

Transact SQL

Transact-SQL (T-SQL) is Microsoft's extension of the SQL (Structured Query Language) standard. It is a procedural programming language that enhances SQL with additional programming constructs, control-of-flow operators, local variables, and error handling capabilities.

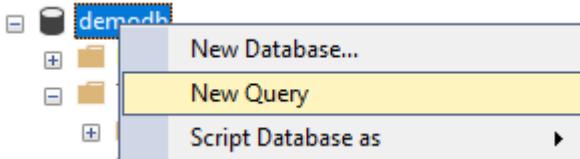
Here's an overview of Transact-SQL:

1. **Data Manipulation:** T-SQL allows for querying, inserting, updating, and deleting data in relational databases. It supports standard SQL operations like SELECT, INSERT, UPDATE, and DELETE.
2. **Data Definition:** T-SQL enables the definition of database objects such as tables, views, stored procedures, functions, indexes, and triggers using commands like CREATE, ALTER, and DROP.
3. **Procedural Programming:** T-SQL includes procedural programming features such as variables, control-of-flow statements (IF...ELSE, WHILE, BEGIN...END), loops, and exception handling with TRY...CATCH blocks.
4. **Transaction Control:** T-SQL supports transactions, allowing developers to ensure data integrity by grouping SQL operations into atomic units of work. Commands like BEGIN TRANSACTION, COMMIT, and ROLLBACK are used to manage transactions.
5. **Error Handling:** T-SQL provides mechanisms for handling errors and exceptions that may occur during the execution of SQL statements or stored procedures. This includes TRY...CATCH blocks for structured error handling.
6. **Dynamic SQL:** T-SQL allows for the dynamic generation and execution of SQL statements within stored procedures or application code, enabling flexibility in query construction based on runtime conditions.
7. **System Functions and Stored Procedures:** T-SQL includes a wide range of built-in functions and system-stored procedures for tasks such as string manipulation, date and time operations, mathematical calculations, and system administration.
8. **Security:** T-SQL supports security features such as permissions, roles, and encryption, allowing developers to control access to database objects and sensitive data.

T-SQL is primarily used with Microsoft SQL Server and Azure SQL Database but can also be found in other Microsoft database-related products and services. It provides a powerful and flexible platform for developing database applications, stored procedures, and scripts for managing and manipulating data in SQL Server environments.

SELECT Clause

1. First select your database then right-click on it, after that, you have to choose New Query.



2. Now the first statement we are going to issue is the Select statement.
3. If you want to see the data within a table in your database, you can go ahead and issue the SELECT statement. If you want to return all the columns of information from within a table, we can do a star FROM.
4. Then choose the schema name that is SalesLT, the default schema that's part of this sample database.
5. After that we give the table name which happens to be Product. Once all it is written then click on Execute.

The screenshot shows the SSMS interface. The Object Explorer pane on the left shows a database named 'demodb'. The main pane contains a query window titled 'SQLQuery1.sql - de...odb (sqladmin (70))'. The query is:

```
SELECT * FROM SalesLT.Product
```

6. Here below you can see the data after execution of the command.

The screenshot shows the SSMS Results pane displaying the output of the query. The table has 16 rows and the following columns:

	ProductID	Name	ProductNumber	Color	StandardCost	ListPrice	Size	Weight	ProductCategoryID	ProductModelID	SellStartDate	SellEndDate
1	680	HL Road Frame - Black, 58	FR-R92B-58	Black	1059.31	1431.50	58	1016.04	18	6	2002-06-01 00:00:00.000	NULL
2	706	HL Road Frame - Red, 58	FR-R92R-58	Red	1059.31	1431.50	58	1016.04	18	6	2002-06-01 00:00:00.000	NULL
3	707	Sport-100 Helmet, Red	HL-U509-R	Red	13.0863	34.99	NULL	NULL	35	33	2005-07-01 00:00:00.000	NULL
4	708	Sport-100 Helmet, Black	HL-U509	Black	13.0863	34.99	NULL	NULL	35	33	2005-07-01 00:00:00.000	NULL
5	709	Mountain Bike Socks, M	SO-B909-M	White	3.3963	9.50	M	NULL	27	18	2005-07-01 00:00:00.000	2006-06-30 00:00:00.000
6	710	Mountain Bike Socks, L	SO-B909-L	White	3.3963	9.50	L	NULL	27	18	2005-07-01 00:00:00.000	2006-06-30 00:00:00.000
7	711	Sport-100 Helmet, Blue	HL-U509-B	Blue	13.0863	34.99	NULL	NULL	35	33	2005-07-01 00:00:00.000	NULL
8	712	AWC Logo Cap	CA-1098	Multi	6.9223	8.99	NULL	NULL	23	2	2005-07-01 00:00:00.000	NULL
9	713	Long-Sleeve Logo Jersey, S	LJ-0192-S	Multi	38.4923	49.99	S	NULL	25	11	2005-07-01 00:00:00.000	NULL
10	714	Long-Sleeve Logo Jersey, M	LJ-0192-M	Multi	38.4923	49.99	M	NULL	25	11	2005-07-01 00:00:00.000	NULL
11	715	Long-Sleeve Logo Jersey, L	LJ-0192-L	Multi	38.4923	49.99	L	NULL	25	11	2005-07-01 00:00:00.000	NULL
12	716	Long-Sleeve Logo Jersey, XL	LJ-0192-X	Multi	38.4923	49.99	XL	NULL	25	11	2005-07-01 00:00:00.000	NULL
13	717	HL Road Frame - Red, 62	FR-R92R-62	Red	868.6342	1431.50	62	1043.26	18	6	2005-07-01 00:00:00.000	NULL
14	718	HL Road Frame - Red, 44	FR-R92R-44	Red	868.6342	1431.50	44	961.61	18	6	2005-07-01 00:00:00.000	NULL
15	719	HL Road Frame - Red, 48	FR-R92R-48	Red	868.6342	1431.50	48	979.75	18	6	2005-07-01 00:00:00.000	NULL
16	720	HL Road Frame - Red, 52	FR-R92R-52	Red	868.6342	1431.50	52	997.90	18	6	2005-07-01 00:00:00.000	NULL

Query executed successfully. demoazure1010.database.windows.net | sqladmin (70) | demodb | 00:00:01 | 295 rows

7. Now if you want to see the number of rows with in the table then you can issue this command.

SQLQuery1.sql - de...odb (sqladmin (70))*

```
SELECT COUNT(*) FROM SalesLT.Product
```

200 %

Results Messages

	(No column name)
1	295

8. If you want to see the details for certain columns, then you can issue this command.

SQLQuery1.sql - de...odb (sqladmin (70))*

```
SELECT ProductID, Name, ProductNumber FROM SalesLT.Product
```

200 %

Results Messages

ProductID	Name	ProductNumber
1	HL Road Frame - Black, 58	FR-R92B-58
2	HL Road Frame - Red, 58	FR-R92R-58
3	Sport-100 Helmet, Red	HL-U509-R
4	Sport-100 Helmet, Black	HL-U509
5	Mountain Bike Socks, M	SO-B909-M
6	Mountain Bike Socks, L	SO-B909-L
7	Sport-100 Helmet, Blue	HL-U509-B
8	AWC Logo Cap	CA-1098
9	Long-Sleeve Logo Jersey, S	LJ-0192-S
10	Long-Sleeve Logo Jersey, M	LJ-0192-M
11	Long-Sleeve Logo Jersey, L	LJ-0192-L
12	Long-Sleeve Logo Jersey, XL	LJ-0192-X
13	HL Road Frame - Red, 62	FR-R92R-62
14	HL Road Frame - Red, 44	FR-R92R-44
15	HL Road Frame - Red, 48	FR-R92R-48
16	HL Road Frame - Red, 52	FR-R92R-52
17	HL Road Frame - Red, 56	FR-R92R-56

Query executed successfully. demoazure1010.database.wind... sqladmin (70) demodb 00:00:00 | 295 rows

⌚ WHERE Clause

1. Now we are going to learn about Where condition. The WHERE clause can be used to extract only those records from within a table that fulfill a specified condition.
2. Below you can see that we add where clause to the statement to find a specific product ID and we got the result accordingly.

The screenshot shows a SQL Server Management Studio window. The query pane contains the following SQL code:

```
SELECT ProductID, Name, ProductNumber FROM SalesLT.Product WHERE ProductID=680
```

The results pane shows a single row of data:

	ProductID	Name	ProductNumber
1	680	HL Road Frame - Black, 58	FR-R92B-58

3. Now if you want to get a list of products where Product ID is greater than that specific number.

The screenshot shows a SQL Server Management Studio window. The query pane contains the following SQL code:

```
SELECT ProductID, Name, ProductNumber FROM SalesLT.Product WHERE ProductID>680
```

The results pane shows a list of products with Product IDs greater than 680:

	ProductID	Name	ProductNumber
1	879	All-Purpose Bike Stand	ST-1401
2	712	AWC Logo Cap	CA-1098
3	877	Bike Wash - Dissolver	CL-9009
4	843	Cable Lock	LO-C100
5	952	Chain	CH-0234
6	866	Classic Vest, L	VE-C304-L
7	865	Classic Vest, M	VE-C304-M
8	864	Classic Vest, S	VE-C304-S
9	878	Fender Set - Mountain	FE-6654
10	948	Front Brakes	FB-9873
11	945	Front Derailleur	FD-2342
12	863	Full-Finger Gloves, L	GL-F110-L
13	862	Full-Finger Gloves, M	GL-F110-M
14	861	Full-Finger Gloves, S	GL-F110-S
15	860	Half-Finger Gloves, L	GL-H102-L
16	859	Half-Finger Gloves, M	GL-H102-M
17	858	Half-Finger Gloves, S	GL-H102-S

At the bottom of the results pane, there is a message: "Query executed successfully." and a status bar showing "demoazure1010.database.windows.net | sqladmin (70) | demodb | 00:00:00 | 294 rows".

4. Now you can also find the data between the set of values.

The screenshot shows a SQL Server Management Studio window. The query in the top pane is:

```
SELECT ProductID, Name, ProductNumber FROM SalesLT.Product WHERE ProductID BETWEEN 750 and 900
```

The results pane displays a table with three columns: ProductID, Name, and ProductNumber. The data is as follows:

ProductID	Name	ProductNumber
1	Road-150 Red, 44	BK-R93R-44
2	Road-150 Red, 48	BK-R93R-48
3	Road-150 Red, 52	BK-R93R-52
4	Road-150 Red, 56	BK-R93R-56
5	Road-450 Red, 58	BK-R68R-58
6	Road-450 Red, 60	BK-R68R-60
7	Road-450 Red, 44	BK-R68R-44
8	Road-450 Red, 48	BK-R68R-48
9	Road-450 Red, 52	BK-R68R-52
10	Road-650 Red, 58	BK-R50R-58
11	Road-650 Red, 60	BK-R50R-60
12	Road-650 Red, 62	BK-R50R-62
13	Road-650 Red, 44	BK-R50R-44
14	Road-650 Red, 48	BK-R50R-48
15	Road-650 Red, 52	BK-R50R-52
16	Road-650 Black, 58	BK-R50B-58
17	Road-650 Black, 60	BK-R50B-60

At the bottom of the results pane, it says "Query executed successfully." and shows the session details: demoazure1010.database.windows.net | sqladmin (70) | demodb | 00:00:00 | 151 rows.

- Now if you want to get the records from the table based on the string keyword within the name.

The screenshot shows a SQL Server Management Studio window. The query in the top pane is:

```
SELECT ProductID, Name, ProductNumber FROM SalesLT.Product WHERE Name Like '%Road%'
```

The results pane displays a table with three columns: ProductID, Name, and ProductNumber. The data is as follows:

ProductID	Name	ProductNumber
1	Road Bottle Cage	BC-R205
2	Road-750 Black, 44	BK-R19B-44
3	Road-750 Black, 48	BK-R19B-48
4	Road-750 Black, 52	BK-R19B-52
5	Road-750 Black, 58	BK-R19B-58
6	Road-650 Black, 44	BK-R50B-44
7	Road-650 Black, 48	BK-R50B-48
8	Road-650 Black, 52	BK-R50B-52
9	Road-650 Black, 58	BK-R50B-58
10	Road-650 Black, 60	BK-R50B-60
11	Road-650 Black, 62	BK-R50B-62
12	Road-650 Red, 44	BK-R50R-44
13	Road-650 Red, 48	BK-R50R-48
14	Road-650 Red, 52	BK-R50R-52
15	Road-650 Red, 58	BK-R50R-58
16	Road-650 Red, 60	BK-R50R-60
17	Road-650 Red, 62	BK-R50R-62

At the bottom of the results pane, it says "Query executed successfully." and shows the session details: demoazure1010.database.windows.net | sqladmin (70) | demodb | 00:00:00 | 96 rows.

😊 ORDER BY Clause

- This clause can be used to sort the results set, the data that's coming in from the table, based on your selected statement in ascending or descending order.

2. Now run this query and you will see that you are getting the information in ascending order.

The screenshot shows a SQL Server Management Studio window with the following details:

- Query Editor:** The query is `SELECT * FROM SalesLT.Product ORDER BY Listprice`.
- Results Grid:** The results show 295 rows of product data. The columns include ProductID, Name, ProductNumber, Color, StandardCost, ListPrice, Size, Weight, ProductCategoryID, ProductModelID, SellStartDate, SellEndDate, and DiscontinuedDate.
- Status Bar:** It shows "Query executed successfully." and the execution details: demoazure1010.database.windows.net | sqladmin (70) | demodb | 00:00:02 | 295 rows.

ProductID	Name	ProductNumber	Color	StandardCost	ListPrice	Size	Weight	ProductCategoryID	ProductModelID	SellStartDate	SellEndDate	DiscontinuedDate
1	Patch Kit& Patches	PK-7098	NULL	0.8565	2.29	NULL	NULL	41	114	2007-07-01 00:00:00.000	NULL	NULL
2	Road Tire Tube	TT-R982	NULL	1.4923	3.99	NULL	NULL	41	93	2007-07-01 00:00:00.000	NULL	NULL
3	Touring Tire Tube	TT-T092	NULL	1.8663	4.99	NULL	NULL	41	94	2007-07-01 00:00:00.000	NULL	NULL
4	Mountain Tire Tube	TT-M928	NULL	1.8663	4.99	NULL	NULL	41	92	2007-07-01 00:00:00.000	NULL	NULL
5	Water Bottle - 30 oz.	WB-H098	NULL	1.8663	4.99	NULL	NULL	32	111	2007-07-01 00:00:00.000	NULL	NULL
6	Bike Wash - Dissolver	CL-9009	NULL	2.9733	7.95	NULL	NULL	33	119	2007-07-01 00:00:00.000	NULL	NULL
7	Road Bottle Cage	BC-R205	NULL	3.3623	8.99	NULL	NULL	32	113	2007-07-01 00:00:00.000	NULL	NULL
8	Racing Socks, M	SO-R809-M	White	3.3623	8.99	M	NULL	27	24	2007-07-01 00:00:00.000	NULL	NULL
9	Racing Socks, L	SO-R809-L	White	3.3623	8.99	L	NULL	27	24	2007-07-01 00:00:00.000	NULL	NULL
10	AWC Logo Cap	CA-1088	Multi	6.9223	8.99	NULL	NULL	23	2	2005-07-01 00:00:00.000	NULL	NULL
11	Mountain Bike Socks, M	SO-B909-M	White	3.3963	9.50	M	NULL	27	18	2005-07-01 00:00:00.000	2006-06-30 00:00:00.000	NULL
12	Mountain Bike Socks, L	SO-B909-L	White	3.3963	9.50	L	NULL	27	18	2005-07-01 00:00:00.000	2006-06-30 00:00:00.000	NULL
13	Mountain Bike Cage	BC-M005	NULL	3.7363	9.99	NULL	NULL	32	112	2007-07-01 00:00:00.000	NULL	NULL
14	Taillights - Battery-Powered	LT-T990	NULL	5.7709	13.99	NULL	NULL	37	108	2006-07-01 00:00:00.000	2007-06-30 00:00:00.000	NULL
15	Minipump	PU-0452	NULL	8.2459	19.99	NULL	NULL	40	116	2006-07-01 00:00:00.000	2007-06-30 00:00:00.000	NULL
16	Chain	CH-0234	Silver	8.9866	20.24	NULL	NULL	11	98	2007-07-01 00:00:00.000	NULL	NULL

3. Now to get the information in descending order you can run this command.

The screenshot shows a SQL Server Management Studio window with the following details:

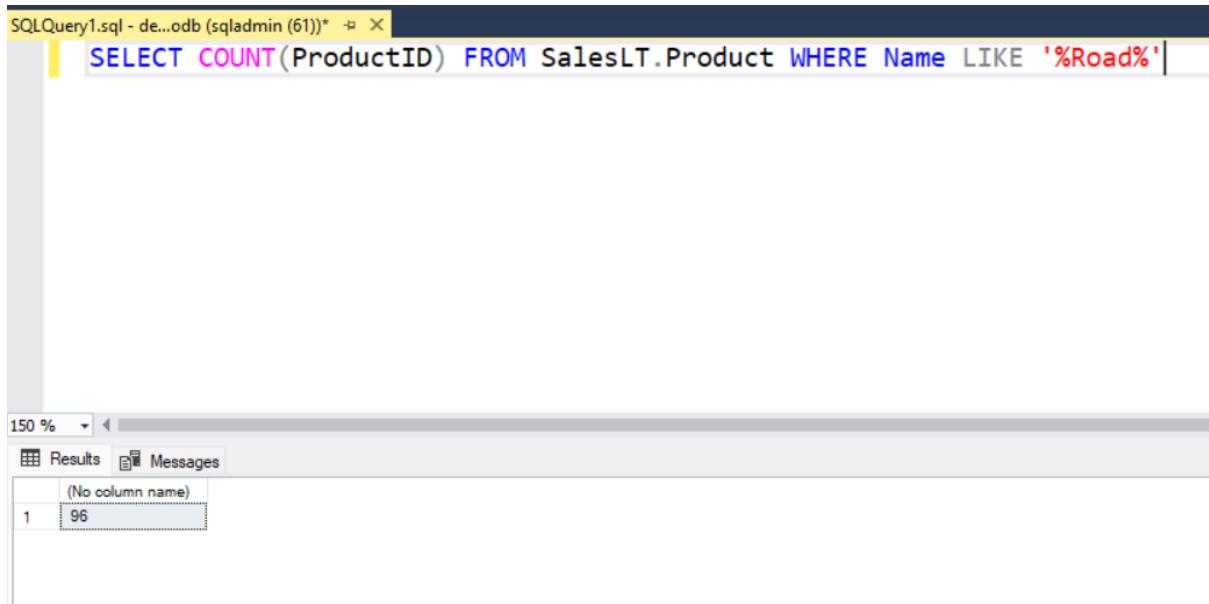
- Query Editor:** The query is `SELECT * FROM SalesLT.Product ORDER BY Listprice DESC`.
- Results Grid:** The results show 295 rows of product data, identical to the first query but ordered in descending order. The columns include ProductID, Name, ProductNumber, Color, StandardCost, ListPrice, Size, Weight, ProductCategoryID, ProductModelID, SellStartDate, SellEndDate, and DiscontinuedDate.
- Status Bar:** It shows "Query executed successfully." and the execution details: demoazure1010.database.windows.net | sqladmin (61) | demodb | 00:00:00 | 295 rows.

ProductID	Name	ProductNumber	Color	StandardCost	ListPrice	Size	Weight	ProductCategoryID	ProductModelID	SellStartDate	SellEndDate	DiscontinuedDate
1	Road-150 Red, 62	BK-R93R-62	Red	2171.2942	3578.27	62	6803.85	6	25	2005-07-01 00:00:00.000	2006-06-30 00:00:00.000	NULL
2	Road-150 Red, 44	BK-R93R-44	Red	2171.2942	3578.27	44	6245.93	6	25	2005-07-01 00:00:00.000	2006-06-30 00:00:00.000	NULL
3	Road-150 Red, 48	BK-R93R-48	Red	2171.2942	3578.27	48	6409.23	6	25	2005-07-01 00:00:00.000	2006-06-30 00:00:00.000	NULL
4	Road-150 Red, 52	BK-R93R-52	Red	2171.2942	3578.27	52	6540.77	6	25	2005-07-01 00:00:00.000	2006-06-30 00:00:00.000	NULL
5	Road-150 Red, 56	BK-R93R-56	Red	2171.2942	3578.27	56	6658.70	6	25	2005-07-01 00:00:00.000	2006-06-30 00:00:00.000	NULL
6	Mountain-100 Silver, 38	BK-M82S-38	Silver	1912.1544	3399.99	38	9230.56	5	19	2005-07-01 00:00:00.000	2006-06-30 00:00:00.000	NULL
7	Mountain-100 Silver, 42	BK-M82S-42	Silver	1912.1544	3399.99	42	9421.06	5	19	2005-07-01 00:00:00.000	2006-06-30 00:00:00.000	NULL
8	Mountain-100 Silver, 44	BK-M82S-44	Silver	1912.1544	3399.99	44	9584.36	5	19	2005-07-01 00:00:00.000	2006-06-30 00:00:00.000	NULL
9	Mountain-100 Silver, 48	BK-M82S-48	Silver	1912.1544	3399.99	48	9715.90	5	19	2005-07-01 00:00:00.000	2006-06-30 00:00:00.000	NULL
10	Mountain-100 Black, 38	BK-M82B-38	Black	1898.0944	3374.99	38	9230.56	5	19	2005-07-01 00:00:00.000	2006-06-30 00:00:00.000	NULL
11	Mountain-100 Black, 42	BK-M82B-42	Black	1898.0944	3374.99	42	9421.06	5	19	2005-07-01 00:00:00.000	2006-06-30 00:00:00.000	NULL
12	Mountain-100 Black, 44	BK-M82B-44	Black	1898.0944	3374.99	44	9584.36	5	19	2005-07-01 00:00:00.000	2006-06-30 00:00:00.000	NULL
13	Mountain-100 Black, 48	BK-M82B-48	Black	1898.0944	3374.99	48	9715.90	5	19	2005-07-01 00:00:00.000	2006-06-30 00:00:00.000	NULL
14	Road-250 Red, 44	BK-R89R-44	Red	1518.7864	2443.35	44	6699.52	6	26	2006-07-01 00:00:00.000	2007-06-30 00:00:00.000	NULL
15	Road-250 Red, 48	BK-R89R-48	Red	1518.7864	2443.35	48	6862.82	6	26	2006-07-01 00:00:00.000	2007-06-30 00:00:00.000	NULL
16	Road-250 Red, 52	BK-R89R-52	Red	1518.7864	2443.35	52	6994.36	6	26	2006-07-01 00:00:00.000	2007-06-30 00:00:00.000	NULL

Aggregate Functions

1. In this part we will learn how to use aggregate functions.

2. If you want to aggregate your data, there are functions available within the structure query language. We have already seen one function before and that is the count function. If you want to look at the number of records that are being returned by a particular statement, a select statement, we can go ahead and use the count function.



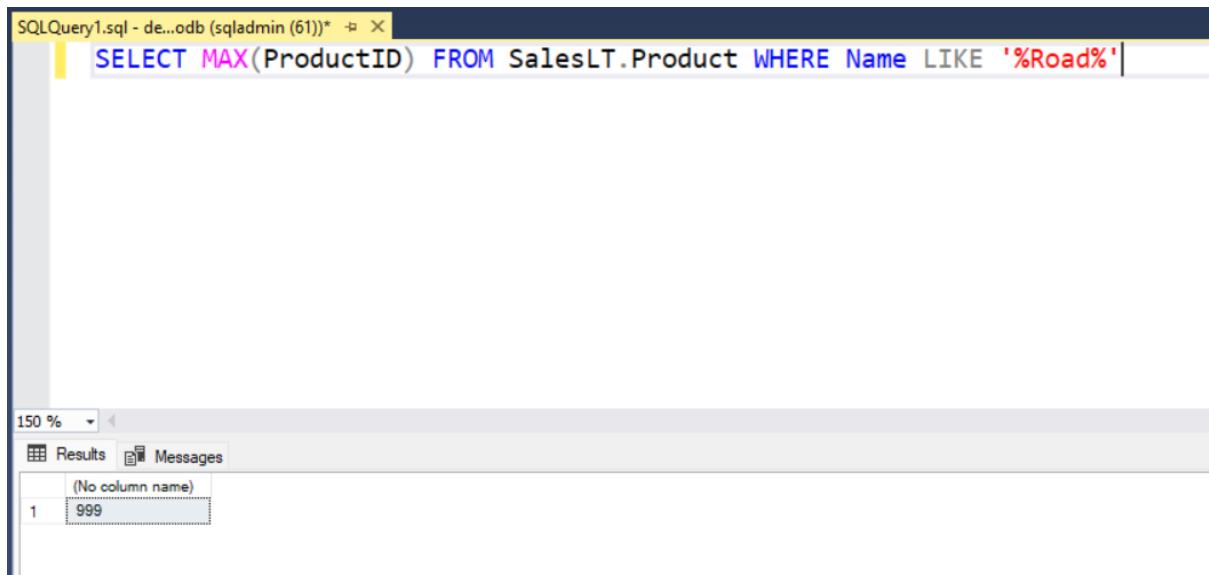
The screenshot shows a SQL Server Management Studio (SSMS) interface. The top bar indicates the connection is to 'SQLQuery1.sql - de...odb (sqladmin (61))'. A query window contains the following T-SQL code:

```
SELECT COUNT(ProductID) FROM SalesLT.Product WHERE Name LIKE '%Road%'
```

The results pane below shows a single row of data:

(No column name)
1
96

3. Below are some examples of how you can use Aggregate functions.



The screenshot shows a SQL Server Management Studio (SSMS) interface. The top bar indicates the connection is to 'SQLQuery1.sql - de...odb (sqladmin (61))'. A query window contains the following T-SQL code:

```
SELECT MAX(ProductID) FROM SalesLT.Product WHERE Name LIKE '%Road%'
```

The results pane below shows a single row of data:

(No column name)
1
999

SQLQuery1.sql - de...odb (sqladmin (61))*

```
SELECT MIN(ProductID) FROM SalesLT.Product WHERE Name LIKE '%Road%'
```

150 %

Results Messages

(No column name)
1 680

SQLQuery1.sql - de...odb (sqladmin (61))*

```
SELECT SUM(ProductID) FROM SalesLT.Product WHERE Name LIKE '%Road%'
```

150 %

Results Messages

(No column name)
1 7735

SQLQuery1.sql - de...odb (sqladmin (61))*

```
SELECT AVG(ProductID) FROM SalesLT.Product WHERE Name LIKE '%Road%'
```

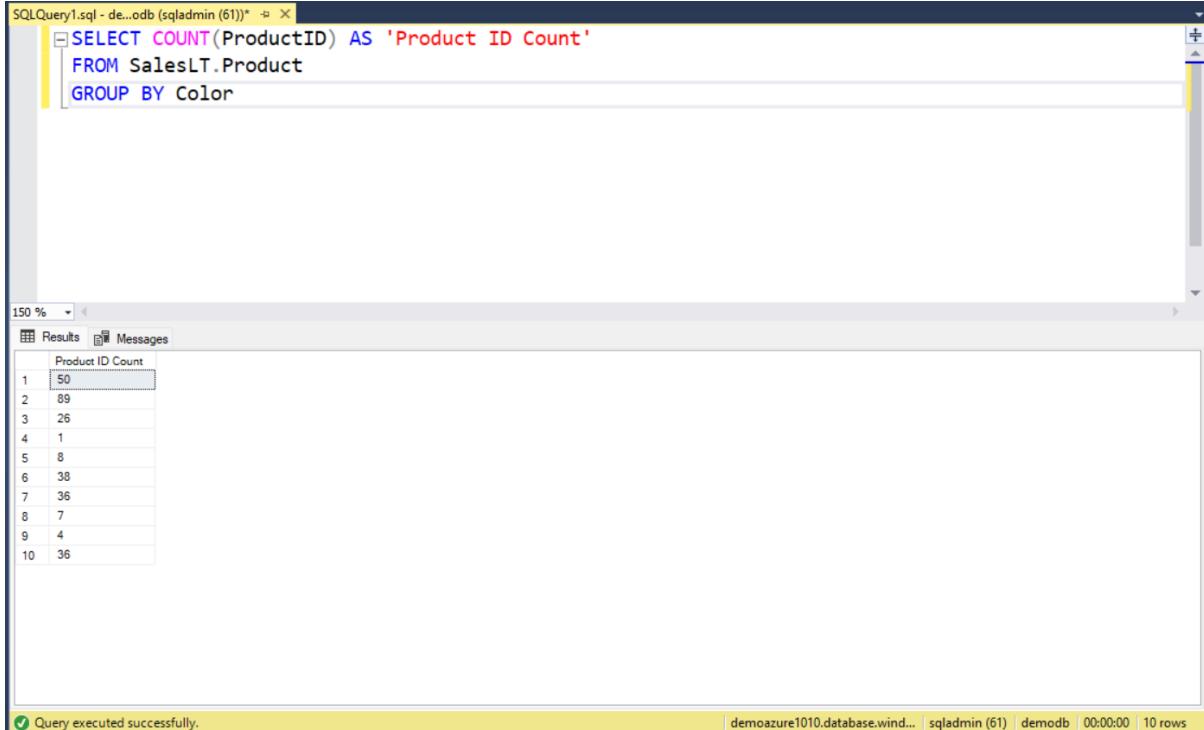
150 %

Results Messages

(No column name)
1 805

GROUP BY Clause

1. Now the group by clause is used to group rows into summary rows and this can be used along with your aggregate functions.
2. Below are some examples on how you can use this clause.



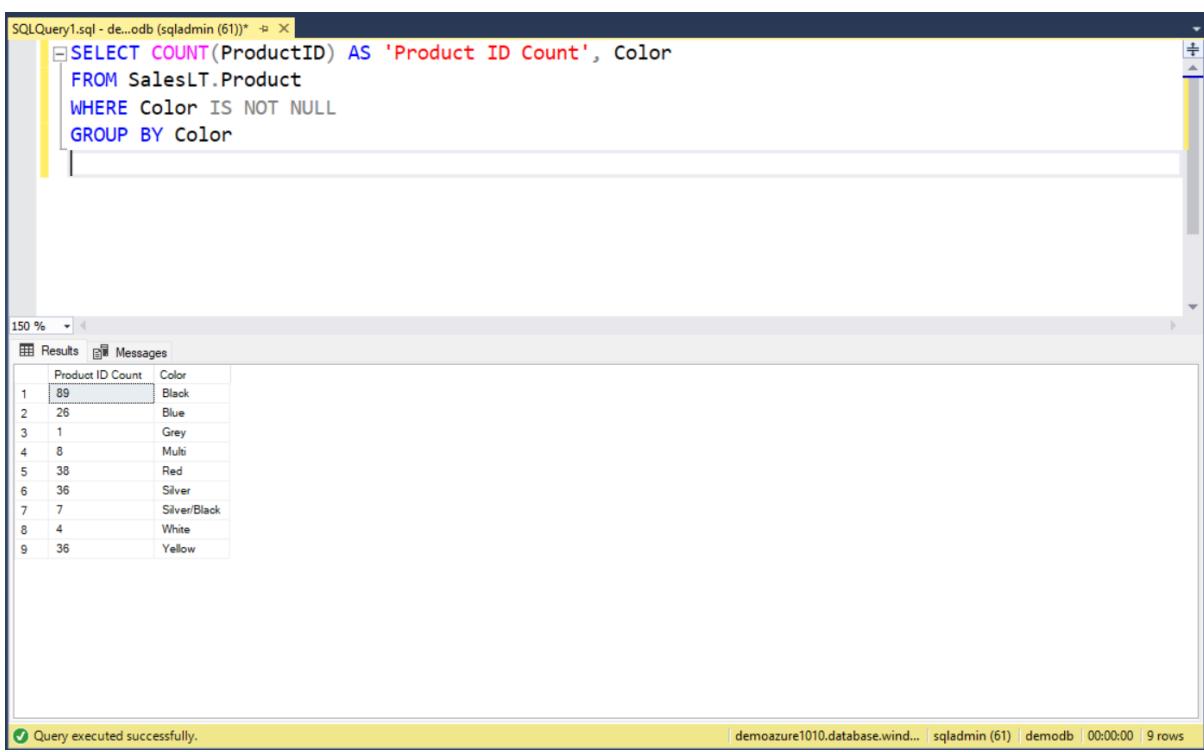
SQLQuery1.sql - de...odb (sqladmin (61))*

```
SELECT COUNT(ProductID) AS 'Product ID Count'
FROM SalesLT.Product
GROUP BY Color
```

Results Messages

	Product ID Count
1	50
2	89
3	26
4	1
5	8
6	38
7	36
8	7
9	4
10	36

Query executed successfully. demoazure1010.database.wind... sqladmin (61) demodb 00:00:00 10 rows



SQLQuery1.sql - de...odb (sqladmin (61))*

```
SELECT COUNT(ProductID) AS 'Product ID Count', Color
FROM SalesLT.Product
WHERE Color IS NOT NULL
GROUP BY Color
```

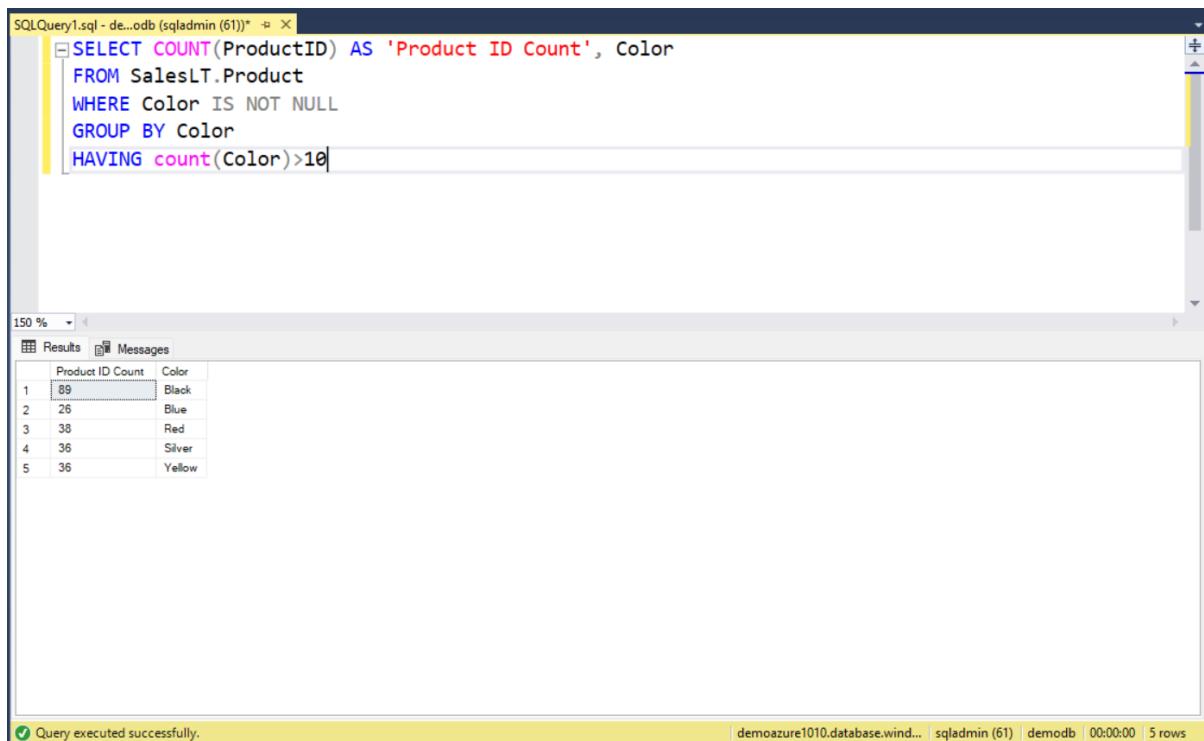
Results Messages

	Product ID Count	Color
1	89	Black
2	26	Blue
3	1	Grey
4	8	Multi
5	38	Red
6	36	Silver
7	7	Silver/Black
8	4	White
9	36	Yellow

Query executed successfully. demoazure1010.database.wind... sqladmin (61) demodb 00:00:00 9 rows

HAVING Clause

- When you start using aggregate functions from within your queries, you can use the WHERE clause to filter on the information that's first returned in the results set. But after the GROUP BY clause, you know, once you have the results set in place, after the filter condition has been applied from the WHERE clause, and we use a GROUP BY clause to our group by Color, let's say in this case, you want to filter down further. We can make use of the HAVING clause.



The screenshot shows a SQL Server Management Studio window. The Query Editor pane contains the following T-SQL code:

```
SELECT COUNT(ProductID) AS 'Product ID Count', Color
FROM SalesLT.Product
WHERE Color IS NOT NULL
GROUP BY Color
HAVING count(Color)>10;
```

The Results pane displays the following table:

	Product ID Count	Color
1	89	Black
2	26	Blue
3	38	Red
4	36	Silver
5	36	Yellow

At the bottom of the Results pane, a status bar indicates: "Query executed successfully." and "demoazure1010.database.windows.net | sqladmin (61) | demodb | 00:00:00 | 5 rows".