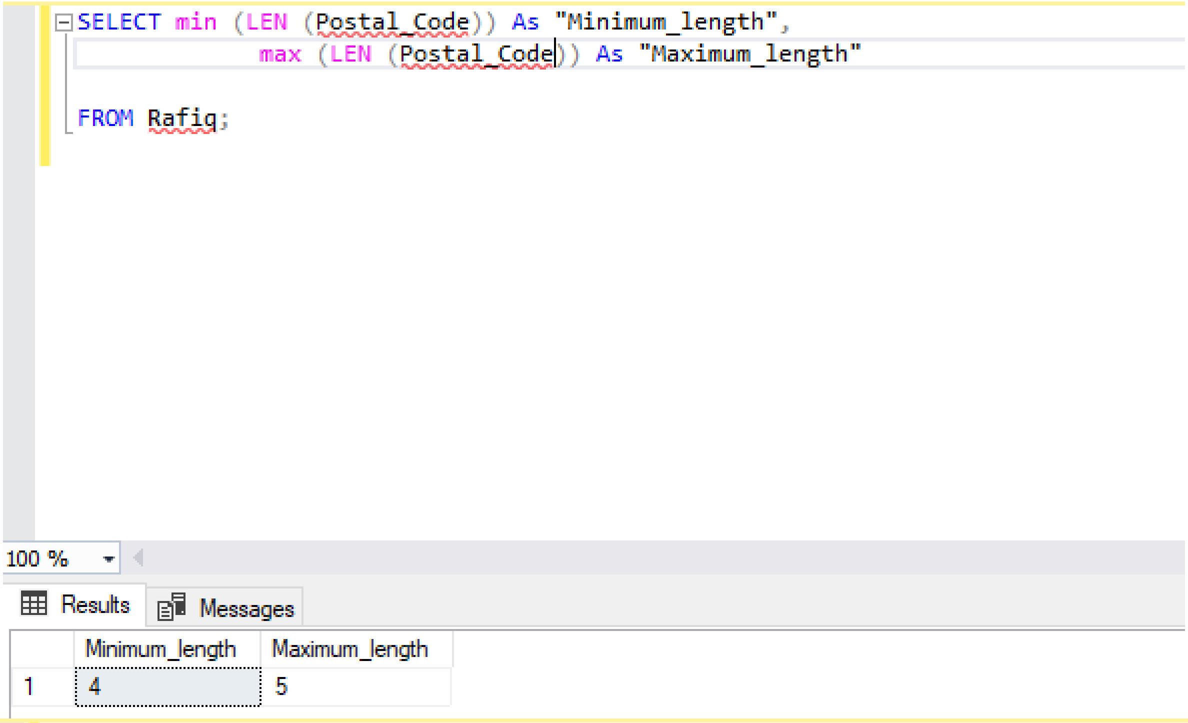
```sql
SELECT min (LEN (Customer_Name)) As "Minimum_length",
                max (LEN (Customer_Name)) As "Maximum_length"

FROM Rafiq;
```

	Minimum_length	Maximum_length
1	7	22



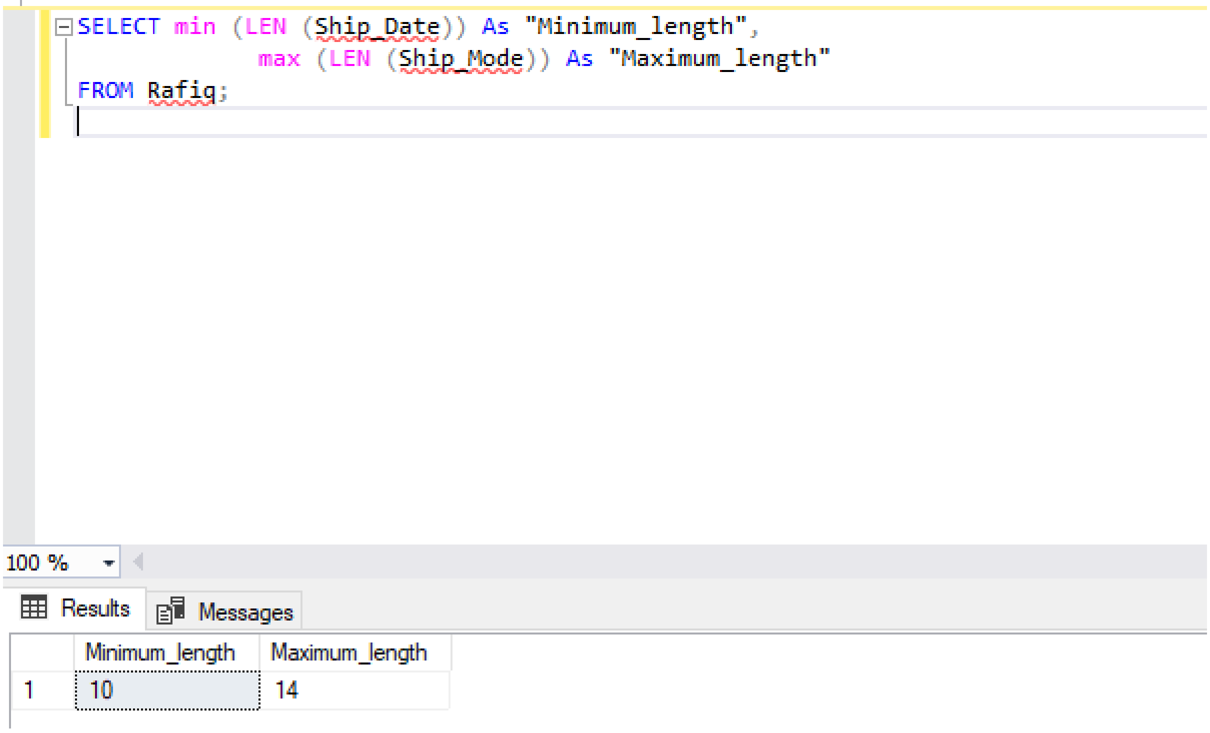
```sql
SELECT min (LEN (Order_Date)) As "Minimum_length",
             max (LEN (Order_Date)) As "Maximum_length"
FROM Rafiq;
```

	Minimum_length	Maximum_length
1	10	10



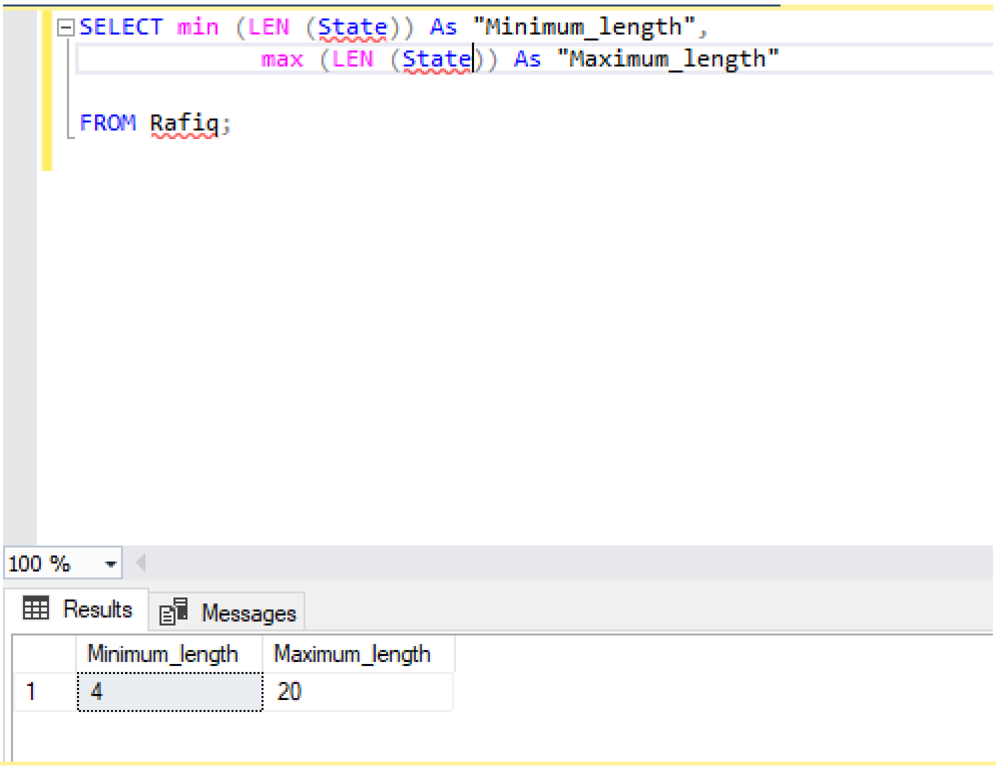
```sql
SELECT min (LEN (Postal_Code)) As "Minimum_length",
                max (LEN (Postal_Code)) As "Maximum_length"

FROM Rafiq;
```

	Minimum_length	Maximum_length
1	4	5



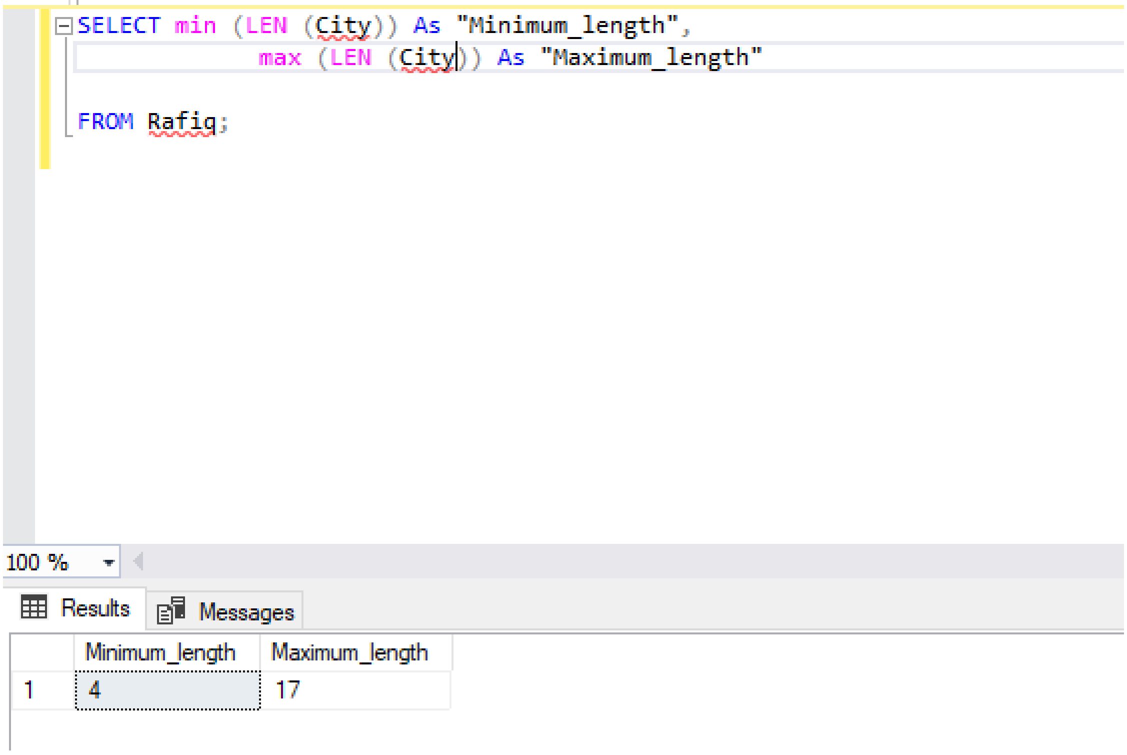
```sql
SELECT min (LEN (Ship_Date)) As "Minimum_length",
                max (LEN (Ship_Mode)) As "Maximum_length"
FROM Rafiq;
```

	Minimum_length	Maximum_length
1	10	14



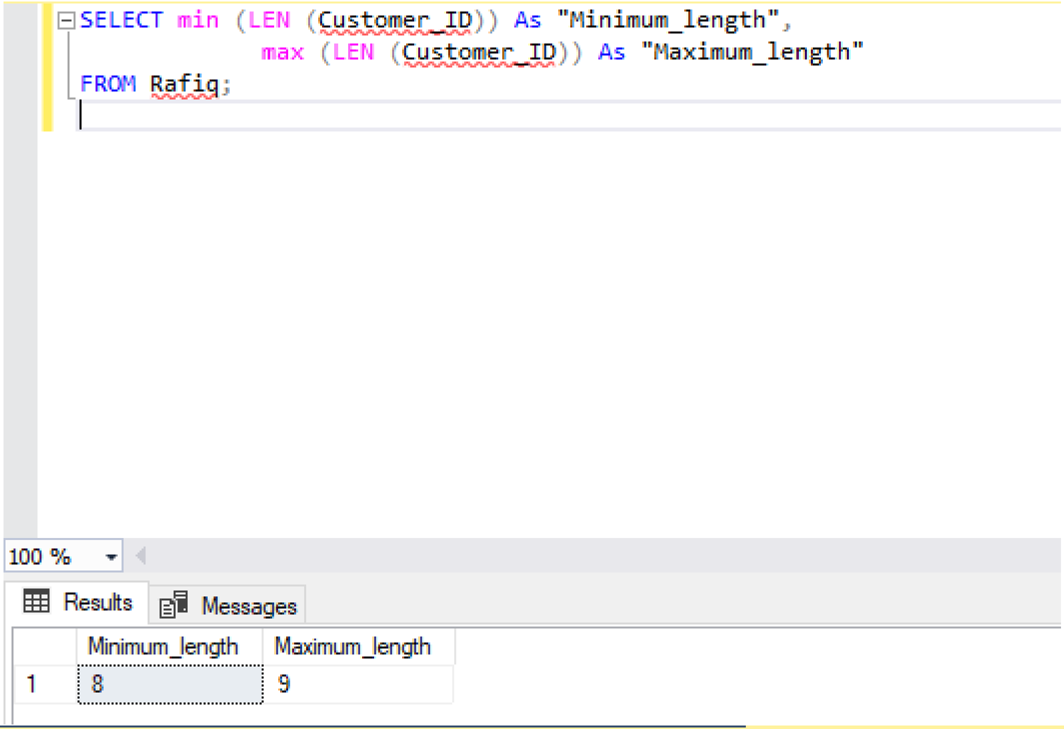
```
SELECT min (LEN (State)) As "Minimum_length",
             max (LEN (State)) As "Maximum_length"

FROM Rafiq;
```

	Minimum_length	Maximum_length
1	4	20



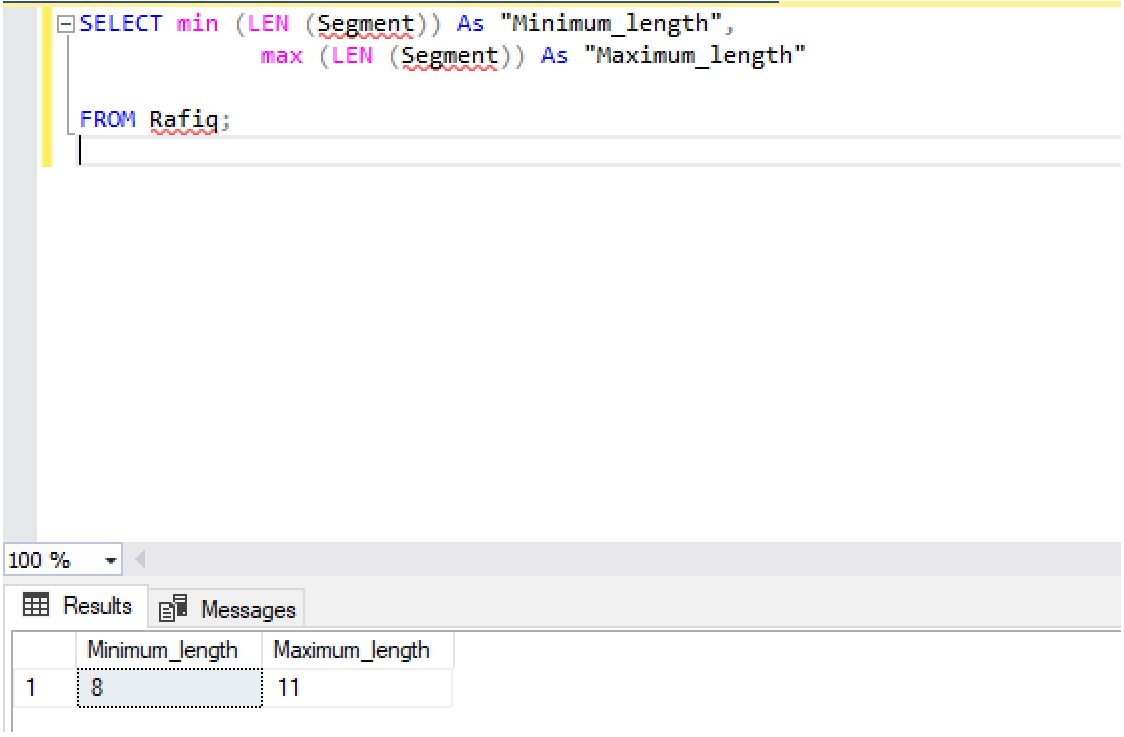
```
SELECT min (LEN (City)) As "Minimum_length",
             max (LEN (City)) As "Maximum_length"

FROM Rafiq;
```

	Minimum_length	Maximum_length
1	4	17



```sql
SELECT min (LEN (Customer_ID)) As "Minimum_length",
                    max (LEN (Customer_ID)) As "Maximum_length"
FROM Rafiq;
```

	Minimum_length	Maximum_length
1	8	9



```sql
SELECT min (LEN (Segment)) As "Minimum_length",
                    max (LEN (Segment)) As "Maximum_length"

FROM Rafiq;
```

	Minimum_length	Maximum_length
1	8	11

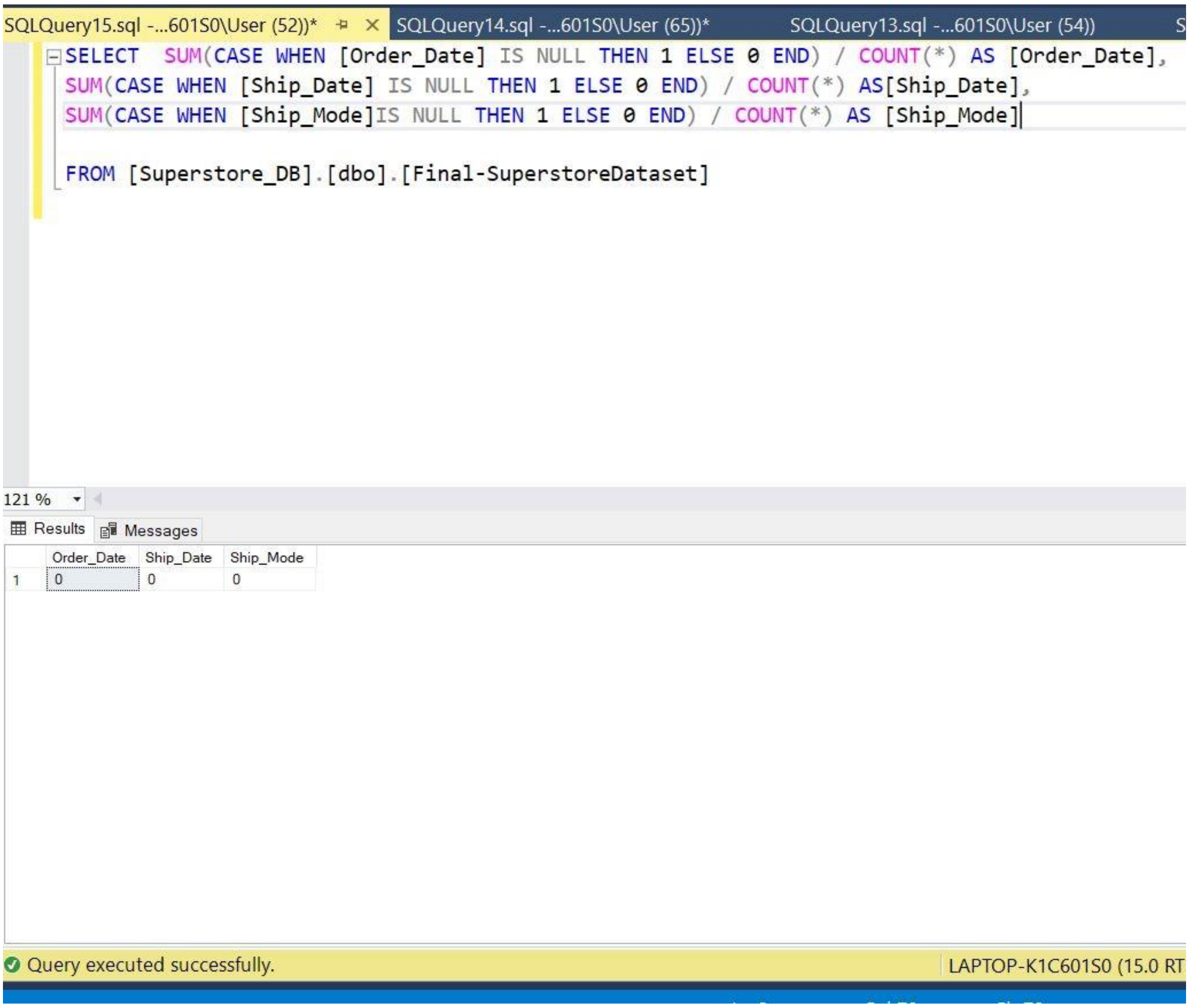
1. **Null Ratio** – Find the number of null values in the columns and reports the percentage of null values in the columns. This profile helps you identify problems in your data, such as an unexpectedly high ratio of null values in a column. For example, you profile a postcode column and discover an unacceptably high percentage of missing codes. (*3 points*)

*** This method reports the percentage of null values available in a specific column. If we have a high ratio of null values in our dataset, it will affect the decision making. We have to

make sure the null ratio in our dataset is within the acceptable limit. The less null values we have the better.
Our column :Order_Date,Ship_Mode,Ship_Date,Customer_ID, Product_ID,Category, Postal_code,Sales,Quantity,Discount,Profit

SQL Statement Template:

```
SELECT SUM(CASE WHEN Column1 IS NULL THEN 1 ELSE 0 END) / COUNT(*) AS Column1,
SUM(CASE WHEN Column2 IS NULL THEN 1 ELSE 0 END) / COUNT(*) AS Column2,
SUM(CASE WHEN Column3 IS NULL THEN 1 ELSE 0 END) / COUNT(*) AS Column3
FROM table;
```

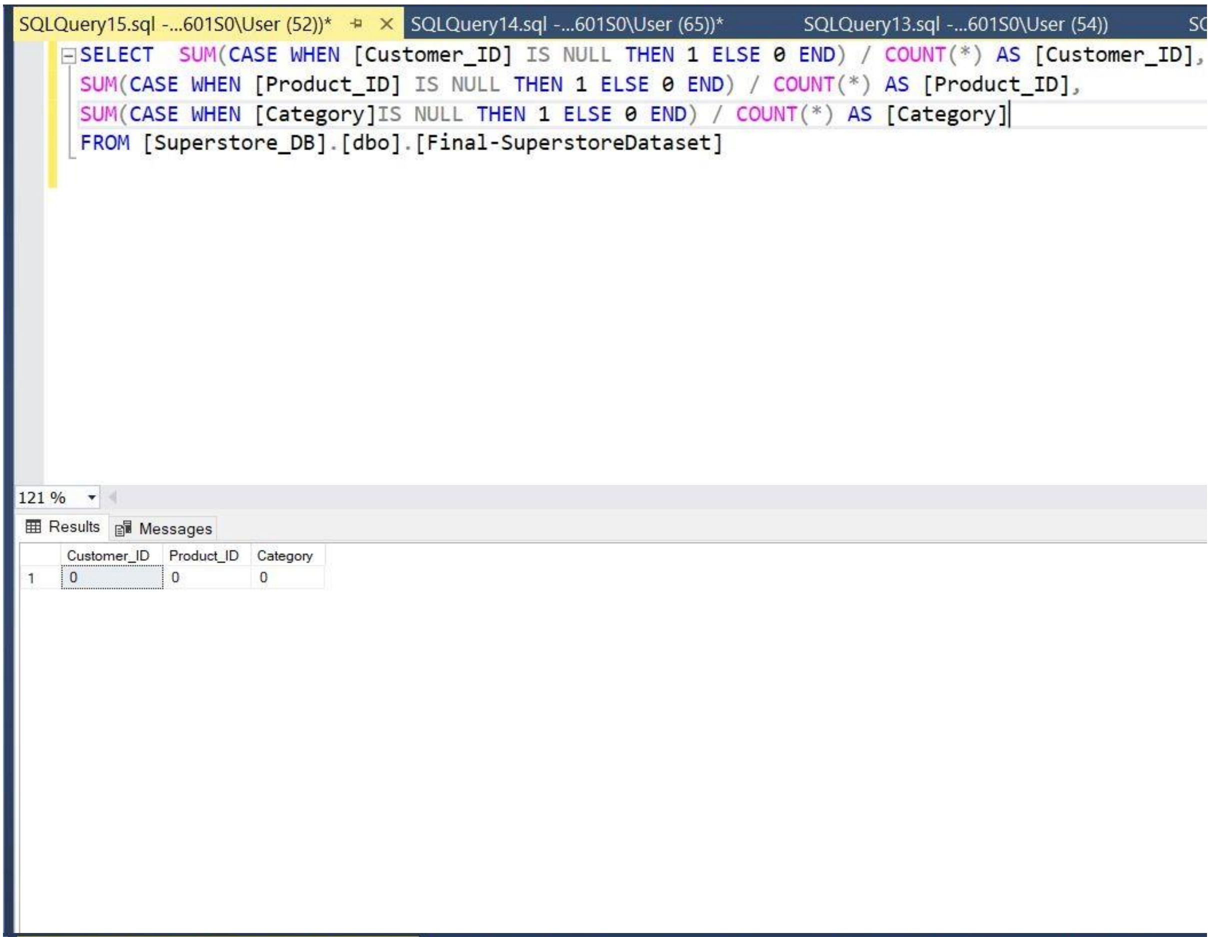


SQLQuery15.sql -...601S0\User (52))*
SQLQuery14.sql -...601S0\User (65))*
SQLQuery13.sql -...601S0\User (54))
121 %
Results
Messages
LAPTOP-K1C601S0 (15.0 RT

```
SELECT  SUM(CASE WHEN [Order_Date] IS NULL THEN 1 ELSE 0 END) / COUNT(*) AS [Order_Date],
SUM(CASE WHEN [Ship_Date] IS NULL THEN 1 ELSE 0 END) / COUNT(*) AS[Ship_Date],
SUM(CASE WHEN [Ship_Mode]IS NULL THEN 1 ELSE 0 END) / COUNT(*) AS [Ship_Mode]

FROM [Superstore_DB].[dbo].[Final-SuperstoreDataset]
```

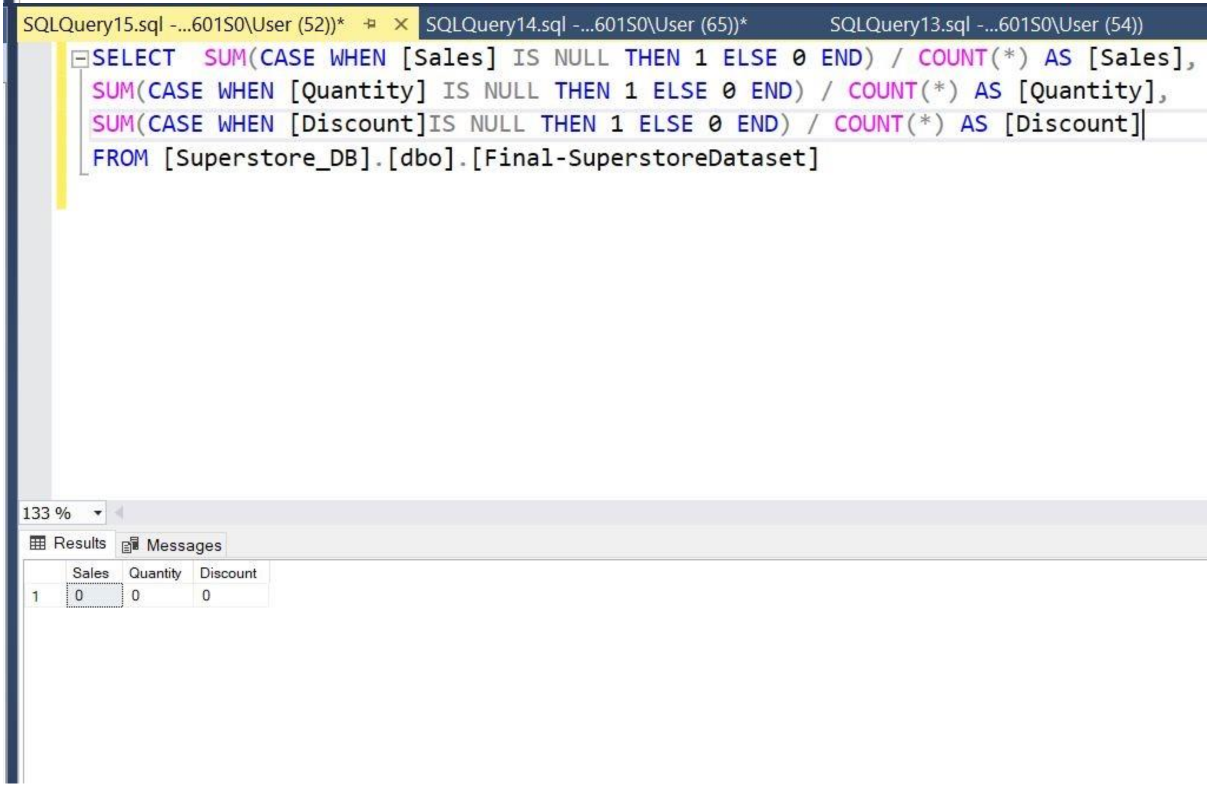
	Order_Date	Ship_Date	Ship_Mode
1	0	0	0

Query executed successfully.


SQLQuery15.sql -...601S0\User (52))*
SQLQuery14.sql -...601S0\User (65))*
SQLQuery13.sql -...601S0\User (54))
SC
121 %
Results
Messages

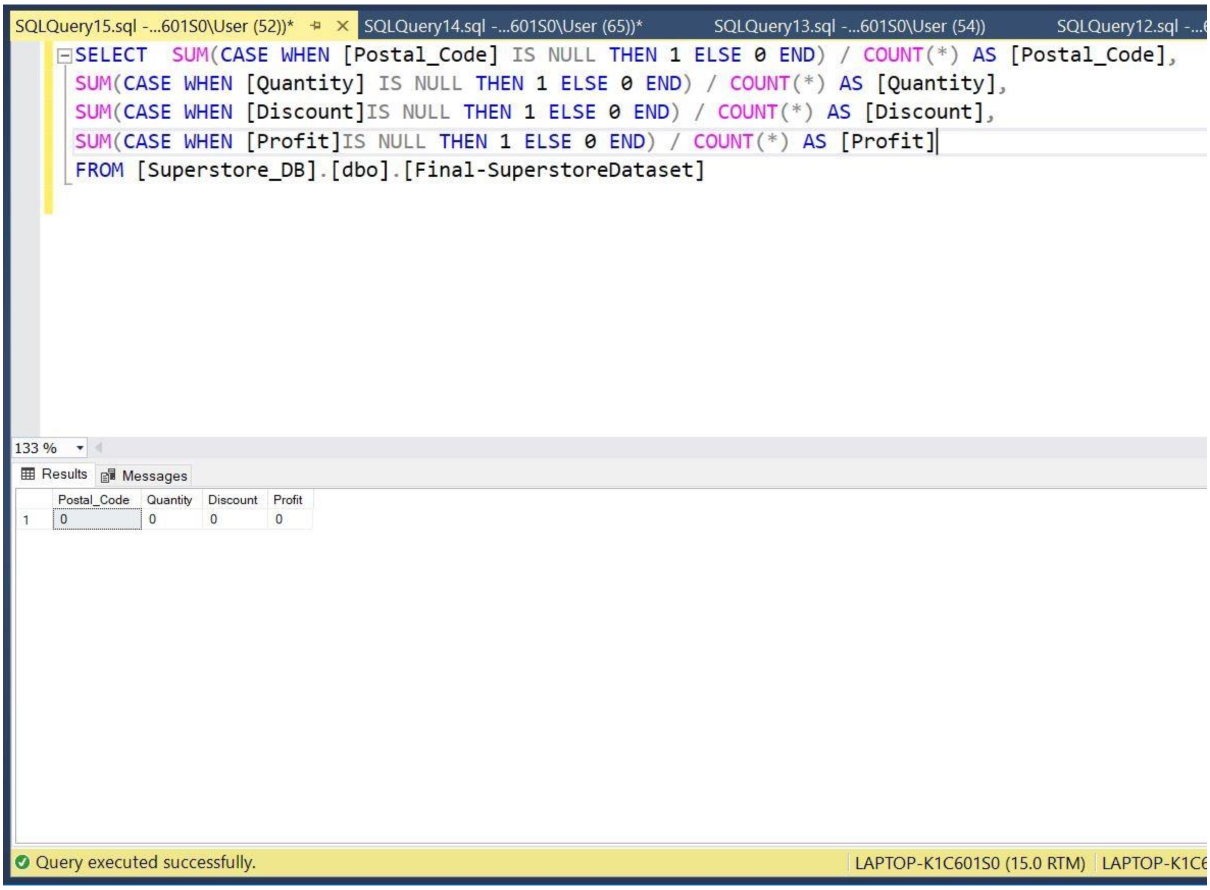
```
SELECT  SUM(CASE WHEN [Customer_ID] IS NULL THEN 1 ELSE 0 END) / COUNT(*) AS [Customer_ID],
SUM(CASE WHEN [Product_ID] IS NULL THEN 1 ELSE 0 END) / COUNT(*) AS [Product_ID],
SUM(CASE WHEN [Category]IS NULL THEN 1 ELSE 0 END) / COUNT(*) AS [Category]
FROM [Superstore_DB].[dbo].[Final-SuperstoreDataset]
```

	Customer_ID	Product_ID	Category
1	0	0	0


SQLQuery15.sql -...601S0\User (52))*
SQLQuery14.sql -...601S0\User (65))*
SQLQuery13.sql -...601S0\User (54))
133 %
Results
Messages

```
SELECT  SUM(CASE WHEN [Sales] IS NULL THEN 1 ELSE 0 END) / COUNT(*) AS [Sales],
SUM(CASE WHEN [Quantity] IS NULL THEN 1 ELSE 0 END) / COUNT(*) AS [Quantity],
SUM(CASE WHEN [Discount]IS NULL THEN 1 ELSE 0 END) / COUNT(*) AS [Discount]
FROM [Superstore_DB].[dbo].[Final-SuperstoreDataset]
```

	Sales	Quantity	Discount
1	0	0	0



SQLQuery15.sql -...601S0\User (52))*
SQLQuery14.sql -...601S0\User (65))*
SQLQuery13.sql -...601S0\User (54))
SQLQuery12.sql -...
133 %
Results
Messages

```
SELECT  SUM(CASE WHEN [Postal_Code] IS NULL THEN 1 ELSE 0 END) / COUNT(*) AS [Postal_Code],
SUM(CASE WHEN [Quantity] IS NULL THEN 1 ELSE 0 END) / COUNT(*) AS [Quantity],
SUM(CASE WHEN [Discount]IS NULL THEN 1 ELSE 0 END) / COUNT(*) AS [Discount],
SUM(CASE WHEN [Profit]IS NULL THEN 1 ELSE 0 END) / COUNT(*) AS [Profit]
FROM [Superstore_DB].[dbo].[Final-SuperstoreDataset]
```

	Postal_Code	Quantity	Discount	Profit
1	0	0	0	0

Query executed successfully.

1. Use your own creativity to produce ONE (1) SQL statement and the corresponding English statement to describe the query. You must use any element indicated below. You may also

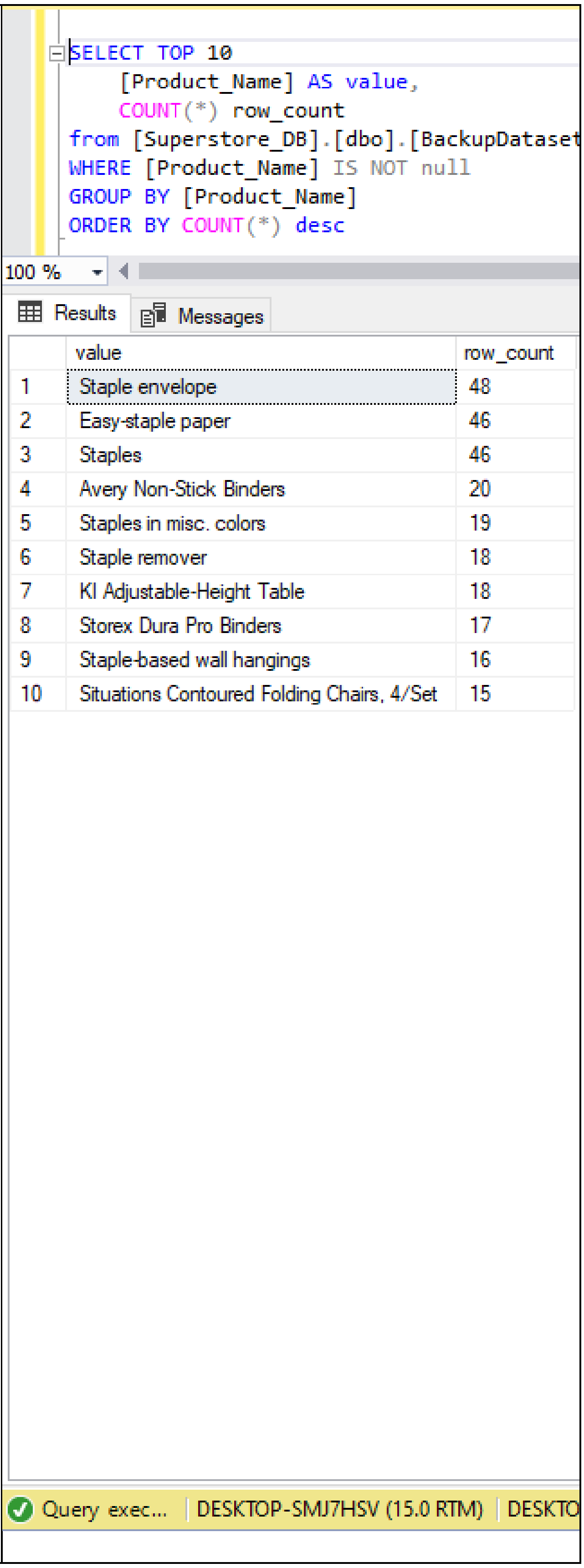
include other additional functions.

(*3 points*)

a. **TOP()** function. **TOP()** function returns the top specified number of rows. Use the function in conjunction with the **ORDER BY** clause.

- Finding the top 10 most bought items.

```
SELECT TOP 10
       [Product_Name
       ] AS value,
       COUNT(*)
row_count
From
[Superstore_DB].[db
o].[BackupDataset]
WHERE
[Product_Name] IS
NOT null
GROUP BY
[Product_Name]
ORDER   BY   COUNT(*)
desc
```



```
SELECT TOP 10
    [Product_Name] AS value,
    COUNT(*) row_count
from [Superstore_DB].[dbo].[BackupDataset
WHERE [Product_Name] IS NOT null
GROUP BY [Product_Name]
ORDER BY COUNT(*) desc
```

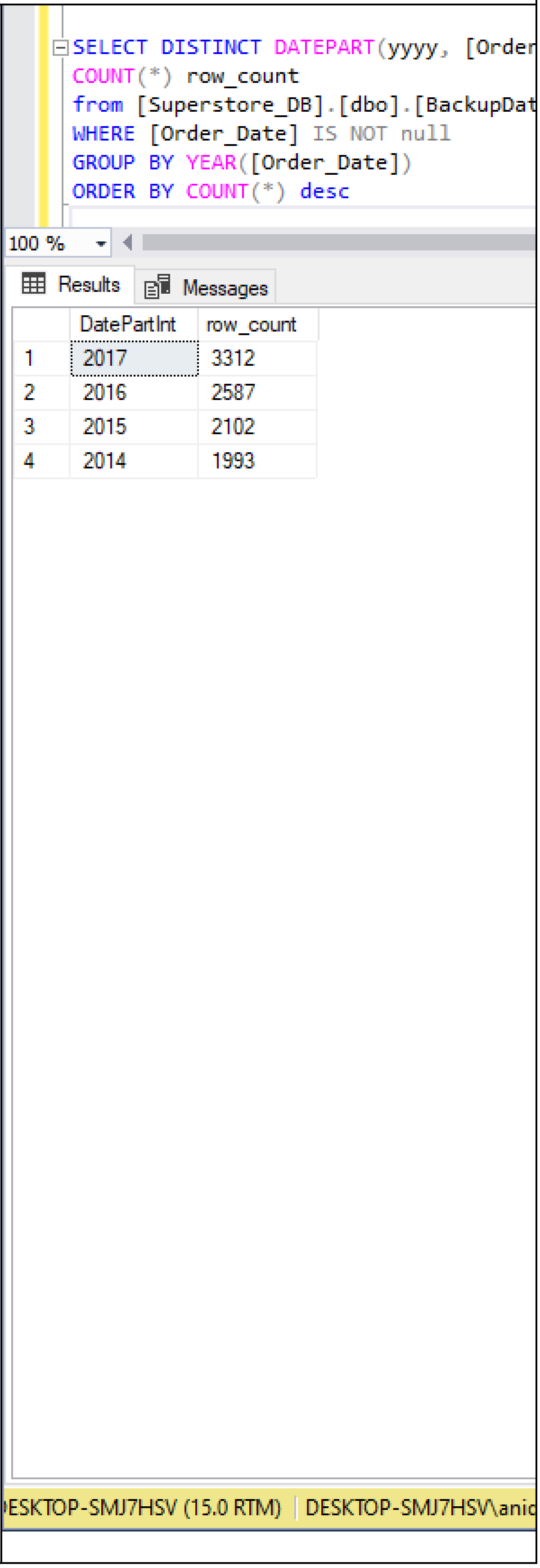
	value	row_count
1	Staple envelope	48
2	Easy-staple paper	46
3	Staples	46
4	Avery Non-Stick Binders	20
5	Staples in misc. colors	19
6	Staple remover	18
7	KI Adjustable-Height Table	18
8	Storex Dura Pro Binders	17
9	Staple-based wall hangings	16
10	Situations Contoured Folding Chairs, 4/Set	15

Query exec... | DESKTOP-SMJ7HSV (15.0 RTM) | DESKTO

a. **DATEPART()** function. **DATEPART()** function returns individual parts of a date.

- Finding how many orders were placed in each distinctive years.

```
SELECT DISTINCT
DATEPART(yyyy,
[Order_Date]) AS
DatePartInt,
COUNT(*) row_count
from
[Superstore_DB].[dbo].
[BackupDataset]
WHERE [Order_Date] IS
NOT null
GROUP BY
YEAR([Order_Date])
ORDER BY COUNT(*) desc
```



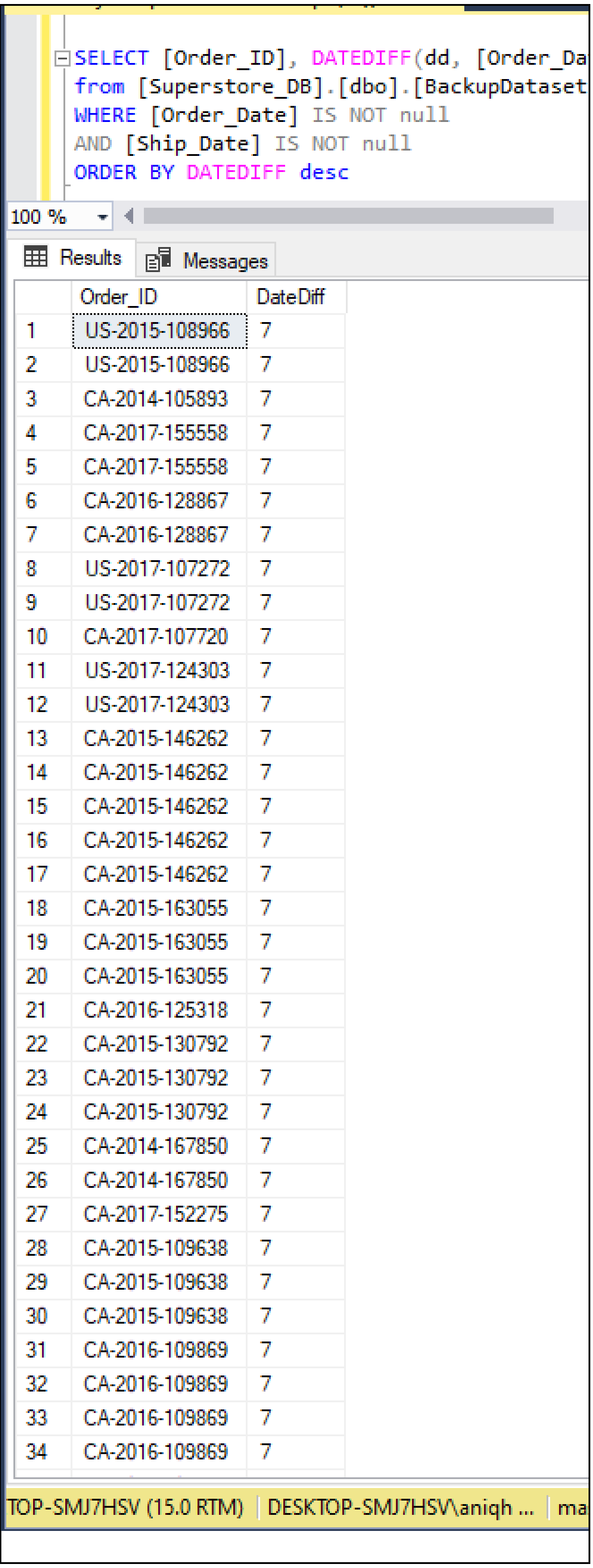
```
SELECT DISTINCT DATEPART(yyyy, [Order
COUNT(*) row_count
from [Superstore_DB].[dbo].[BackupDat
WHERE [Order_Date] IS NOT null
GROUP BY YEAR([Order_Date])
ORDER BY COUNT(*) desc
```

	DatePartInt	row_count
1	2017	3312
2	2016	2587
3	2015	2102
4	2014	1993

a. **DATEDIFF()**. **DATEDIFF()** function returns the date difference in days, months, year etc.

- Finding the difference between shipping date and the time the order was placed (order date).

```
SELECT [Order_ID],
DATEDIFF(dd,
[Order_Date],
[Ship_Date]) AS DateDiff
from
[Superstore_DB].[dbo].[BackupDataset]
WHERE [Order_Date] IS NOT
null
AND [Ship_Date] IS NOT
null
ORDER BY DATEDIFF desc
```



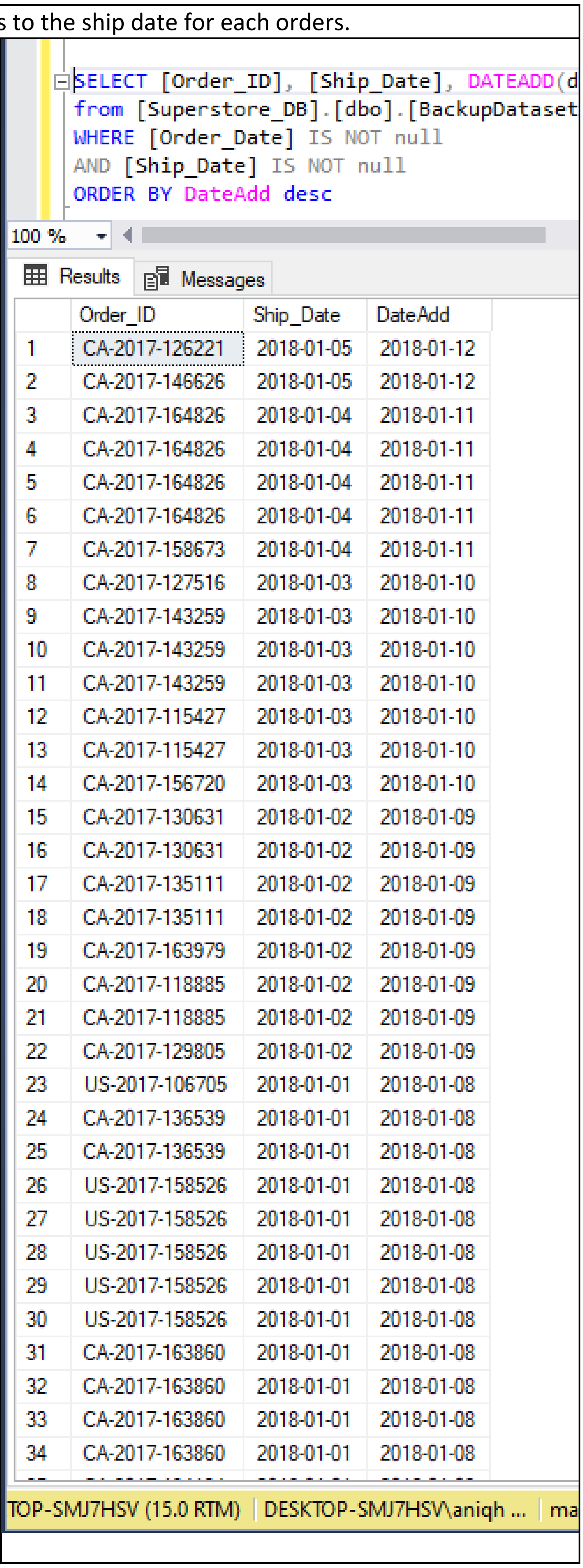
```
SELECT [Order_ID], DATEDIFF(dd, [Order_Da
from [Superstore_DB].[dbo].[BackupDataset
WHERE [Order_Date] IS NOT null
AND [Ship_Date] IS NOT null
ORDER BY DATEDIFF desc
```

	Order_ID	DateDiff
1	US-2015-108966	7
2	US-2015-108966	7
3	CA-2014-105893	7
4	CA-2017-155558	7
5	CA-2017-155558	7
6	CA-2016-128867	7
7	CA-2016-128867	7
8	US-2017-107272	7
9	US-2017-107272	7
10	CA-2017-107720	7
11	US-2017-124303	7
12	US-2017-124303	7
13	CA-2015-146262	7
14	CA-2015-146262	7
15	CA-2015-146262	7
16	CA-2015-146262	7
17	CA-2015-146262	7
18	CA-2015-163055	7
19	CA-2015-163055	7
20	CA-2015-163055	7
21	CA-2016-125318	7
22	CA-2015-130792	7
23	CA-2015-130792	7
24	CA-2015-130792	7
25	CA-2014-167850	7
26	CA-2014-167850	7
27	CA-2017-152275	7
28	CA-2015-109638	7
29	CA-2015-109638	7
30	CA-2015-109638	7
31	CA-2016-109869	7
32	CA-2016-109869	7
33	CA-2016-109869	7
34	CA-2016-109869	7

a. **DATEADD()**. **DATEADD()** adds or subtracts a specified time interval from a date.

- Adding 7 days to the ship date for each orders.

```
SELECT [Order_ID],
[Ship_Date], DATEADD (dd,
7, [Ship_Date]) AS
DateAdd
from
[Superstore_DB].[dbo].[BackupDataset]
WHERE [Order_Date] IS NOT
null
AND [Ship_Date] IS NOT
null
ORDER BY DateAdd desc
```



```
SELECT [Order_ID], [Ship_Date], DATEADD(d
from [Superstore_DB].[dbo].[BackupDataset
WHERE [Order_Date] IS NOT null
AND [Ship_Date] IS NOT null
ORDER BY DateAdd desc
```

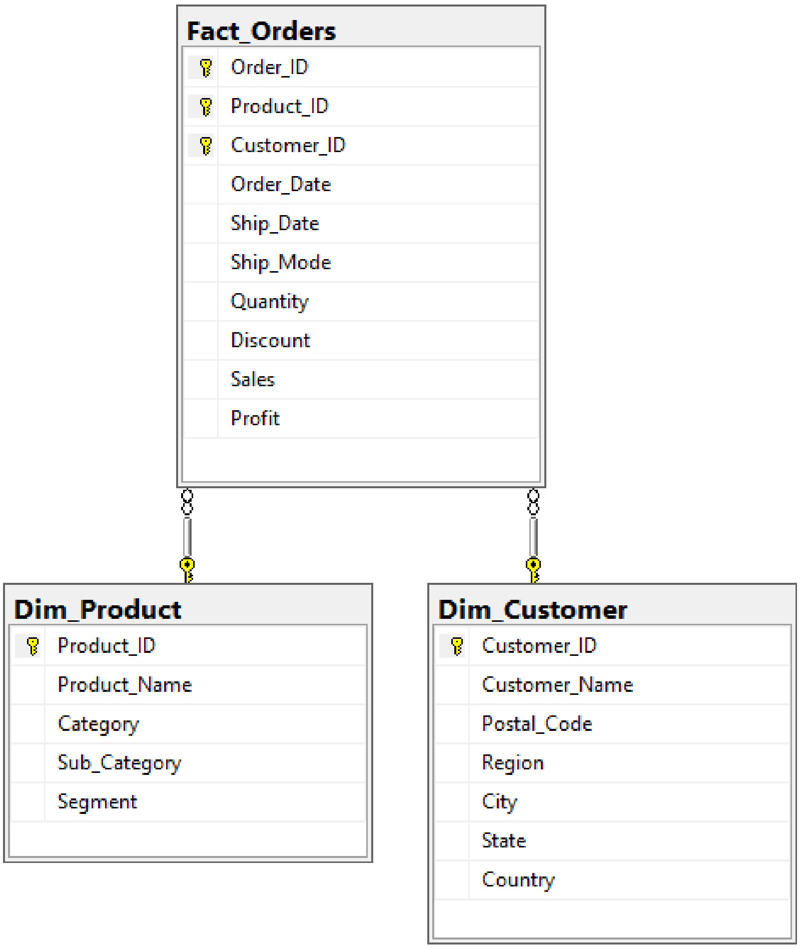
	Order_ID	Ship_Date	DateAdd
1	CA-2017-126221	2018-01-05	2018-01-12
2	CA-2017-146626	2018-01-05	2018-01-12
3	CA-2017-164826	2018-01-04	2018-01-11
4	CA-2017-164826	2018-01-04	2018-01-11
5	CA-2017-164826	2018-01-04	2018-01-11
6	CA-2017-164826	2018-01-04	2018-01-11
7	CA-2017-158673	2018-01-04	2018-01-11
8	CA-2017-127516	2018-01-03	2018-01-10
9	CA-2017-143259	2018-01-03	2018-01-10
10	CA-2017-143259	2018-01-03	2018-01-10
11	CA-2017-143259	2018-01-03	2018-01-10
12	CA-2017-115427	2018-01-03	2018-01-10
13	CA-2017-115427	2018-01-03	2018-01-10
14	CA-2017-156720	2018-01-03	2018-01-10
15	CA-2017-130631	2018-01-02	2018-01-09
16	CA-2017-130631	2018-01-02	2018-01-09
17	CA-2017-135111	2018-01-02	2018-01-09
18	CA-2017-135111	2018-01-02	2018-01-09
19	CA-2017-163979	2018-01-02	2018-01-09
20	CA-2017-118885	2018-01-02	2018-01-09
21	CA-2017-118885	2018-01-02	2018-01-09
22	CA-2017-129805	2018-01-02	2018-01-09
23	US-2017-106705	2018-01-01	2018-01-08
24	CA-2017-136539	2018-01-01	2018-01-08
25	CA-2017-136539	2018-01-01	2018-01-08
26	US-2017-158526	2018-01-01	2018-01-08
27	US-2017-158526	2018-01-01	2018-01-08
28	US-2017-158526	2018-01-01	2018-01-08
29	US-2017-158526	2018-01-01	2018-01-08
30	US-2017-158526	2018-01-01	2018-01-08
31	CA-2017-163860	2018-01-01	2018-01-08
32	CA-2017-163860	2018-01-01	2018-01-08
33	CA-2017-163860	2018-01-01	2018-01-08
34	CA-2017-163860	2018-01-01	2018-01-08

1. DATA WAREHOUSE BUS MATRIX

Use the BUS MATRIX (Must be accompanied with a brief explanation of your proposal),

1. To propose a business process that can be formed from the data set sourced for the data warehouse to be created and to explain the business process. (*3 points*)

Business Process	Fact Table	Granularity	Facts	Dim_Customer	Dim_Product
Order Creation	Fact_Orders	One row per new order entry	Order ID, Order_Date, Customer_ID	X	X
Superstore Sales	Fact_Orders	One row per Order Sale	Order ID, Sales	X	X
Order Fulfillment	Fact_Orders	One row per Order to be shipped out	Order ID, Ship_Date	X	X



1. To identify and explain the proposed fact and dimensions tables, their granularities, and facts. (*4 points*)

The measurement of the business process is contained in the fact table, as is FK for the dimensional tables. On the other hand, dimensional tables contain measurement attributes stored in fact tables.
As you can see, the following are the facts that define our company. (Order_ID, Product_ID, Customer_ID, Order_Date, Ship_Date, Ship_Mode, Quantity, Discount, Sales, and Profit).
Alternatively, all of the FK are contain dimensional tables. It also contains properties of measurement in dimensional tables. The primary keys to our fact table are Dim Product and Dim Customers. So, the fact table will contain to our all-primary key. Product_ID, Product_Name, Category, Sub Category, Segment, Prostal Code, Region, City, State, and Country all are contain to our FK.

Importing data:
Dimensional tables must load first because they contain primary keys that are unique numbers and all of the details, whereas fact tables contain foreign keys. As a result, we cannot load the fact table without a foreign key.