SCENARIO 1

Sort a result set by one column in ascending or descending order

Consider the table **[Person].[CountryRegion]** and it has the following data:
(To execute any query, select the query and go to the menu option Query -> Execute
 OR simply **select the query and hit F5**)

```
select * from [Person].[CountryRegion]
```
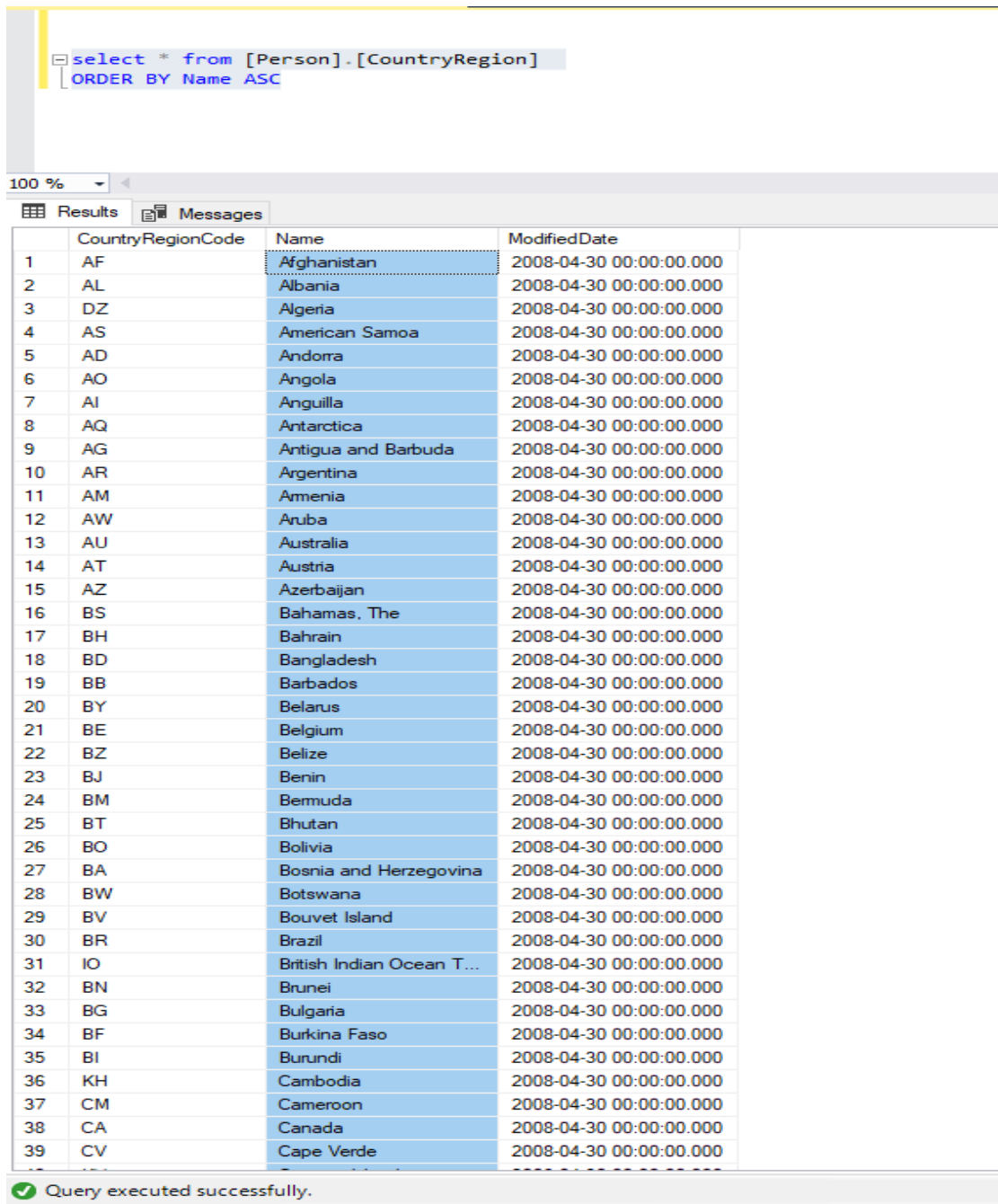
100 %

Results | Messages

	CountryRegionCode	Name	ModifiedDate
1	AD	Andorra	2008-04-30 00:00:00.000
2	AE	United Arab Emirates	2008-04-30 00:00:00.000
3	AF	Afghanistan	2008-04-30 00:00:00.000
4	AG	Antigua and Barbuda	2008-04-30 00:00:00.000
5	AI	Anguilla	2008-04-30 00:00:00.000
6	AL	Albania	2008-04-30 00:00:00.000
7	AM	Armenia	2008-04-30 00:00:00.000
8	AN	Netherlands Antilles	2008-04-30 00:00:00.000
9	AO	Angola	2008-04-30 00:00:00.000
10	AQ	Antarctica	2008-04-30 00:00:00.000
11	AR	Argentina	2008-04-30 00:00:00.000
12	AS	American Samoa	2008-04-30 00:00:00.000
13	AT	Austria	2008-04-30 00:00:00.000
14	AU	Australia	2008-04-30 00:00:00.000
15	AW	Aruba	2008-04-30 00:00:00.000
16	AZ	Azerbaijan	2008-04-30 00:00:00.000
17	BA	Bosnia and Herzegovina	2008-04-30 00:00:00.000
18	BB	Barbados	2008-04-30 00:00:00.000
19	BD	Bangladesh	2008-04-30 00:00:00.000
20	BE	Belgium	2008-04-30 00:00:00.000
21	BF	Burkina Faso	2008-04-30 00:00:00.000
22	BG	Bulgaria	2008-04-30 00:00:00.000
23	BH	Bahrain	2008-04-30 00:00:00.000
24	BI	Burundi	2008-04-30 00:00:00.000
25	BJ	Benin	2008-04-30 00:00:00.000
26	BM	Bermuda	2008-04-30 00:00:00.000
27	BN	Brunei	2008-04-30 00:00:00.000
28	BO	Bolivia	2008-04-30 00:00:00.000
29	BR	Brazil	2008-04-30 00:00:00.000
30	BS	Bahamas, The	2008-04-30 00:00:00.000
31	BT	Bhutan	2008-04-30 00:00:00.000
32	BV	Bouvet Island	2008-04-30 00:00:00.000
33	BW	Botswana	2008-04-30 00:00:00.000
34	BY	Belarus	2008-04-30 00:00:00.000
35	BZ	Belize	2008-04-30 00:00:00.000
36	CA	Canada	2008-04-30 00:00:00.000
37	CC	Cocos (Keeling) Islands	2008-04-30 00:00:00.000
38	CD	Congo (DRC)	2008-04-30 00:00:00.000
39	CF	Central African Republic	2008-04-30 00:00:00.000

✓ Query executed successfully.

You can see the 'Name' column having the name of different countries but in different order.

If you want to order the country names in the ascending order from A-Z, you use **ORDER BY** clause for the column **'Name'** along with the ASC keyword.

select * from [Person].[CountryRegion]
ORDER BY Name ASC

```
select * from [Person].[CountryRegion]
ORDER BY Name ASC
```

100 %

▦ Results   ▤ Messages

	CountryRegionCode	Name	ModifiedDate
1	AF	Afghanistan	2008-04-30 00:00:00.000
2	AL	Albania	2008-04-30 00:00:00.000
3	DZ	Algeria	2008-04-30 00:00:00.000
4	AS	American Samoa	2008-04-30 00:00:00.000
5	AD	Andorra	2008-04-30 00:00:00.000
6	AO	Angola	2008-04-30 00:00:00.000
7	AI	Anguilla	2008-04-30 00:00:00.000
8	AQ	Antarctica	2008-04-30 00:00:00.000
9	AG	Antigua and Barbuda	2008-04-30 00:00:00.000
10	AR	Argentina	2008-04-30 00:00:00.000
11	AM	Armenia	2008-04-30 00:00:00.000
12	AW	Aruba	2008-04-30 00:00:00.000
13	AU	Australia	2008-04-30 00:00:00.000
14	AT	Austria	2008-04-30 00:00:00.000
15	AZ	Azerbaijan	2008-04-30 00:00:00.000
16	BS	Bahamas, The	2008-04-30 00:00:00.000
17	BH	Bahrain	2008-04-30 00:00:00.000
18	BD	Bangladesh	2008-04-30 00:00:00.000
19	BB	Barbados	2008-04-30 00:00:00.000
20	BY	Belarus	2008-04-30 00:00:00.000
21	BE	Belgium	2008-04-30 00:00:00.000
22	BZ	Belize	2008-04-30 00:00:00.000
23	BJ	Benin	2008-04-30 00:00:00.000
24	BM	Bermuda	2008-04-30 00:00:00.000
25	BT	Bhutan	2008-04-30 00:00:00.000
26	BO	Bolivia	2008-04-30 00:00:00.000
27	BA	Bosnia and Herzegovina	2008-04-30 00:00:00.000
28	BW	Botswana	2008-04-30 00:00:00.000
29	BV	Bouvet Island	2008-04-30 00:00:00.000
30	BR	Brazil	2008-04-30 00:00:00.000
31	IO	British Indian Ocean T...	2008-04-30 00:00:00.000
32	BN	Brunei	2008-04-30 00:00:00.000
33	BG	Bulgaria	2008-04-30 00:00:00.000
34	BF	Burkina Faso	2008-04-30 00:00:00.000
35	BI	Burundi	2008-04-30 00:00:00.000
36	KH	Cambodia	2008-04-30 00:00:00.000
37	CM	Cameroon	2008-04-30 00:00:00.000
38	CA	Canada	2008-04-30 00:00:00.000
39	CV	Cape Verde	2008-04-30 00:00:00.000

✔ Query executed successfully.

In case if you want to order the country names in the descending order from Z-A, you use **ORDER BY** clause for the column 'Name' along with the **DESC** keyword.

```sql
select * from [Person].[CountryRegion]
ORDER BY Name DESC
```

100 %

Results | **Messages**

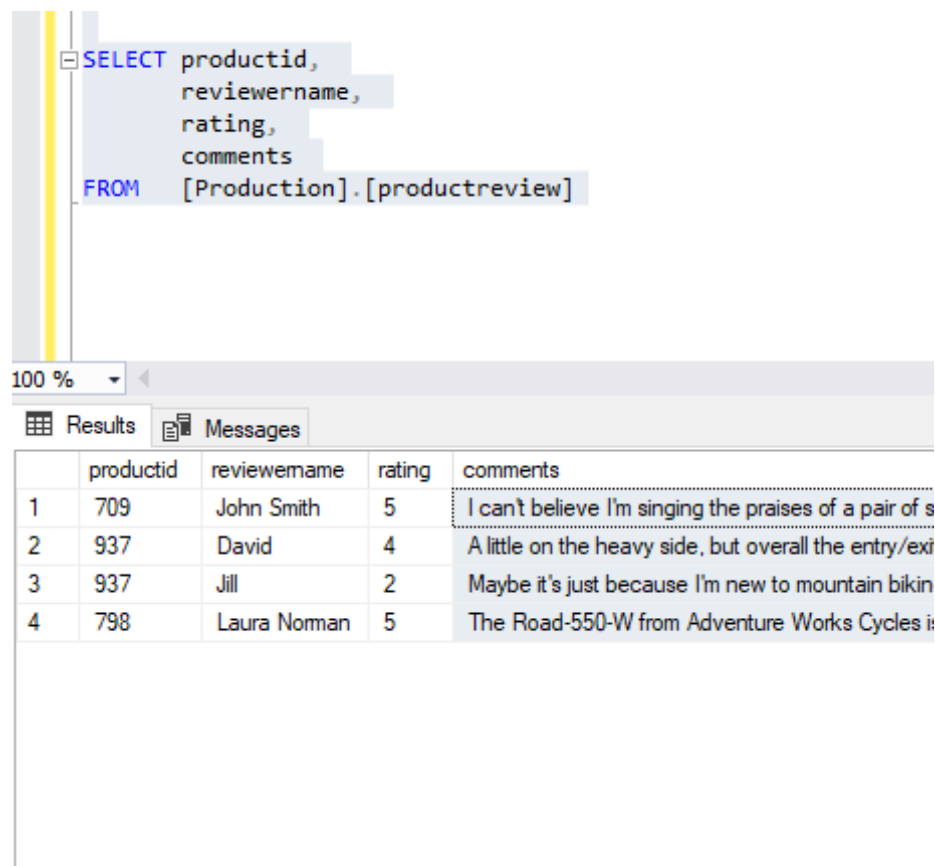|	CountryRegionCode	Name	ModifiedDate
1	ZW	Zimbabwe	2008-04-30 00:00:00.000
2	ZM	Zambia	2008-04-30 00:00:00.000
3	YE	Yemen	2008-04-30 00:00:00.000
4	WF	Wallis and Futuna	2008-04-30 00:00:00.000
5	VI	Virgin Islands, U.S.	2008-04-30 00:00:00.000
6	VG	Virgin Islands, British	2008-04-30 00:00:00.000
7	VN	Vietnam	2008-04-30 00:00:00.000
8	VE	Venezuela	2008-04-30 00:00:00.000
9	VA	Vatican City	2008-04-30 00:00:00.000
10	VU	Vanuatu	2008-04-30 00:00:00.000
11	UZ	Uzbekistan	2008-04-30 00:00:00.000
12	UY	Uruguay	2008-04-30 00:00:00.000
13	US	United States	2008-04-30 00:00:00.000
14	GB	United Kingdom	2008-04-30 00:00:00.000
15	AE	United Arab Emirates	2008-04-30 00:00:00.000
16	UA	Ukraine	2008-04-30 00:00:00.000
17	UG	Uganda	2008-04-30 00:00:00.000
18	UM	U.S. Minor Outlying Islands	2008-04-30 00:00:00.000
19	TV	Tuvalu	2008-04-30 00:00:00.000
20	TC	Turks and Caicos Islands	2008-04-30 00:00:00.000
21	TM	Turkmenistan	2008-04-30 00:00:00.000
22	TR	Turkey	2008-04-30 00:00:00.000
23	TN	Tunisia	2008-04-30 00:00:00.000
24	TT	Trinidad and Tobago	2008-04-30 00:00:00.000
25	TO	Tonga	2008-04-30 00:00:00.000
26	TK	Tokelau	2008-04-30 00:00:00.000
27	TG	Togo	2008-04-30 00:00:00.000
28	TL	Timor-Leste	2008-04-30 00:00:00.000
29	TH	Thailand	2008-04-30 00:00:00.000
30	TZ	Tanzania	2008-04-30 00:00:00.000
31	TJ	Tajikistan	2008-04-30 00:00:00.000
32	TW	Taiwan	2008-04-30 00:00:00.000
33	SY	Syria	2008-04-30 00:00:00.000
34	CH	Switzerland	2008-04-30 00:00:00.000
35	SE	Sweden	2008-04-30 00:00:00.000
36	SZ	Swaziland	2008-04-30 00:00:00.000
37	SJ	Svalbard and Jan Mayen	2008-04-30 00:00:00.000
38	SR	Suriname	2008-04-30 00:00:00.000
39	SD	Sudan	2008-04-30 00:00:00.000

Query executed successfully.

SCENARIO 2

Sort a result set by an expression

Consider the Product Review table **[Production].[ProductReview]**
Query this table to find the Product ID, Reviewer name, rating and comments people left for the products.

SELECT productid,
    reviewername,
    rating,
    comments
FROM   [Production].[productreview]

```
SELECT productid,
       reviewername,
       rating,
       comments
FROM   [Production].[productreview]
```

100 %   ▾ ◂

▦ Results   ▤ Messages

	productid	reviewername	rating	comments
1	709	John Smith	5	I can't believe I'm singing the praises of a pair of s
2	937	David	4	A little on the heavy side, but overall the entry/exi
3	937	Jill	2	Maybe it's just because I'm new to mountain bikin
4	798	Laura Norman	5	The Road-550-W from Adventure Works Cycles is

Now we want to list the *comments with more words* to get more insight about a product.

In order to list the comments with more words, you use the **ORDER BY** clause with the expression **LEN(Comments)** and key word **DESC**.**LEN(Comments)** will compute the length of each comment and the key word DESC will order the comment with the most word on top.

SELECT productid,
    reviewername,
    rating,
    comments
FROM   [Production].[productreview]
ORDER BY Len(comments) DESC

```
SELECT productid,
        reviewername,
        rating,
        comments
FROM    [Production].[productreview]
ORDER   BY Len(comments) DESC
```

100 %

Results | Messages

	productid	reviewername	rating	comments
1	798	Laura Norman	5	The Road-550-W from Adventure Works Cycles is everything it's advertised t
2	709	John Smith	5	I can't believe I'm singing the praises of a pair of socks, but I just came back f
3	937	David	4	A little on the heavy side, but overall the entry/exit is easy in all conditions. I'v
4	937	Jill	2	Maybe it's just because I'm new to mountain biking, but I had a terrible time ge

In the Output, you can see that the comments with more words got to the top and the words with less words got to the bottom in the column "comments".

SCENARIO 3

Retrieve 10 percent of the result set

Consider the table **[Production].[TransactionHistory]** storing the transactions related with products. You can execute the below query to see the transactions ordered by the most recent date.

```sql
SELECT transactionid,
    productid,
    transactiondate,
    transactiontype
FROM   [Production].[transactionhistory]
ORDER BY TransactionDate DESC
```

```sql
SELECT transactionid,
       productid,
       transactiondate,
       transactiontype
FROM   [Production].[transactionhistory]
ORDER BY TransactionDate DESC
```

100 %

Results | Messages

	transactionid	productid	transactiondate	transactiontype
1	213406	358	2014-08-03 00:00:00.000	P
2	213407	375	2014-08-03 00:00:00.000	P
3	213408	376	2014-08-03 00:00:00.000	P
4	213409	377	2014-08-03 00:00:00.000	P
5	213410	378	2014-08-03 00:00:00.000	P
6	213411	356	2014-08-03 00:00:00.000	P
7	213412	357	2014-08-03 00:00:00.000	P
8	213413	394	2014-08-03 00:00:00.000	P
9	213414	395	2014-08-03 00:00:00.000	P
10	213415	396	2014-08-03 00:00:00.000	P
11	213416	397	2014-08-03 00:00:00.000	P
12	213417	490	2014-08-03 00:00:00.000	P
13	213418	513	2014-08-03 00:00:00.000	P
14	213419	410	2014-08-03 00:00:00.000	P
15	213420	411	2014-08-03 00:00:00.000	P
16	213421	412	2014-08-03 00:00:00.000	P
17	213422	413	2014-08-03 00:00:00.000	P
18	213423	931	2014-08-03 00:00:00.000	P
19	213424	932	2014-08-03 00:00:00.000	P
20	213425	424	2014-08-03 00:00:00.000	P
21	213426	425	2014-08-03 00:00:00.000	P
22	213427	426	2014-08-03 00:00:00.000	P
23	213428	341	2014-08-03 00:00:00.000	P
24	213429	342	2014-08-03 00:00:00.000	P
25	213430	931	2014-08-03 00:00:00.000	P
26	213431	932	2014-08-03 00:00:00.000	P
27	213432	347	2014-08-03 00:00:00.000	P
28	213433	348	2014-08-03 00:00:00.000	P
29	213434	349	2014-08-03 00:00:00.000	P
30	213435	928	2014-08-03 00:00:00.000	P
31	213436	929	2014-08-03 00:00:00.000	P
32	213437	930	2014-08-03 00:00:00.000	P
33	213438	332	2014-08-03 00:00:00.000	P
34	213439	907	2014-08-03 00:00:00.000	P
35	213440	948	2014-08-03 00:00:00.000	P
36	213441	325	2014-08-03 00:00:00.000	P
37	213442	326	2014-08-03 00:00:00.000	P

Query executed successfully.

A total of **113,443 rows** will be returned. You can see this at the bottom of the screen.

AdventureWorks2017 | 00:00:00 | 113,443 rows

Now the requirement is to get only the first 10 percent rows.
10 percent of 113,443 is a fractional value (11344.3), SQL Server rounds up to the next whole number which is **11,345** in this case. So, to get the first 10 percent rows of the result set (**11,345**) you use the **TOP 10 PERCENT** keyword.

SELECT TOP 10 PERCENT transactionid,

```
        productid,
        transactiondate,
        transactiontype
FROM   [Production].[transactionhistory]
ORDER BY TransactionDate DESC
```

```
SELECT TOP 10 PERCENT transactionid,
       productid,
       transactiondate,
       transactiontype
FROM   [Production].[transactionhistory]
ORDER BY TransactionDate DESC
```

100 %

	transactionid	productid	transactiondate	transactiontype
1	213406	358	2014-08-03 00:00:00.000	P
2	213407	375	2014-08-03 00:00:00.000	P
3	213408	376	2014-08-03 00:00:00.000	P
4	213409	377	2014-08-03 00:00:00.000	P
5	213410	378	2014-08-03 00:00:00.000	P
6	213411	356	2014-08-03 00:00:00.000	P
7	213412	357	2014-08-03 00:00:00.000	P
8	213413	394	2014-08-03 00:00:00.000	P
9	213414	395	2014-08-03 00:00:00.000	P
10	213415	396	2014-08-03 00:00:00.000	P
11	213416	397	2014-08-03 00:00:00.000	P

Now, you can see at the bottom of the screen that 10 percent of the result set (**11,345**) got returned.
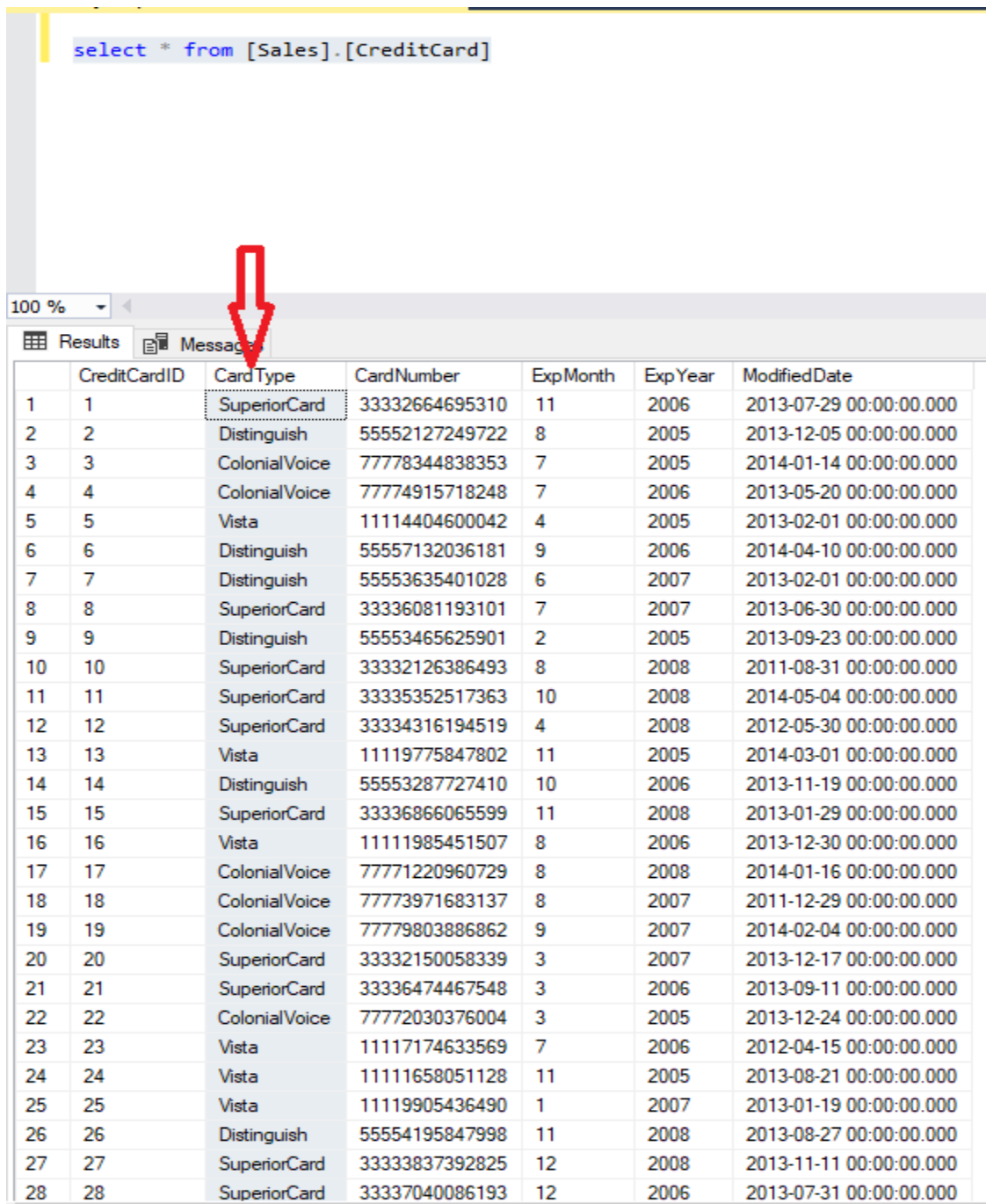
AdventureWorks2017 | 00:00:00 | 11,345 rows

SCENARIO 4

Retrieve distinct values

Consider the table **[Sales].[CreditCard]**, it has different Card types.

select * from [Sales].[CreditCard]
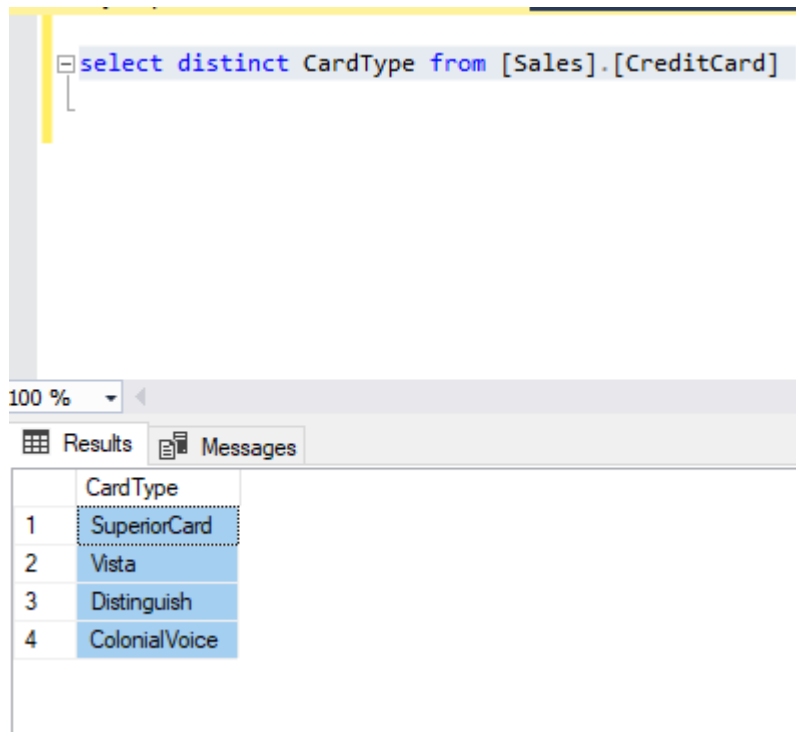
```
select * from [Sales].[CreditCard]
```


100 %

Results   Messages

	CreditCardID	Card Type	CardNumber	Exp Month	Exp Year	ModifiedDate
1	1	SuperiorCard	33332664695310	11	2006	2013-07-29 00:00:00.000
2	2	Distinguish	55552127249722	8	2005	2013-12-05 00:00:00.000
3	3	ColonialVoice	77778344838353	7	2005	2014-01-14 00:00:00.000
4	4	ColonialVoice	77774915718248	7	2006	2013-05-20 00:00:00.000
5	5	Vista	11114404600042	4	2005	2013-02-01 00:00:00.000
6	6	Distinguish	55557132036181	9	2006	2014-04-10 00:00:00.000
7	7	Distinguish	55553635401028	6	2007	2013-02-01 00:00:00.000
8	8	SuperiorCard	33336081193101	7	2007	2013-06-30 00:00:00.000
9	9	Distinguish	55553465625901	2	2005	2013-09-23 00:00:00.000
10	10	SuperiorCard	33332126386493	8	2008	2011-08-31 00:00:00.000
11	11	SuperiorCard	33335352517363	10	2008	2014-05-04 00:00:00.000
12	12	SuperiorCard	33334316194519	4	2008	2012-05-30 00:00:00.000
13	13	Vista	11119775847802	11	2005	2014-03-01 00:00:00.000
14	14	Distinguish	55553287727410	10	2006	2013-11-19 00:00:00.000
15	15	SuperiorCard	33336866065599	11	2008	2013-01-29 00:00:00.000
16	16	Vista	11111985451507	8	2006	2013-12-30 00:00:00.000
17	17	ColonialVoice	77771220960729	8	2008	2014-01-16 00:00:00.000
18	18	ColonialVoice	77773971683137	8	2007	2011-12-29 00:00:00.000
19	19	ColonialVoice	77779803886862	9	2007	2014-02-04 00:00:00.000
20	20	SuperiorCard	33332150058339	3	2007	2013-12-17 00:00:00.000
21	21	SuperiorCard	33336474467548	3	2006	2013-09-11 00:00:00.000
22	22	ColonialVoice	77772030376004	3	2005	2013-12-24 00:00:00.000
23	23	Vista	11117174633569	7	2006	2012-04-15 00:00:00.000
24	24	Vista	11111658051128	11	2005	2013-08-21 00:00:00.000
25	25	Vista	11119905436490	1	2007	2013-01-19 00:00:00.000
26	26	Distinguish	55554195847998	11	2008	2013-08-27 00:00:00.000
27	27	SuperiorCard	33333837392825	12	2008	2013-11-11 00:00:00.000
28	28	SuperiorCard	33337040086193	12	2006	2013-07-31 00:00:00.000

As you can see from the output, the Card Types are duplicate. Now you want to find the total Card Types without any duplication. To do that you add the **DISTINCT** keyword as follows:

select distinct CardType from [Sales].[CreditCard]

```sql
select distinct CardType from [Sales].[CreditCard]
```

100 %

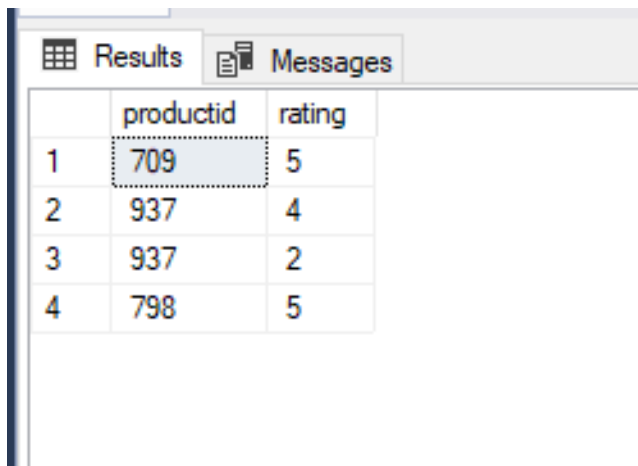|	CardType
1	SuperiorCard
2	Vista
3	Distinguish
4	ColonialVoice

Now from the output you can see that there are only 4 Card Types in this table and you do this using the **DISTINCT** keyword as shown in this scenario.

SCENARIO 5

Return values based on condition

Consider the table **[Production].[ProductReview]** and you can find the customer rating for each Product using this query:

SELECT productid,
   rating
FROM [Production].[productreview]

	productid	rating
1	709	5
2	937	4
3	937	2
4	798	5

The values in the "Rating" column are numbers, which are not meaningful in this case. To make the output more understandable (1-Poor, 2-Fair, 3-Good, 4-Very Good, 5- Excellent), you can use the **CASE** expression. The CASE expression will add the required logic to the "Rating" column.

SELECT productid,
   CASE rating
       WHEN 1 THEN 'Poor'
         WHEN 2 THEN 'Fair'
         WHEN 3 THEN 'Good'
         WHEN 4 THEN 'Very Good'
         WHEN 5 THEN 'Excellent'
       END AS rating
FROM [Production].[productreview]

Now you can see from the output that, if the value is 5 in the column "rating" then 'Excellent' is returned and if value is 2 then 'Fair' returned.

```sql
SELECT productid,
       CASE rating
         WHEN 1 THEN 'Poor'
         WHEN 2 THEN 'Fair'
         WHEN 3 THEN 'Good'
         WHEN 4 THEN 'Very Good'
         WHEN 5 THEN 'Excellent'
       END AS rating
FROM   [Production].[productreview]
```

10 %

Results | Messages

	productid	rating
1	709	Excellent
2	937	Very Good
3	937	Fair
4	798	Excellent

If you want to return values based on a condition, you can use the **CASE** expression as shown in this scenario.
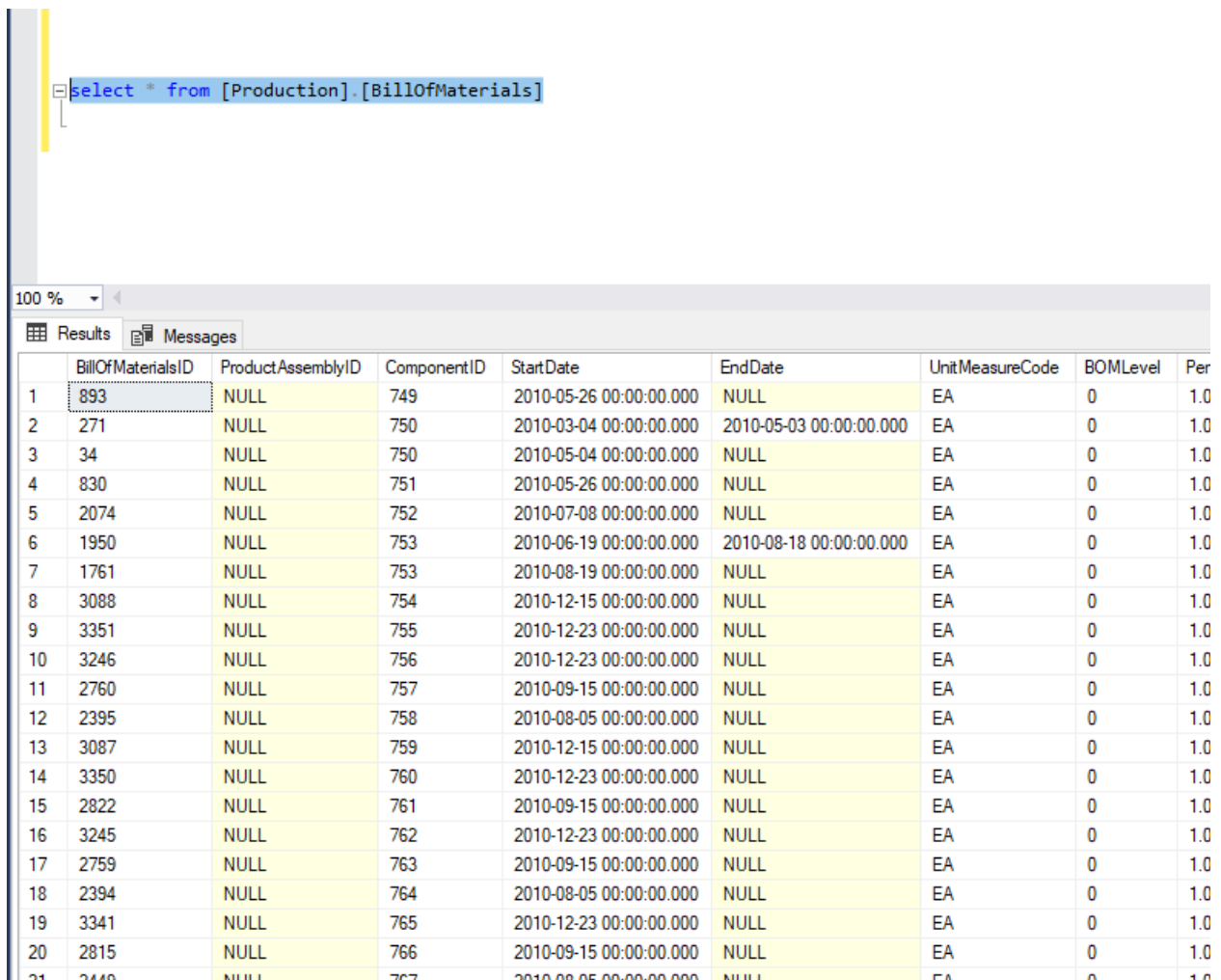
SCENARIO 6

Replace NULL values with specific values

Consider this table **[Production].[BillOfMaterials]**
You can see there are 2 columns (ProductAssemblyID, EndDate) with NULL Values.

select * from [Production].[BillOfMaterials]

The requirement is, in the output instead of NULL values need to have value 0 for
'ProductAssemblyID' column *without changing any values in the table*. To achieve this, you can
use **ISNULL()** function in the select to replace the NULL values with a specific value.

```sql
select
    BillOfMaterialsID,
        ISNULL(ProductAssemblyID,0) as ProductAssemblyID,
        EndDate
from [Production].[BillOfMaterials]
```

```sql
select
    BillOfMaterialsID,
    ISNULL(ProductAssemblyID,0) as ProductAssemblyID,
    EndDate
from [Production].[BillOfMaterials]
```

) %   ▼ ◄

⊞ Results   📄 Messages

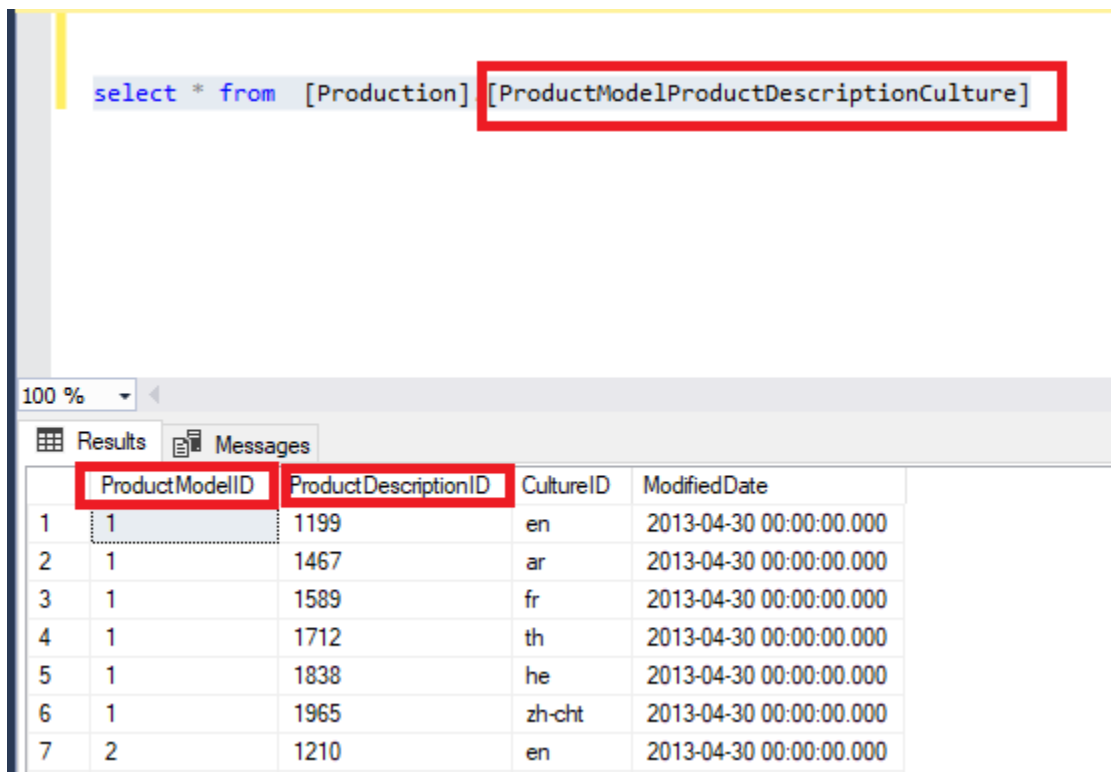|	BillOfMaterialsID	ProductAssemblyID	EndDate
	893	0	NULL
	271	0	2010-05-03 00:00:00.000
	34	0	NULL
	830	0	NULL
	2074	0	NULL
	1950	0	2010-08-18 00:00:00.000
	1761	0	NULL
	3088	0	NULL
	3351	0	NULL
0	3246	0	NULL
1	2760	0	NULL
2	2395	0	NULL
3	3087	0	NULL
4	3350	0	NULL
5	2822	0	NULL
6	3245	0	NULL
7	2759	0	NULL
8	2394	0	NULL
9	3341	0	NULL
0	2815	0	NULL
1	2449	0	NULL
2	2899	0	2010-11-14 00:00:00.000
3	2738	0	NULL
4	2363	0	NULL
5	1265	0	NULL
6	1195	0	NULL
7	3017	0	NULL
8	3281	0	NULL

In the output you can see that the ProductAssemblyID values with NULL got changed to 0.

SCENARIO 7

Replacing the table or column name temporarily

Consider the table **[Production].[ProductModelProductDescriptionCulture].**
Let's have a look at the columns and data in this table.

select * from [Production].[ProductModelProductDescriptionCulture]

You can see that the table name and column names are longer. It will be difficult to type the long names when we try to query between tables (which we will see later in **SQL Joins**). To name a column or table name with an Alias you will use the keyword **AS**.

select
    ProductModelID,
       ProductDescriptionID
from [Production].[ProductModelProductDescriptionCulture]

```
select
        ProductModelID,
        ProductDescriptionID
from  [Production].[ProductModelProductDescriptionCulture]
```

100 %   ▼ ◄

⊞ Results 📄 Messages

	ProductModelID	ProductDescriptionID
1	1	1199
2	1	1467
3	1	1589
4	1	1712
5	1	1838
6	1	1965
7	2	1210
8	2	1476
9	2	1598
10	2	1721

The above query itself can be re-written using the keyword AS and there will be no change in output.

select
    ProductModelID AS ID,
        ProductDescriptionID AS DescID
from [Production].[ProductModelProductDescriptionCulture] AS A

```
select
     ProductModelID AS ID,          ⬅
     ProductDescriptionID AS DescID    ⬅
from   [Production].[ProductModelProductDescriptionCulture] AS A    ⬅
```

100 % ▾ ◂

⊞ Results 🗐 Messages

	ID	DescID
1	1	1199
2	1	1467
3	1	1589
4	1	1712
5	1	1838
6	1	1965
7	2	1210

These alias names exists only during the duration of the query and will NOT change the names of the tables or columns permanently.

SCENARIO 8

Filtering out Information

Consider the table **[Person].[AddressType].**
You can see the table has the following columns and data.

select * from [Person].[AddressType]

	AddressTypeID	Name	rowguid	ModifiedDate
1	1	Billing	B84F78B1-4EFE-4A0E-8CB7-70E9F112F886	2008-04-30 00:00:00.000
2	2	Home	41BC2FF6-F0FC-475F-8EB9-CEC0805AA0F2	2008-04-30 00:00:00.000
3	3	Main Office	8EEEC28C-07A2-4FB9-AD0A-42D4A0BBC575	2008-04-30 00:00:00.000
4	4	Primary	24CB3088-4345-47C4-86C5-17B535133D1E	2008-04-30 00:00:00.000
5	5	Shipping	B29DA3F8-19A3-47DA-9DAA-15C84F4A83A5	2008-04-30 00:00:00.000
6	6	Archive	A67F238A-5BA2-444B-966C-0467ED9C427F	2008-04-30 00:00:00.000

Now you want to filter rows that has the name 'Archive' and list other rows in the output.
To filter out data, you use the operator **NOT**. You can re-write the above query as follows with the NOT operator:

select * from [Person].[AddressType] where NOT Name = 'Archive'

```
select * from [Person].[AddressType] where NOT Name = 'Archive'
```

100 %

Results | Messages

	AddressTypeID	Name	rowguid	ModifiedDate
1	1	Billing	B84F78B1-4EFE-4A0E-8CB7-70E9F112F886	2008-04-30 00:00:00.000
2	2	Home	41BC2FF6-F0FC-475F-8EB9-CEC0805AA0F2	2008-04-30 00:00:00.000
3	3	Main Office	8EEEC28C-07A2-4FB9-AD0A-42D4A0BBC575	2008-04-30 00:00:00.000
4	4	Primary	24CB3088-4345-47C4-86C5-17B535133D1E	2008-04-30 00:00:00.000
5	5	Shipping	B29DA3F8-19A3-47DA-9DAA-15C84F4A83A5	2008-04-30 00:00:00.000

You can see from the above output that the row with data 'Archive' got filtered in the result set.

SCENARIO 9

Filtering on more than 1 condition

The table [Purchasing].[PurchaseOrderDetail] has the below data.

select * from [Purchasing].[PurchaseOrderDetail]

In the above output, you can see that there is a column for ProductID and UnitPrice.
Now the requirement is to find all Purchase Orders for the ProductId = 512 that costs less than $35 Unit Price. In case, if you are curious to know what is the name of the Product that has the ProductID as 512, you refer the table **[Production].[Product].**

select * from [Production].[Product] where ProductID = 512

Now back to our requirement.
We have to filter data in the table **[Purchasing].[PurchaseOrderDetail]** based on two conditions, one is ProductID needs to be 512 and it should cost less than $35 Unit Price. Now you use the **AND** operator to filter data based on more than 1 condition.

select * from [Purchasing].[PurchaseOrderDetail]
where ProductID = 512 AND UnitPrice < 35

	PurchaseOrderID	PurchaseOrderDetailID	DueDate	OrderQty	ProductID	UnitPrice	LineTotal	ReceivedQty	RejectedQty	StockedQty
1	111	255	2012-03-22 00:00:00.000	550	512	34.9755	19236.525	550.00	0.00	550.00
2	190	463	2012-07-06 00:00:00.000	550	512	34.9755	19236.525	550.00	0.00	550.00
3	269	628	2012-10-31 00:00:00.000	550	512	34.9755	19236.525	550.00	0.00	550.00
4	348	797	2013-05-28 00:00:00.000	550	512	34.9755	19236.525	550.00	0.00	550.00
5	457	1011	2013-08-20 00:00:00.000	550	512	34.9755	19236.525	550.00	0.00	550.00
6	536	1203	2013-08-26 00:00:00.000	550	512	34.9755	19236.525	550.00	0.00	550.00
7	623	1395	2013-09-02 00:00:00.000	550	512	34.9755	19236.525	550.00	0.00	550.00
8	639	1424	2013-09-04 00:00:00.000	550	512	34.9755	19236.525	550.00	0.00	550.00
9	706	1582	2013-09-10 00:00:00.000	550	512	34.9755	19236.525	550.00	0.00	550.00
10	789	1764	2013-09-17 00:00:00.000	550	512	34.9755	19236.525	550.00	0.00	550.00
11	955	2135	2013-10-06 00:00:00.000	550	512	34.9755	19236.525	550.00	0.00	550.00
12	1118	2538	2013-11-22 00:00:00.000	550	512	34.9755	19236.525	550.00	0.00	550.00
13	1197	2708	2013-12-02 00:00:00.000	550	512	34.9755	19236.525	550.00	0.00	550.00
14	1276	2890	2013-12-11 00:00:00.000	550	512	34.9755	19236.525	550.00	0.00	550.00
15	1355	3077	2013-12-19 00:00:00.000	550	512	34.9755	19236.525	550.00	0.00	550.00
16	1513	3411	2014-01-06 00:00:00.000	550	512	34.9755	19236.525	550.00	0.00	550.00
17	1671	3744	2014-01-23 00:00:00.000	550	512	34.9755	19236.525	550.00	0.00	550.00
18	1750	3924	2014-02-03 00:00:00.000	550	512	34.9755	19236.525	550.00	0.00	550.00
19	1829	4099	2014-02-10 00:00:00.000	550	512	34.9755	19236.525	550.00	0.00	550.00
20	1908	4285	2014-02-20 00:00:00.000	550	512	34.9755	19236.525	550.00	0.00	550.00
21	1987	4450	2014-02-27 00:00:00.000	550	512	34.9755	19236.525	550.00	0.00	550.00

You can see that the data got filtered based on more than 1 condition in the above output.

Now there is a slight change in the requirement. You need to find all Purchase Orders for ProductId = 512 that has Unit Price greater than $35 but less than $40. Now to accommodate this change in requirement you can include the **OR** operator in the query. The query can be re-written as,

select * from [Purchasing].[PurchaseOrderDetail]
where ProductID = 512 AND (UnitPrice > 35 OR UnitPrice < 40)

Now in the output below you can see that all the Purchase Orders got listed for the ProductID 512 that has Unit Price greater than $35 but less than $40.
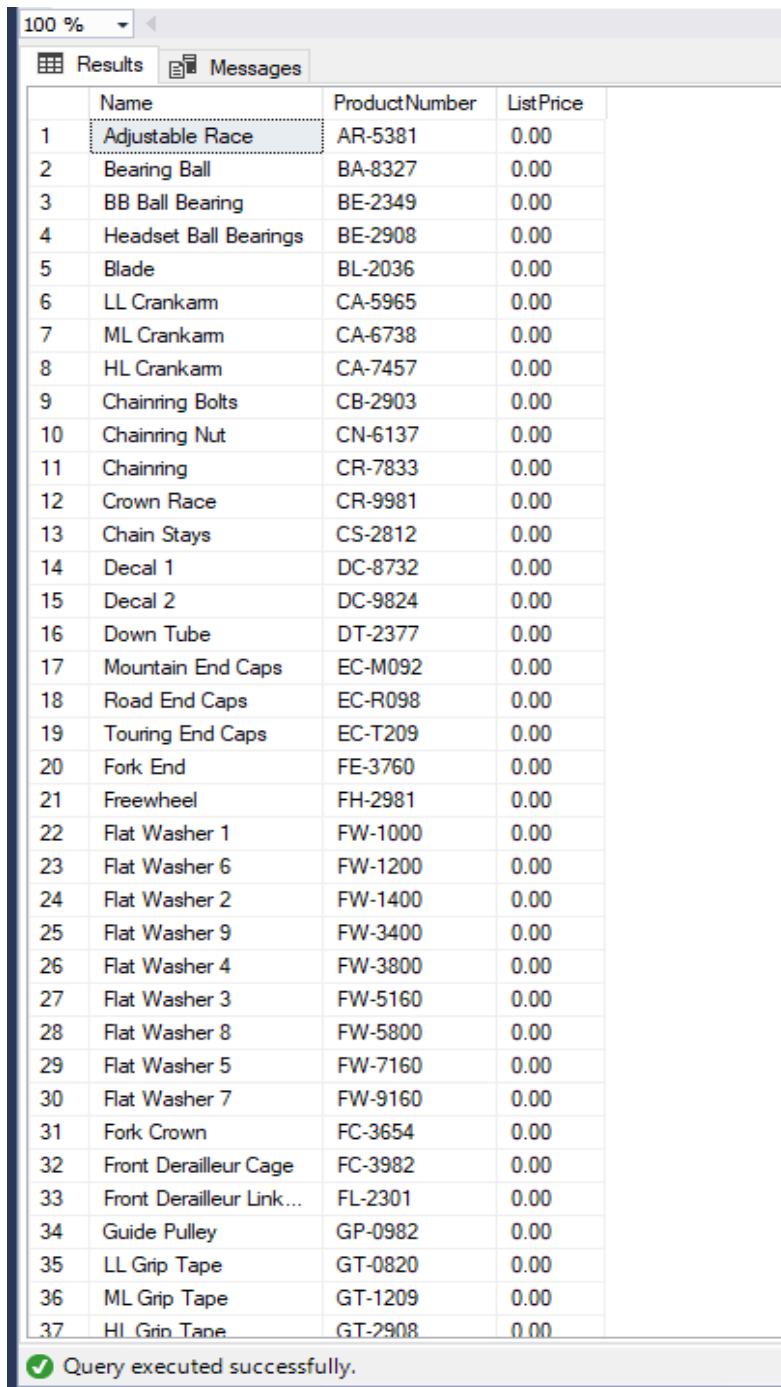
	PurchaseOrderID	PurchaseOrderDetailID	DueDate	OrderQty	ProductID	UnitPrice	LineTotal	ReceivedQty	R
1	5	6	2011-05-14 00:00:00.000	550	512	37.086	20397.30	550.00	0
2	84	188	2012-02-23 00:00:00.000	550	512	37.086	20397.30	550.00	5
3	111	255	2012-03-22 00:00:00.000	550	512	34.9755	19236.525	550.00	0
4	163	386	2012-06-13 00:00:00.000	550	512	37.086	20397.30	550.00	0
5	190	463	2012-07-06 00:00:00.000	550	512	34.9755	19236.525	550.00	0
6	242	570	2012-10-05 00:00:00.000	550	512	37.086	20397.30	550.00	8
7	269	628	2012-10-31 00:00:00.000	550	512	34.9755	19236.525	550.00	0
8	321	721	2013-05-09 00:00:00.000	550	512	37.086	20397.30	550.00	0
9	348	797	2013-05-28 00:00:00.000	550	512	34.9755	19236.525	550.00	0
10	400	892	2013-07-09 00:00:00.000	550	512	37.086	20397.30	550.00	0
11	430	948	2013-08-18 00:00:00.000	550	512	37.086	20397.30	550.00	0
12	457	1011	2013-08-20 00:00:00.000	550	512	34.9755	19236.525	550.00	0
13	509	1130	2013-08-25 00:00:00.000	550	512	37.086	20397.30	550.00	0
14	536	1203	2013-08-26 00:00:00.000	550	512	34.9755	19236.525	550.00	0
15	592	1317	2013-08-31 00:00:00.000	550	512	37.086	20397.30	550.00	8
16	623	1395	2013-09-02 00:00:00.000	550	512	34.9755	19236.525	550.00	0
17	639	1424	2013-09-04 00:00:00.000	550	512	34.9755	19236.525	550.00	0
18	679	1515	2013-09-08 00:00:00.000	550	512	37.086	20397.30	550.00	0
19	706	1582	2013-09-10 00:00:00.000	550	512	34.9755	19236.525	550.00	0
20	762	1687	2013-09-15 00:00:00.000	550	512	37.086	20397.30	550.00	8
21	789	1764	2013-09-17 00:00:00.000	550	512	34.9755	19236.525	550.00	0
22	845	1895	2013-09-22 00:00:00.000	550	512	37.086	20397.30	550.00	0
23	928	2084	2013-10-01 00:00:00.000	550	512	37.086	20397.30	550.00	0
24	955	2135	2013-10-06 00:00:00.000	550	512	34.9755	19236.525	550.00	0
25	1027	2313	2013-10-27 00:00:00.000	550	512	37.086	20397.30	550.00	0
26	1091	2461	2013-11-13 00:00:00.000	550	512	37.086	20397.30	550.00	0
27	1118	2538	2013-11-22 00:00:00.000	550	512	34.9755	19236.525	550.00	0
28	1170	2645	2013-11-27 00:00:00.000	550	512	37.086	20397.30	550.00	0

SCENARIO 10

Search within a range of values

If you look at the table **[Production].[Product],** it has the List Price of all the products.

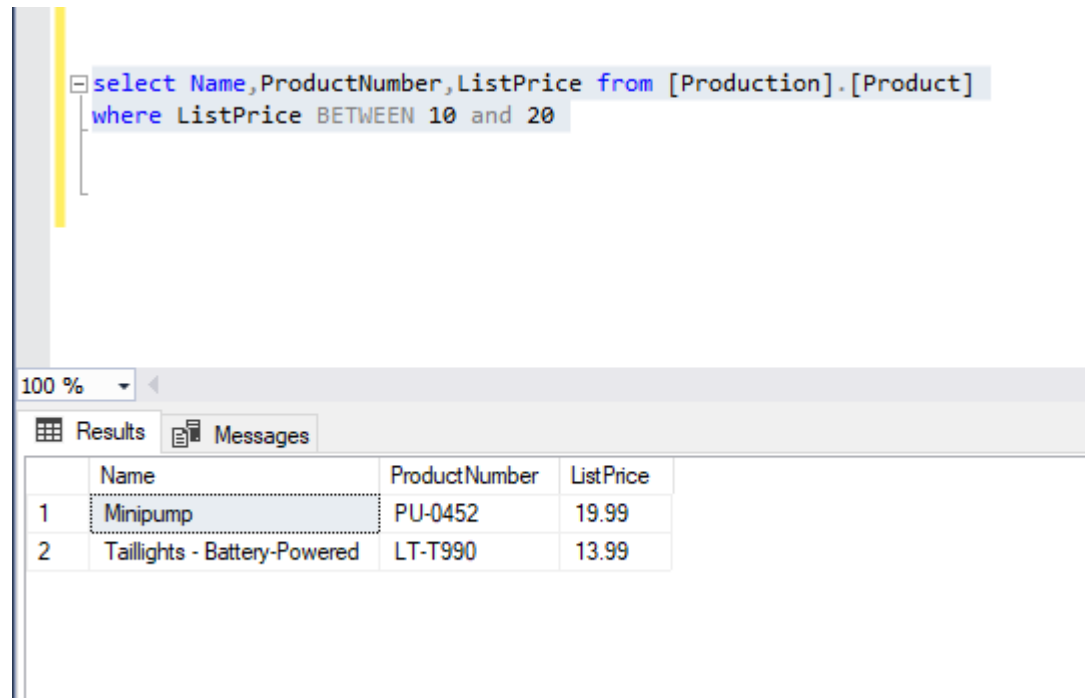select Name, ProductNumber, ListPrice from [Production].[Product]

	Name	ProductNumber	ListPrice
1	Adjustable Race	AR-5381	0.00
2	Bearing Ball	BA-8327	0.00
3	BB Ball Bearing	BE-2349	0.00
4	Headset Ball Bearings	BE-2908	0.00
5	Blade	BL-2036	0.00
6	LL Crankarm	CA-5965	0.00
7	ML Crankarm	CA-6738	0.00
8	HL Crankarm	CA-7457	0.00
9	Chainring Bolts	CB-2903	0.00
10	Chainring Nut	CN-6137	0.00
11	Chainring	CR-7833	0.00
12	Crown Race	CR-9981	0.00
13	Chain Stays	CS-2812	0.00
14	Decal 1	DC-8732	0.00
15	Decal 2	DC-9824	0.00
16	Down Tube	DT-2377	0.00
17	Mountain End Caps	EC-M092	0.00
18	Road End Caps	EC-R098	0.00
19	Touring End Caps	EC-T209	0.00
20	Fork End	FE-3760	0.00
21	Freewheel	FH-2981	0.00
22	Flat Washer 1	FW-1000	0.00
23	Flat Washer 6	FW-1200	0.00
24	Flat Washer 2	FW-1400	0.00
25	Flat Washer 9	FW-3400	0.00
26	Flat Washer 4	FW-3800	0.00
27	Flat Washer 3	FW-5160	0.00
28	Flat Washer 8	FW-5800	0.00
29	Flat Washer 5	FW-7160	0.00
30	Flat Washer 7	FW-9160	0.00
31	Fork Crown	FC-3654	0.00
32	Front Derailleur Cage	FC-3982	0.00
33	Front Derailleur Link...	FL-2301	0.00
34	Guide Pulley	GP-0982	0.00
35	LL Grip Tape	GT-0820	0.00
36	ML Grip Tape	GT-1209	0.00
37	HL Grip Tape	GT-2908	0.00

✓ Query executed successfully.

The requirement is to find the name of products that has a List Price in the range of $10-$20.
To search data within a range, you use the **BETWEEN** operator.
The above query can be re-written as,

select Name,ProductNumber,ListPrice from [Production].[Product]
where ListPrice BETWEEN 10 and 20

```
select Name,ProductNumber,ListPrice from [Production].[Product]
where ListPrice BETWEEN 10 and 20
```

100 %

Results | Messages

	Name	Product Number	List Price
1	Minipump	PU-0452	19.99
2	Taillights - Battery-Powered	LT-T990	13.99

From the above output, you can see that there are 2 products in the price range $10-$20.
The BETWEEN operator searches for records within the range of values specified.

SCENARIO 11

Filtering out data by comparing values

Consider the **[Production].[WorkOrder]** table. It has the below data:

select * from [Production].[WorkOrder]

Now you want to find the WorkOrderID's for Products with ProductID 995.
To find a particular value, you use the = **(EQUAL)** Operator.

select * from [Production].[WorkOrder] where ProductID = 995

And now you can see in the below output that WorkOrderID's for Products with ProductID 995 got listed.

If you are curious to know what is the name of the Product with the ProductID 995, you can refer the table [Production].[Product].

select * from [Production].[Product] where ProductID = 995

And now back to our requirement. You want to find out WorkOrderID's for ProductID 995 that has more than 500 Orders. To filter data based on more than condition, you can use the operator **> (GREATER THAN)**

select * from [Production].[WorkOrder]
where ProductID = 995 and OrderQty > 500

	WorkOrderID	ProductID	OrderQty	StockedQty	ScrappedQty	StartDate
1	16081	995	700	700	0	2012-06-02 00:00:00.0
2	17732	995	1026	1026	0	2012-07-03 00:00:00.0
3	19469	995	965	965	0	2012-08-03 00:00:00.0
4	21045	995	575	575	0	2012-09-02 00:00:00.0
5	22706	995	923	923	0	2012-10-03 00:00:00.0
6	24287	995	770	770	0	2012-11-02 00:00:00.0
7	27813	995	785	785	0	2013-01-03 00:00:00.0
8	29411	995	576	576	0	2013-01-31 00:00:00.0
9	31146	995	639	639	0	2013-03-03 00:00:00.0
10	32888	995	809	809	0	2013-04-02 00:00:00.0
11	34805	995	620	620	0	2013-05-03 00:00:00.0
12	36740	995	793	793	0	2013-06-02 00:00:00.0
13	39422	995	1219	1219	0	2013-07-03 00:00:00.0
14	42063	995	1203	1203	0	2013-08-03 00:00:00.0
15	44784	995	738	724	14	2013-09-02 00:00:00.0
16	47658	995	1053	1053	0	2013-10-03 00:00:00.0
17	50528	995	1051	1051	0	2013-11-02 00:00:00.0
18	53862	995	559	559	0	2013-12-03 00:00:00.0
19	56876	995	894	894	0	2014-01-03 00:00:00.0
20	59811	995	910	910	0	2014-02-01 00:00:00.0
21	62969	995	720	720	0	2014-03-04 00:00:00.0
22	66128	995	1075	1075	0	2014-04-03 00:00:00.0
23	69481	995	1021	1021	0	2014-05-04 00:00:00.0

From the above output, you can see the WorkOrderID's for Product ID 995 with more than 500 Orders.
Now again you need to find WorkOrderID's for Product ID 995 with more than 500 Orders that was received before May 3,2013. You can re-write the above query with < **(LESS THAN)** Operator to accommodate this change in requirement.

select * from [Production].[WorkOrder]
where ProductID = 995 and OrderQty > 500 and StartDate < '2013-05-03 00:00:00.000'

Results | Messages

	WorkOrderID	ProductID	OrderQty	StockedQty	ScrappedQty	StartDate	EndDate
1	16081	995	700	700	0	2012-06-02 00:00:00.000	2012-06-18 00
2	17732	995	1026	1026	0	2012-07-03 00:00:00.000	2012-07-19 00
3	19469	995	965	965	0	2012-08-03 00:00:00.000	2012-08-19 00
4	21045	995	575	575	0	2012-09-02 00:00:00.000	2012-09-18 00
5	22706	995	923	923	0	2012-10-03 00:00:00.000	2012-10-19 00
6	24287	995	770	770	0	2012-11-02 00:00:00.000	2012-11-18 00
7	27813	995	785	785	0	2013-01-03 00:00:00.000	2013-01-19 00
8	29411	995	576	576	0	2013-01-31 00:00:00.000	2013-02-16 00
9	31146	995	639	639	0	2013-03-03 00:00:00.000	2013-03-19 00
10	32888	995	809	809	0	2013-04-02 00:00:00.000	2013-04-18 00

In the above output you can see the WorkOrderID's for Product ID 995 with more than 500 Orders that was received before May 3,2013. And you have used all 3 comparison Operators (=, >, <) in the query.

SCENARIO 12

Finding rows based on a list of values

Consider the table [Production].[Product]. It has the below data:

select * from [Production].[Product]

	ProductID	Name	ProductNumber	MakeFlag	FinishedGoodsFlag	Color	SafetyStockLevel	ReorderPoint	StandardCost	ListPrice	Size
1	1	Adjustable Race	AR-5381	0	0	NULL	1000	750	0.00	0.00	NULL
2	2	Bearing Ball	BA-8327	0	0	NULL	1000	750	0.00	0.00	NULL
3	3	BB Ball Bearing	BE-2349	1	0	NULL	800	600	0.00	0.00	NULL
4	4	Headset Ball Bearings	BE-2908	0	0	NULL	800	600	0.00	0.00	NULL
5	316	Blade	BL-2036	1	0	NULL	800	600	0.00	0.00	NULL
6	317	LL Crankarm	CA-5965	0	0	Black	500	375	0.00	0.00	NULL
7	318	ML Crankarm	CA-6738	0	0	Black	500	375	0.00	0.00	NULL
8	319	HL Crankarm	CA-7457	0	0	Black	500	375	0.00	0.00	NULL
9	320	Chainring Bolts	CB-2903	0	0	Silver	1000	750	0.00	0.00	NULL
10	321	Chainring Nut	CN-6137	0	0	Silver	1000	750	0.00	0.00	NULL
11	322	Chainring	CR-7833	0	0	Black	1000	750	0.00	0.00	NULL
12	323	Crown Race	CR-9981	0	0	NULL	1000	750	0.00	0.00	NULL
13	324	Chain Stays	CS-2812	1	0	NULL	1000	750	0.00	0.00	NULL

Now the requirement is to find the name of Products that has
these 3 ListPrice values: **106.50, 1003.91, 333.42.**

To find out the name of Products whose list price is one of the following values: 106.50,
1003.91, 333.42, you use the **IN** operator.

select Name,ListPrice from [Production].[Product]
where ListPrice IN (106.50, 1003.91,333.42)

	Name	List Price
1	HL Touring Frame - Yellow, 60	1003.91
2	LL Touring Frame - Yellow, 62	333.42
3	HL Touring Frame - Yellow, 46	1003.91
4	HL Touring Frame - Yellow, 50	1003.91
5	HL Touring Frame - Yellow, 54	1003.91
6	HL Touring Frame - Blue, 46	1003.91
7	HL Touring Frame - Blue, 50	1003.91
8	HL Touring Frame - Blue, 54	1003.91
9	HL Touring Frame - Blue, 60	1003.91
10	LL Touring Frame - Blue, 50	333.42
11	LL Touring Frame - Blue, 54	333.42
12	LL Touring Frame - Blue, 58	333.42
13	LL Touring Frame - Blue, 62	333.42
14	LL Touring Frame - Yellow, 44	333.42
15	LL Touring Frame - Yellow, 50	333.42
16	LL Touring Frame - Yellow, 54	333.42
17	LL Touring Frame - Yellow, 58	333.42
18	LL Touring Frame - Blue, 44	333.42
19	Rear Brakes	106.50
20	Front Brakes	106.50

From the above output you can see that the name of the Products whose list price is one of the following values: 106.50, 1003.91, 333.42 got listed using the **IN** operator.

SCENARIO 13

Finding rows having a specific string
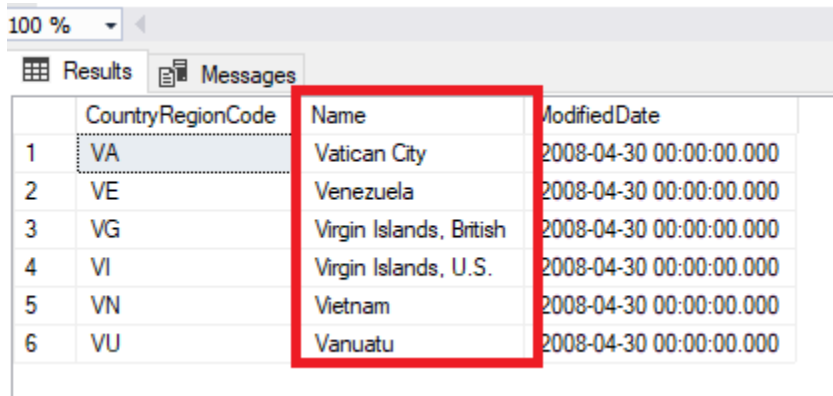
Consider the table **[Person].[CountryRegion].** It has the following data:

select * from [Person].[CountryRegion]

	CountryRegionCode	Name	ModifiedDate
1	AD	Andorra	2008-04-30 00:00:00.000
2	AE	United Arab Emirates	2008-04-30 00:00:00.000
3	AF	Afghanistan	2008-04-30 00:00:00.000
4	AG	Antigua and Barbuda	2008-04-30 00:00:00.000
5	AI	Anguilla	2008-04-30 00:00:00.000
6	AL	Albania	2008-04-30 00:00:00.000
7	AM	Armenia	2008-04-30 00:00:00.000
8	AN	Netherlands Antilles	2008-04-30 00:00:00.000
9	AO	Angola	2008-04-30 00:00:00.000
10	AQ	Antarctica	2008-04-30 00:00:00.000
11	AR	Argentina	2008-04-30 00:00:00.000
12	AS	American Samoa	2008-04-30 00:00:00.000
13	AT	Austria	2008-04-30 00:00:00.000
14	AU	Australia	2008-04-30 00:00:00.000
15	AW	Aruba	2008-04-30 00:00:00.000
16	AZ	Azerbaijan	2008-04-30 00:00:00.000
17	BA	Bosnia and Herzegovina	2008-04-30 00:00:00.000
18	BB	Barbados	2008-04-30 00:00:00.000
19	BD	Bangladesh	2008-04-30 00:00:00.000
20	BE	Belgium	2008-04-30 00:00:00.000
21	BF	Burkina Faso	2008-04-30 00:00:00.000
22	BG	Bulgaria	2008-04-30 00:00:00.000
23	BH	Bahrain	2008-04-30 00:00:00.000
24	BI	Burundi	2008-04-30 00:00:00.000

Now you got an issue in the application with the 2-letter Country Region Codes that starts with letter V , so you need to find the name of the countries that start with the letter V.
To filter data based on the matching character("V") , you use the **LIKE** operator.

select * from [Person].[CountryRegion] where Name like 'V%'

Using the **LIKE** Operator, you can find the Name of Countries that matches the character "V", as shown in the above output.

SCENARIO 14

Filtering rows having no data value in the column

Consider the table **[Production].[WorkOrder].** It has the following data:

select * from [Production].[WorkOrder]

	WorkOrderID	ProductID	OrderQty	StockedQty	ScrappedQty	StartDate	EndDate	DueDate	ScrapReasonID	ModifiedDate
I	1	722	8	8	0	2011-06-03 00:00:00.000	2011-06-13 00:00:00.000	2011-06-14 00:00:00.000	NULL	2011-06-13 00:00:00.000
2	2	725	15	15	0	2011-06-03 00:00:00.000	2011-06-13 00:00:00.000	2011-06-14 00:00:00.000	NULL	2011-06-13 00:00:00.000
3	3	726	9	9	0	2011-06-03 00:00:00.000	2011-06-13 00:00:00.000	2011-06-14 00:00:00.000	NULL	2011-06-13 00:00:00.000
4	4	729	16	16	0	2011-06-03 00:00:00.000	2011-06-13 00:00:00.000	2011-06-14 00:00:00.000	NULL	2011-06-13 00:00:00.000
5	5	730	14	14	0	2011-06-03 00:00:00.000	2011-06-13 00:00:00.000	2011-06-14 00:00:00.000	NULL	2011-06-13 00:00:00.000
6	6	732	16	16	0	2011-06-03 00:00:00.000	2011-06-13 00:00:00.000	2011-06-14 00:00:00.000	NULL	2011-06-13 00:00:00.000
7	7	733	4	4	0	2011-06-03 00:00:00.000	2011-06-13 00:00:00.000	2011-06-14 00:00:00.000	NULL	2011-06-13 00:00:00.000
8	8	738	19	19	0	2011-06-03 00:00:00.000	2011-06-13 00:00:00.000	2011-06-14 00:00:00.000	NULL	2011-06-13 00:00:00.000
9	9	741	2	2	0	2011-06-03 00:00:00.000	2011-06-13 00:00:00.000	2011-06-14 00:00:00.000	NULL	2011-06-13 00:00:00.000
10	10	742	3	3	0	2011-06-03 00:00:00.000	2011-06-13 00:00:00.000	2011-06-14 00:00:00.000	NULL	2011-06-13 00:00:00.000
11	11	743	1	1	0	2011-06-03 00:00:00.000	2011-06-13 00:00:00.000	2011-06-14 00:00:00.000	NULL	2011-06-13 00:00:00.000
12	12	745	1	1	0	2011-06-03 00:00:00.000	2011-06-13 00:00:00.000	2011-06-14 00:00:00.000	NULL	2011-06-13 00:00:00.000
13	13	747	4	4	0	2011-06-03 00:00:00.000	2011-06-19 00:00:00.000	2011-06-14 00:00:00.000	NULL	2011-06-19 00:00:00.000
14	14	748	2	2	0	2011-06-03 00:00:00.000	2011-06-19 00:00:00.000	2011-06-14 00:00:00.000	NULL	2011-06-19 00:00:00.000
15	15	749	4	4	0	2011-06-03 00:00:00.000	2011-06-19 00:00:00.000	2011-06-14 00:00:00.000	NULL	2011-06-19 00:00:00.000
16	16	753	14	14	0	2011-06-03 00:00:00.000	2011-06-19 00:00:00.000	2011-06-14 00:00:00.000	NULL	2011-06-19 00:00:00.000
17	17	754	27	27	0	2011-06-03 00:00:00.000	2011-06-19 00:00:00.000	2011-06-14 00:00:00.000	NULL	2011-06-19 00:00:00.000
18	18	755	11	11	0	2011-06-03 00:00:00.000	2011-06-19 00:00:00.000	2011-06-14 00:00:00.000	NULL	2011-06-19 00:00:00.000
19	19	756	14	14	0	2011-06-03 00:00:00.000	2011-06-19 00:00:00.000	2011-06-14 00:00:00.000	NULL	2011-06-19 00:00:00.000

You can see from the above output that the column "ScrapReasonID" has many NULL values. But you want to find Products that has a Scrap Reason. So first, you want to filter rows that has no value in the column "ScrapReasonID". You can use the **NOT NULL** operator to filter rows having NULL values.

select * from [Production].[WorkOrder] where ScrapReasonID IS NOT NULL

	WorkOrderID	ProductID	OrderQty	StockedQty	ScrappedQty	StartDate	EndDate	DueDate	ScrapReasonID	ModifiedDate
1	4	518	98	97	1	2011-06-03 00:00:00.000	2011-06-19 00:00:00.000	2011-06-14 00:00:00.000	7	2011-06-19 00:00:0(
2	69	810	120	117	3	2011-06-03 00:00:00.000	2011-06-19 00:00:00.000	2011-06-14 00:00:00.000	11	2011-06-19 00:00:0(
3	85	994	224	220	4	2011-06-03 00:00:00.000	2011-06-19 00:00:00.000	2011-06-14 00:00:00.000	14	2011-06-19 00:00:0(
4	91	328	240	236	4	2011-06-03 00:00:00.000	2011-06-13 00:00:00.000	2011-06-14 00:00:00.000	15	2011-06-13 00:00:0(
5	496	3	40	39	1	2011-06-14 00:00:00.000	2011-06-24 00:00:00.000	2011-06-25 00:00:00.000	16	2011-06-24 00:00:0(
6	1026	3	50	49	1	2011-06-28 00:00:00.000	2011-07-08 00:00:00.000	2011-07-09 00:00:00.000	4	2011-07-08 00:00:0(
7	1302	517	423	411	12	2011-07-04 00:00:00.000	2011-07-20 00:00:00.000	2011-07-15 00:00:00.000	1	2011-07-20 00:00:0(
8	1316	733	72	70	2	2011-07-04 00:00:00.000	2011-07-14 00:00:00.000	2011-07-15 00:00:00.000	3	2011-07-14 00:00:0(
9	1325	744	62	61	1	2011-07-04 00:00:00.000	2011-07-14 00:00:00.000	2011-07-15 00:00:00.000	4	2011-07-14 00:00:0(
10	1344	945	1132	1111	21	2011-07-04 00:00:00.000	2011-07-20 00:00:00.000	2011-07-15 00:00:00.000	7	2011-07-20 00:00:0(
11	1352	327	1431	1389	42	2011-07-04 00:00:00.000	2011-07-14 00:00:00.000	2011-07-15 00:00:00.000	9	2011-07-14 00:00:0(
12	1358	401	1418	1376	42	2011-07-04 00:00:00.000	2011-07-14 00:00:00.000	2011-07-15 00:00:00.000	10	2011-07-14 00:00:0(
13	1365	804	635	623	12	2011-07-04 00:00:00.000	2011-07-20 00:00:00.000	2011-07-15 00:00:00.000	11	2011-07-20 00:00:0(
14	1756	3	50	49	1	2011-07-14 00:00:00.000	2011-07-24 00:00:00.000	2011-07-25 00:00:00.000	9	2011-07-24 00:00:0(
15	1906	3	60	59	1	2011-07-18 00:00:00.000	2011-07-28 00:00:00.000	2011-07-29 00:00:00.000	1	2011-07-28 00:00:0(
16	2416	3	80	78	2	2011-08-01 00:00:00.000	2011-08-11 00:00:00.000	2011-08-12 00:00:00.000	3	2011-08-11 00:00:0(
17	2573	806	407	393	14	2011-08-04 00:00:00.000	2011-08-20 00:00:00.000	2011-08-15 00:00:00.000	12	2011-08-20 00:00:0(
18	2577	812	128	125	3	2011-08-04 00:00:00.000	2011-08-20 00:00:00.000	2011-08-15 00:00:00.000	13	2011-08-20 00:00:0(
19	2589	950	407	395	12	2011-08-04 00:00:00.000	2011-08-20 00:00:00.000	2011-08-15 00:00:00.000	15	2011-08-20 00:00:0(
20	2593	996	409	395	14	2011-08-04 00:00:00.000	2011-08-20 00:00:00.000	2011-08-15 00:00:00.000	15	2011-08-20 00:00:0(
21	2595	324	2080	2039	41	2011-08-04 00:00:00.000	2011-08-14 00:00:00.000	2011-08-15 00:00:00.000	16	2011-08-14 00:00:0(
22	2600	399	1040	1014	26	2011-08-04 00:00:00.000	2011-08-14 00:00:00.000	2011-08-15 00:00:00.000	1	2011-08-14 00:00:0(
23	2607	802	379	369	10	2011-08-04 00:00:00.000	2011-08-20 00:00:00.000	2011-08-15 00:00:00.000	2	2011-08-20 00:00:0(
24	2896	3	70	68	2	2011-08-12 00:00:00.000	2011-08-22 00:00:00.000	2011-08-23 00:00:00.000	16	2011-08-22 00:00:0(
25	3547	532	40	39	1	2011-08-30 00:00:00.000	2011-09-09 00:00:00.000	2011-09-10 00:00:00.000	8	2011-09-09 00:00:0(
26	3703	516	187	181	6	2011-09-03 00:00:00.000	2011-09-19 00:00:00.000	2011-09-14 00:00:00.000	1	2011-09-19 00:00:0(
27	3739	810	187	182	5	2011-09-03 00:00:00.000	2011-09-19 00:00:00.000	2011-09-14 00:00:00.000	6	2011-09-19 00:00:0(
28	3755	994	351	344	7	2011-09-03 00:00:00.000	2011-09-19 00:00:00.000	2011-09-14 00:00:00.000	9	2011-09-19 00:00:0(

Now you can see the Products that has a Scrap Reason.
If you want to dig further what is the name of the Product and the Scrap Reason you can check the tables: **[Production].[Product]** and **[Production].[ScrapReason]**

For example, If you want to see what is the name of the Product with ProductID 518 and Scrap Reason with ScrapReasonID 7, you can use the below queries:

```sql
select ProductID, Name from [Production].[Product] where ProductID = 518
```

```sql
select ScrapReasonID, Name from [Production].[ScrapReason] where ScrapReasonID = 7
```

	ProductID	Name
1	518	ML Road Seat Assembly

	ScrapReasonID	Name
1	7	Handling damage

In the above output, you can see the Product Name and Scrap Reason displayed.

SCENARIO 15

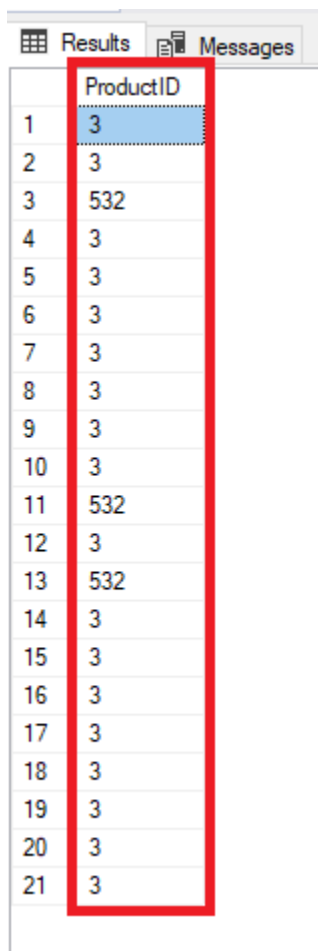## Filtering rows based on some values in a sub-query

Consider the table **[Production].[WorkOrder].** It has the following data:

`select * from [Production].[WorkOrder]`

	WorkOrderID	ProductID	OrderQty	StockedQty	ScrappedQty	StartDate	EndDate
1	1	722	8	8	0	2011-06-03 00:00:00.000	2011-06
2	2	725	15	15	0	2011-06-03 00:00:00.000	2011-06
3	3	726	9	9	0	2011-06-03 00:00:00.000	2011-06
4	4	729	16	16	0	2011-06-03 00:00:00.000	2011-06
5	5	730	14	14	0	2011-06-03 00:00:00.000	2011-06
6	6	732	16	16	0	2011-06-03 00:00:00.000	2011-06
7	7	733	4	4	0	2011-06-03 00:00:00.000	2011-06
8	8	738	19	19	0	2011-06-03 00:00:00.000	2011-06
9	9	741	2	2	0	2011-06-03 00:00:00.000	2011-06
10	10	742	3	3	0	2011-06-03 00:00:00.000	2011-06
11	11	743	1	1	0	2011-06-03 00:00:00.000	2011-06
12	12	745	1	1	0	2011-06-03 00:00:00.000	2011-06
13	13	747	4	4	0	2011-06-03 00:00:00.000	2011-06
14	14	748	2	2	0	2011-06-03 00:00:00.000	2011-06
15	15	749	4	4	0	2011-06-03 00:00:00.000	2011-06
16	16	753	14	14	0	2011-06-03 00:00:00.000	2011-06
17	17	754	27	27	0	2011-06-03 00:00:00.000	2011-06
18	18	755	11	11	0	2011-06-03 00:00:00.000	2011-06
19	19	756	14	14	0	2011-06-03 00:00:00.000	2011-06

Now you want to find the ProductID's having more than 20,000 Order Quantity.
You are interested only in the column "ProductID".
So, you re-write the above query as follows and you call this Query as **"Query 1"**

select ProductID from [Production].[WorkOrder]
where OrderQty > 20000

	ProductID
1	3
2	3
3	532
4	3
5	3
6	3
7	3
8	3
9	3
10	3
11	532
12	3
13	532
14	3
15	3
16	3
17	3
18	3
19	3
20	3
21	3

In the above output, you can see the ProductID's but now you want to find the Product names associated with these ProductID's. The names of products can be found in the table **[Production].[Product]** .You can do it using the below query and lets call it "**Query 2**".

select ProductID, Name from [Production].[Product]
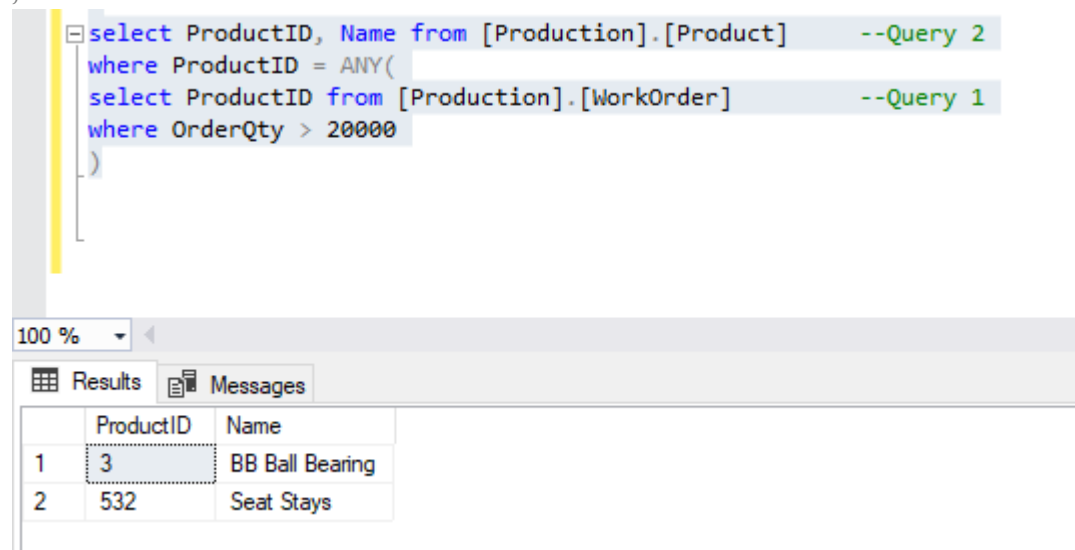
	ProductID	Name
1	1	Adjustable Race
2	879	All-Purpose Bike Stand
3	712	AWC Logo Cap
4	3	BB Ball Bearing
5	2	Bearing Ball
6	877	Bike Wash - Dissolver
7	316	Blade
8	843	Cable Lock
9	952	Chain
10	324	Chain Stays
11	322	Chainring
12	320	Chainring Bolts
13	321	Chainring Nut
14	866	Classic Vest, L
15	865	Classic Vest, M
16	864	Classic Vest, S
17	505	Cone-Shaped Race
18	323	Crown Race
19	504	Cup-Shaped Race
20	325	Decal 1
21	326	Decal 2
22	327	Down Tube
23	409	External Lock Washer 1
24	411	External Lock Washer 2
25	403	External Lock Washer 3
26	404	External Lock Washer 4

You can see that ALL Product ID and associated Product Names are returned.
But we are interested to find the name of Products ONLY for the Product ID's returned by the
below **"Query 1"** :

select ProductID from [Production].[WorkOrder]
where OrderQty > 20000

To find that you <u>need to pass the output from</u> "Query 1" to "Query 2".
This can be done using the **ANY** operator as follows:

select ProductID, Name from [Production].[Product]   --Query 2
where ProductID = ANY(
select ProductID from [Production].[WorkOrder]   --Query 1
where OrderQty > 20000
)

```
select ProductID, Name from [Production].[Product]      --Query 2
  where ProductID = ANY(
  select ProductID from [Production].[WorkOrder]        --Query 1
  where OrderQty > 20000
  )
```

100 %

Results   Messages

	ProductID	Name
1	3	BB Ball Bearing
2	532	Seat Stays

Since we pass the output of Query 1 to Query 2 to be compared by the **ANY** operator,
we call the Query 1 as the "**Sub-Query**" to Query 2.

SCENARIO 16

Return values by converting them into Upper or Lower case

Consider the table **[Production].[Product].** It has the following data:

select * from [Production].[Product]

	ProductID	Name	ProductNumber	MakeFlag	FinishedGoodsFlag	Color	SafetyStockLevel	ReorderPoint	Standard
1	1	Adjustable Race	AR-5381	0	0	NULL	1000	750	0.00
2	2	Bearing Ball	BA-8327	0	0	NULL	1000	750	0.00
3	3	BB Ball Bearing	BE-2349	1	0	NULL	800	600	0.00
4	4	Headset Ball Bearings	BE-2908	0	0	NULL	800	600	0.00
5	316	Blade	BL-2036	1	0	NULL	800	600	0.00
6	317	LL Crankarm	CA-5965	0	0	Black	500	375	0.00
7	318	ML Crankarm	CA-6738	0	0	Black	500	375	0.00
8	319	HL Crankarm	CA-7457	0	0	Black	500	375	0.00
9	320	Chainring Bolts	CB-2903	0	0	Silver	1000	750	0.00

In the above output, we are interested only in the 2 columns "Name" and "ProductNumber".
You can re-write the above query as follows,

select Name, ProductNumber from [Production].[Product]

You can see in the above output that the "ProductNumber" column has Characters in uppercase. So, you need to have the characters in "Name" column also in uppercase. To do that, you use the **UPPER()** function to convert all characters in "Name" column to uppercase.
You re-write the above query using the UPPER() function:

select UPPER(Name), ProductNumber from [Production].[Product]

Also, if you want to convert the characters in column "ProductNumber" to lower case, you can use the **LOWER()** function. And the query will look like this,

```sql
select Name, LOWER(ProductNumber) from [Production].[Product]
```


	Name	(No column name)
1	Adjustable Race	ar-5381
2	Bearing Ball	ba-8327
3	BB Ball Bearing	be-2349
4	Headset Ball Bearings	be-2908
5	Blade	bl-2036
6	LL Crankarm	ca-5965
7	ML Crankarm	ca-6738
8	HL Crankarm	ca-7457
9	Chainring Bolts	cb-2903
10	Chainring Nut	cn-6137
11	Chainring	cr-7833

You can see from the above queries, how characters can be converted to UPPER or LOWER cases using the SQL function UPPER() and LOWER().

SCENARIO 17

Return values by extracting specific characters

Consider the table [Production].[Product]. It has the following data:

```sql
select * from [Production].[Product]
```

We are interested in the 2 columns "Name" and "ProductNumber".

select Name, ProductNumber from [Production].[Product]

In the above output, you can see that the values in the column "ProductNumber" has 2 characters and 5 numbers. Now you want to get only the 2 characters from the "ProductNumber" for each "Name". To do that, you use the **LEFT()** function as shown below:

select Name, LEFT(ProductNumber,2) from [Production].[Product]

	Name	(No column name)
1	Adjustable Race	AR
2	Bearing Ball	BA
3	BB Ball Bearing	BE
4	Headset Ball Bearings	BE
5	Blade	BL
6	LL Crankarm	CA
7	ML Crankarm	CA
8	HL Crankarm	CA
9	Chainring Bolts	CB
10	Chainring Nut	CN

In case, if the requirement is to display only the last 4 digits from the column "ProductNumber", you use the **RIGHT()** function as shown below:

select Name, RIGHT(ProductNumber,4) from [Production].[Product]

	Name	(No column name)
1	Adjustable Race	5381
2	Bearing Ball	8327
3	BB Ball Bearing	2349
4	Headset Ball Bearings	2908
5	Blade	2036
6	LL Crankarm	5965
7	ML Crankarm	6738
8	HL Crankarm	7457
9	Chainring Bolts	2903
10	Chainring Nut	6137

You can see form the above output that there is no name for the column. This is because the column values are derived from the original column values. But if you want to have a name for the column, you can rename it using the keyword **AS**,

select Name, RIGHT(ProductNumber,4) AS ProductNumber from [Production].[Product]

	Name	Product Number
1	Adjustable Race	5381
2	Bearing Ball	8327
3	BB Ball Bearing	2349
4	Headset Ball Bearings	2908
5	Blade	2036
6	LL Crankarm	5965
7	ML Crankarm	6738
8	HL Crankarm	7457
9	Chainring Bolts	2903
10	Chainring Nut	6137
11	Chainring	7833
12	Crown Race	9981
13	Chain Stays	2812
14	Decal 1	8733

SCENARIO 18

Select records that has matching values in two tables

Consider the table **[Production].[WorkOrder].** It has the following data:

select * from [Production].[WorkOrder]

	WorkOrderID	ProductID	OrderQty	StockedQty	ScrappedQty	StartDate	EndDate
1	1	722	8	8	0	2011-06-03 00:00:00.000	2011-06-13
2	2	725	15	15	0	2011-06-03 00:00:00.000	2011-06-13
3	3	726	9	9	0	2011-06-03 00:00:00.000	2011-06-13
4	4	729	16	16	0	2011-06-03 00:00:00.000	2011-06-13
5	5	730	14	14	0	2011-06-03 00:00:00.000	2011-06-13

Let us consider the 2 columns "WorkOrderID" and "ProductID" in [Production].[WorkOrder].
You can re-write the above query as,

select WorkOrderID, ProductID from [Production].[WorkOrder]

	WorkOrderID	ProductID
1	88	3
2	129	3
3	170	3
4	201	3
5	241	3
6	274	3
7	312	3
8	350	3
9	391	3
10	427	3
11	459	3
12	496	3
13	526	3
14	559	3

Now the requirement is, you want to find the Product name of each ProductID <u>along with</u> the WorkOrderID.
You know that the name of the Product can be found from the table [Production].[Product] .

select ProductID, Name from [Production].[Product]

To find the Product names of "ProductID" <u>along with</u> the "WorkOrderID", you combine the two tables [Production].[WorkOrder] and [Production].[Product] using the keyword **INNER JOIN** that matches values for "ProductID" column in both the tables. You can combine the two tables using the keyword INNER JOIN as shown in below query,

select A.WorkOrderID, A.ProductID, B.Name from [Production].[WorkOrder] AS A
INNER JOIN [Production].[Product] AS B
ON A.ProductID = B.ProductID

Combining the two tables [Production].[WorkOrder] and [Production].[Product] using the keyword INNER JOIN based on the matching values in the _common_ column "ProductID",

you can get the Product names of "ProductID" along with the "WorkOrderID". In this way, you have selected records that has matching values (ProductID) in two tables [Production].[WorkOrder] and [Production].[Product]

SCENARIO 19

Select all records from first table and only the matching records from second table

Consider the table **[Production].[Product].** It has the following data:

select * from [Production].[Product]

	ProductID	Name	ProductNumber	MakeFlag	FinishedGoodsFlag	C
1	1	Adjustable Race	AR-5381	0	0	N
2	2	Bearing Ball	BA-8327	0	0	N
3	3	BB Ball Bearing	BE-2349	1	0	N
4	4	Headset Ball Bearings	BE-2908	0	0	N
5	316	Blade	BL-2036	1	0	N
6	317	LL Crankarm	CA-5965	0	0	B
7	318	ML Crankarm	CA-6738	0	0	B
8	319	HL Crankarm	CA-7457	0	0	B
9	320	Chainring Bolts	CB-2903	0	0	S

Let us consider only the first two columns "ProductID" and "Name".
The query can be re-written as,

select ProductID,Name from [Production].[Product]

		Results	Messages
		ProductID	Name
1	1	Adjustable Race
2	879	All-Purpose Bike Stand
3	712	AWC Logo Cap
4	3	BB Ball Bearing
5	2	Bearing Ball
6	877	Bike Wash - Dissolver
7	316	Blade
8	843	Cable Lock
9	952	Chain
10	324	Chain Stays
11	322	Chainring

Now the requirement is that, you want to find the Sales Orders for ALL ProductID's in the above output along with the ProductID and Name. Sales Orders can be found separately from another table [Sales].[SalesOrderDetail]

select ProductID, SalesOrderID from [Sales].[SalesOrderDetail]

	ProductID	SalesOrderID
1	707	43665
2	707	43668
3	707	43673
4	707	43677
5	707	43678
6	707	43680
7	707	43681
8	707	43683
9	707	43692
10	707	43693
11	707	43694
12	707	43849

But you want to find the Sales Orders for ALL ProductID's and Name's in the table [Production].[Product] in the same output. To do that you combine the two tables [Production].[Product] and [Sales].[SalesOrderDetail] using the keyword **LEFT JOIN** based on the common column between both tables. i.e. "ProductID" column. You can combine both the tables using LEFT JOIN as shown in below query:

select A.ProductID, A.Name, B.SalesOrderID from [Production].[Product] AS A
LEFT JOIN [Sales].[SalesOrderDetail] AS B
ON A.ProductID = B.ProductID

	ProductID	Name	SalesOrderID
204	531	Steerer	NULL
205	532	Seat Stays	NULL
206	533	Seat Tube	NULL
207	534	Top Tube	NULL
208	535	Tension Pulley	NULL
209	679	Rear Derailleur Cage	NULL
210	680	HL Road Frame - Black, 58	NULL
211	706	HL Road Frame - Red, 58	NULL
212	707	Sport-100 Helmet, Red	43665
213	707	Sport-100 Helmet, Red	43668
214	707	Sport-100 Helmet, Red	43673
215	707	Sport-100 Helmet, Red	43677
216	707	Sport-100 Helmet, Red	43678
217	707	Sport-100 Helmet, Red	43680
218	707	Sport-100 Helmet, Red	43681
219	707	Sport-100 Helmet, Red	43683
220	707	Sport-100 Helmet, Red	43692
221	707	Sport-100 Helmet, Red	43693
222	707	Sport-100 Helmet, Red	43694
223	707	Sport-100 Helmet, Red	43849
224	707	Sport-100 Helmet, Red	43851
225	707	Sport-100 Helmet, Red	43857
226	707	Sport-100 Helmet, Red	43861
227	707	Sport-100 Helmet, Red	43867
228	707	Sport-100 Helmet, Red	43871
229	707	Sport-100 Helmet, Red	43872
230	707	Sport-100 Helmet, Red	43873

In the above output, you can see that the Sales Orders are displayed for ALL "ProductID" and "Name" columns in [Production].[Product].In this way, you have selected all records from first table [Production].[Product] and only the matching records from second table [Sales]. [SalesOrderDetail].

SCENARIO 20

Select all records from second table and only the matching records from first table

Consider the products table [Production].[Product].It has the name of all products.

select ProductID, Name from [Production].[Product]

	ProductID	Name
1	1	Adjustable Race
2	879	All-Purpose Bike Stand
3	712	AWC Logo Cap
4	3	BB Ball Bearing
5	2	Bearing Ball
6	877	Bike Wash - Dissolver
7	316	Blade
8	843	Cable Lock
9	952	Chain
10	324	Chain Stays
11	322	Chainring
12	320	Chainring Bolts
13	321	Chainring Nut
14	866	Classic Vest, L
15	865	Classic Vest, M
16	864	Classic Vest, S
17	505	Cone Shaped Race

Now the requirement is to find the reviews of products <u>along with</u> the product name.
To find the customer review about a product you can check the table [Production].
[ProductReview]

select ProductID,Comments from [Production].[ProductReview]

	ProductID	Comments
1	709	I can't believe I'm singing the praises of a pair of socks, but I just came back from a grueling 3-day ride and these socks r...
2	937	A little on the heavy side, but overall the entry/exit is easy in all conditions. I've used these pedals for more than 3 years a...
3	937	Maybe it's just because I'm new to mountain biking, but I had a terrible time getting use to these pedals. In my first outing, ...
4	798	The Road-550-W from Adventure Works Cycles is everything it's advertised to be. Finally, a quality bike that is actually bu...

To find ONLY the name of products that has a Customer review, you combine the tables [Production].[Product] and [Production].[ProductReview] using the keyword **RIGHT JOIN** based on the common column between both tables. i.e. "ProductID" column. You can combine both the tables using RIGHT JOIN as shown in below query:

select B.ProductID, B.Comments, A.Name from [Production].[Product] AS A
RIGHT JOIN [Production].[ProductReview] AS B
ON B.ProductID = A.ProductID

	ProductID	Comments	Name
1	709	I can't believe I'm singing the praises of a pair of sock...	Mountain Bike Socks, M
2	937	A little on the heavy side, but overall the entry/exit is ...	HL Mountain Pedal
3	937	Maybe it's just because I'm new to mountain biking, b...	HL Mountain Pedal
4	798	The Road-550-W from Adventure Works Cycles is ev...	Road-550-W Yellow, 40

In the above output, you can see the reviews of products <u>along with</u> the product name. In this way, you have selected all records from second table [Production].[ProductReview] and only the matching records from first table [Production].[Product].

SCENARIO 21

Select all records from two tables when there is a match between them or not

Consider the Products table [Production].[Product]. It has the following data:

select * from [Production].[Product]

	ProductID	Name	ProductNumber	MakeFlag	FinishedGoodsFlag	Color	SafetyStockLevel	ReorderPoint
1	1	Adjustable Race	AR-5381	0	0	NULL	1000	750
2	2	Bearing Ball	BA-8327	0	0	NULL	1000	750
3	3	BB Ball Bearing	BE-2349	1	0	NULL	800	600
4	4	Headset Ball Bearings	BE-2908	0	0	NULL	800	600
5	316	Blade	BL-2036	1	0	NULL	800	600
6	317	LL Crankarm	CA-5965	0	0	Black	500	375

Let us consider only 3 columns from this table "ProductID", "Name", "ProductSubcategoryID". The above query can be re-written as,

select ProductID, Name, ProductSubcategoryID from [Production].[Product]

In the above output, you can see the Product names and the Sub-Category ID to which each Product belongs. But the requirement is, you want to find the Sub-Category name to which each Product belongs and also find if any sub-category name is not assigned to a Product name.

The Sub-category names alone can be found separately from the table [Production].[ProductSubcategory] using the below query,

select ProductSubcategoryID, Name from [Production].[ProductSubcategory]

Since you want to find the sub-category names for ALL products in table [Production].[Product] and product names associated with ALL sub-category names in table [Production]. [ProductSubcategory], you use the syntax **FULL JOIN** combining the two tables based on the common column between them i.e. "ProductSubcategoryID". The query to combine to two tables using FULL JOIN shown below,

```
select A.ProductID, A.Name, B.ProductSubcategoryID, B.Name
from [Production].[Product] AS A
FULL JOIN [Production].[ProductSubcategory] AS B
ON A.ProductSubcategoryID = B.ProductSubcategoryID
```

	ProductID	Name	ProductSubcategoryID	Name
203	530	Seat Post	NULL	NULL
204	531	Steerer	NULL	NULL
205	532	Seat Stays	NULL	NULL
206	533	Seat Tube	NULL	NULL
207	534	Top Tube	NULL	NULL
208	535	Tension Pulley	NULL	NULL
209	679	Rear Derailleur Cage	NULL	NULL
210	680	HL Road Frame - Black, 58	14	Road Frames
211	706	HL Road Frame - Red, 58	14	Road Frames
212	707	Sport-100 Helmet, Red	31	Helmets
213	708	Sport-100 Helmet, Black	31	Helmets
214	709	Mountain Bike Socks, M	23	Socks
215	710	Mountain Bike Socks, L	23	Socks
216	711	Sport-100 Helmet, Blue	31	Helmets
217	712	AWC Logo Cap	19	Caps
218	713	Long-Sleeve Logo Jersey, S	21	Jerseys
219	714	Long-Sleeve Logo Jersey, M	21	Jerseys
220	715	Long-Sleeve Logo Jersey, L	21	Jerseys
221	716	Long-Sleeve Logo Jersey, XL	21	Jerseys
222	717	HL Road Frame - Red, 62	14	Road Frames
223	718	HL Road Frame - Red, 44	14	Road Frames
224	719	HL Road Frame - Red, 48	14	Road Frames
225	720	HL Road Frame - Red, 52	14	Road Frames
226	721	HL Road Frame - Red, 56	14	Road Frames
227	722	LL Road Frame - Black, 58	14	Road Frames
228	723	LL Road Frame - Black, 60	14	Road Frames
229	724	LL Road Frame - Black, 62	14	Road Frames
230	725	LL Road Frame - Red, 44	14	Road Frames
221	726	LL Road Frame - Red, 48	14	Road Frames

In the above output you can see the sub-category names for ALL Products and there is no sub-category without a product name.

NOTE: The FULL JOIN keyword returns all matching records from both tables whether the other table matches or not based on the common column "ProductSubcategoryID".
So, if there are rows in "[Production].[ProductSubcategory]" that do not have matches in "[Production].[Product]", or if there are rows in "[Production].[Product]" that do not have matches in "[Production].[ProductSubcategory]", those rows will be listed as well.

SCENARIO 22

Return the number of items found in a result set

Consider the table [Production].[Product]. It has the following data:

select * from [Production].[Product]

Now the requirement is to check how many Product Numbers are there without any duplication. To check the count of Product numbers, you use the function **COUNT()** and to check the count without considering the duplicates, you use the keyword **DISTINCT**.

You use the below query to check the count of Product Numbers without any duplication,

select distinct COUNT(ProductNumber) from [Production].[Product]

The above output shows that there are 504 unique Product Numbers in the table [Production].[Product].

SCENARIO 23

Compute the total amount

Consider the table [Sales].[SalesOrderDetail]. It has the following data:

select * from [Sales].[SalesOrderDetail]

We are interested only in the 4 columns- SalesOrderID, ProductID, LineTotal, ModifiedDate. The above query can be re-written as follows,

select SalesOrderID, ProductID, LineTotal, ModifiedDate from [Sales].[SalesOrderDetail]
where ModifiedDate between '2011-01-01' and '2011-12-31'

	SalesOrderID	ProductID	Line Total	ModifiedDate
1	43659	776	2024.994000	2011-05-31 00:00:00.000
2	43659	777	6074.982000	2011-05-31 00:00:00.000
3	43659	778	2024.994000	2011-05-31 00:00:00.000
4	43659	771	2039.994000	2011-05-31 00:00:00.000
5	43659	772	2039.994000	2011-05-31 00:00:00.000
6	43659	773	4079.988000	2011-05-31 00:00:00.000
7	43659	774	2039.994000	2011-05-31 00:00:00.000
8	43659	714	86.521200	2011-05-31 00:00:00.000
9	43659	716	28.840400	2011-05-31 00:00:00.000
10	43659	709	34.200000	2011-05-31 00:00:00.000
11	43659	712	10.373000	2011-05-31 00:00:00.000
12	43659	711	80.746000	2011-05-31 00:00:00.000
13	43660	762	419.458900	2011-05-31 00:00:00.000
14	43660	758	874.794000	2011-05-31 00:00:00.000

The requirement is, you need to find the Total Revenue from the Product 777 sold in the year 2011.First, re-write the query to list the records for Product 777 sold in the year 2011,

select SalesOrderID, ProductID, LineTotal, ModifiedDate from [Sales].[SalesOrderDetail]
where ModifiedDate between '2011-01-01' and '2011-12-31'
and ProductID = 777

	SalesOrderID	ProductID	Line Total	ModifiedDate
4	43665	777	4049.988000	2011-05-31 00:00:00.000
5	43683	777	12149.964000	2011-05-31 00:00:00.000
6	43693	777	6074.982000	2011-05-31 00:00:00.000
7	43695	777	10124.970000	2011-05-31 00:00:00.000
8	43732	777	3374.990000	2011-06-08 00:00:00.000
9	43757	777	3374.990000	2011-06-14 00:00:00.000
10	43767	777	3374.990000	2011-06-17 00:00:00.000
11	43813	777	3374.990000	2011-06-26 00:00:00.000
12	43814	777	3374.990000	2011-06-26 00:00:00.000
13	43837	777	3374.990000	2011-06-29 00:00:00.000
14	43843	777	10124.970000	2011-07-01 00:00:00.000
15	43844	777	6074.982000	2011-07-01 00:00:00.000
16	43849	777	4049.988000	2011-07-01 00:00:00.000
17	43853	777	4049.988000	2011-07-01 00:00:00.000
18	43856	777	2024.994000	2011-07-01 00:00:00.000
19	43862	777	6074.982000	2011-07-01 00:00:00.000
20	43866	777	2024.994000	2011-07-01 00:00:00.000
21	43867	777	4049.988000	2011-07-01 00:00:00.000
22	43869	777	4049.988000	2011-07-01 00:00:00.000
23	43875	777	4049.988000	2011-07-01 00:00:00.000

In the above output, you can see that the records listed for Product 777 sold in the year 2011. Now to find the Total Revenue from the Product 777 in the year 2011, you use the **SUM()** function on the column "LineTotal". The query can be re-written as below:

```
select SUM(LineTotal) from [Sales].[SalesOrderDetail]
where ModifiedDate between '2011-01-01' and '2011-12-31'
and ProductID = 777
```


	(No column name)
1	800119.679268

From the above output, you can see the Total Revenue from the Product 777 in the year 2011

SCENARIO 24
Compute the average value

Consider the table [Sales].[SalesOrderDetail]. It has the following data:

select * from [Sales].[SalesOrderDetail]

	SalesOrderID	SalesOrderDetailID	CarrierTrackingNumber	OrderQty	ProductID	SpecialOfferI
1	43659	1	4911-403C-98	1	776	1
2	43659	2	4911-403C-98	3	777	1
3	43659	3	4911-403C-98	1	778	1
4	43659	4	4911-403C-98	1	771	1
5	43659	5	4911-403C-98	1	772	1
6	43659	6	4911-403C-98	2	773	1
7	43659	7	4911-403C-98	1	774	1
8	43659	8	4911-403C-98	3	714	1
9	43659	9	4911-403C-98	1	716	1
10	43659	10	4911-403C-98	6	709	1
11	43659	11	4911-403C-98	2	712	1

We are interested only in the 3 columns- ProductID, LineTotal , ModifiedDate.
The above query can be re-written as below,

select ProductID, LineTotal , ModifiedDate from [Sales].[SalesOrderDetail]

	ProductID	LineTotal	ModifiedDate
1	776	2024.994000	2011-05-31 00:00:00.000
2	777	6074.982000	2011-05-31 00:00:00.000
3	778	2024.994000	2011-05-31 00:00:00.000
4	771	2039.994000	2011-05-31 00:00:00.000
5	772	2039.994000	2011-05-31 00:00:00.000
6	773	4079.988000	2011-05-31 00:00:00.000
7	774	2039.994000	2011-05-31 00:00:00.000
8	714	86.521200	2011-05-31 00:00:00.000
9	716	28.840400	2011-05-31 00:00:00.000
10	709	34.200000	2011-05-31 00:00:00.000
11	712	10.373000	2011-05-31 00:00:00.000
12	711	80.746000	2011-05-31 00:00:00.000
13	762	419.458900	2011-05-31 00:00:00.000
14	758	874.794000	2011-05-31 00:00:00.000
15	745	809.760000	2011-05-31 00:00:00.000

The Requirement is, to find the average price on which the Product 777 got sold in 2011.
First, re-write the query to list the records for Product 777 sold in the year 2011,

select ProductID, LineTotal , ModifiedDate from [Sales].[SalesOrderDetail]
where ModifiedDate between '2011-01-01' and '2011-12-31'
and ProductID = 777

In the above output, you can see that the records listed for Product 777 sold in the year 2011. Now to find the average price on which the Product 777 got sold in 2011, you use the **AVG()** function on the column "LineTotal". The query can be re-written as below:

```
select AVG(LineTotal) from [Sales].[SalesOrderDetail]
where ModifiedDate between '2011-01-01' and '2011-12-31'
and ProductID = 777
```


From the above output, you can see the average price on which the Product 777 got sold in 2011.

SCENARIO 25

Compute the lowest value

Consider the table [Production].[ProductInventory]. It has the following data:

select * from [Production].[ProductInventory]

In the above output, you can see that there are several products(ProductID) having different quantities(Quantity) in Stock based on the LocationID. Now the requirement is, you need to find the lowest quantity in stock for the ProductID 944. You can re-write the above query using the **MIN()** function to find the lowest quantity in stock for the ProductID 944,

select MIN(Quantity) from [Production].[ProductInventory]
where ProductID = 944

You can see the lowest quantity in stock for the ProductID 944 from the above output.

SCENARIO 26
Compute the largest value

Consider the table [Production].[ProductInventory]. It has the following data:

select * from [Production].[ProductInventory]

	ProductID	LocationID	Shelf	Bin	Quantity	rowguid
1	1	1	A	1	408	47A242
2	1	6	B	5	324	D4544D
3	1	50	A	5	353	BFF7DC
4	2	1	A	2	427	F407C0
5	2	6	B	1	318	CA1FF2
6	2	50	A	6	364	D38CFE
7	3	1	A	7	585	E18A51
8	3	6	B	9	443	3C860C

In the above output, you can see that there are several ProductID's having different quantities(Quantity) in Stock based on the LocationID. Now the requirement is, you need to find the largest quantity in stock for the ProductID 747. You can re-write the above query using the **MAX()** function to find the largest quantity in stock for the ProductID 747,

select MAX(Quantity) from [Production].[ProductInventory]
where ProductID = 747

	(No column name)
1	161

You can see the largest quantity in stock for the ProductID 747 from the above output.

SCENARIO 27

Combine values from two columns into one column

Consider the table [Person].[StateProvince]. It has the Following data:

select * from [Person].[StateProvince]

	StateProvinceID	StateProvinceCode	CountryRegionCode	IsOnlyStateProvinceFlag	Name
1	1	AB	CA	0	Alberta
2	2	AK	US	0	Alaska
3	3	AL	US	0	Alabama
4	4	AR	US	0	Arkansas
5	5	AS	AS	1	American Samoa
6	6	AZ	US	0	Arizona
7	7	BC	CA	0	British Columbia
8	8	BY	DE	0	Bayern
9	9	CA	US	0	California
10	10	CO	US	0	Colorado
11	11	CT	US	0	Connecticut
12	12	DC	US	0	District of Columbia
13	13	DE	US	0	Delaware
14	14	ENG	GB	1	England

We are interested only in 3 columns- StateProvinceID, StateProvinceCode, Name.
The above query can be re-written as below,

select StateProvinceID,StateProvinceCode,Name from [Person].[StateProvince]

Now the requirement is, display State code and State name in this format:

State Code-State Name

For example, need to display as "**AK-Alaska**", for State Code-AK and State Name- Alaska. You use the **CONCAT()** function to combine the 2 columns- StateProvinceCode & Name as single column. The above query can be re-written as follows using the CONCAT() function,

select StateProvinceID, CONCAT(StateProvinceCode,'-',Name) from [Person].[StateProvince]

	StateProvinceID	(No column name)
1	1	AB -Alberta
2	2	AK -Alaska
3	3	AL -Alabama
4	4	AR -Arkansas
5	5	AS -American Samoa
6	6	AZ -Arizona
7	7	BC -British Columbia
8	8	BY -Bayern
9	9	CA -California
10	10	CO -Colorado
11	11	CT -Connecticut
12	12	DC -District of Columbia
13	13	DE -Delaware
14	14	ENG-England
15	15	FL -Florida
16	16	FM -Micronesia
17	17	GA -Georgia

In the above output, you can see that the 2 columns-- StateProvinceCode & Name got combined as a single column in the output. But you can notice that there is no column name for the new column. If you want to give a name to the newly created column, use the **AS** keyword,

select StateProvinceID, CONCAT(StateProvinceCode,'-',Name) AS State from [Person].[StateProvince]

	StateProvinceID	State
1	1	AB -Alberta
2	2	AK -Alaska
3	3	AL -Alabama
4	4	AR -Arkansas
5	5	AS -American Samoa
6	6	AZ -Arizona
7	7	BC -British Columbia
8	8	BY -Bayern
9	9	CA -California
10	10	CO -Colorado
11	11	CT -Connecticut
12	12	DC -District of Columbia
13	13	DE -Delaware
14	14	ENG-England
15	15	FL -Florida

Now you can see from the output that the newly created column got the name as "State".

SCENARIO 28

Create a Calculated Field

Consider the table [Purchasing].[PurchaseOrderDetail]. It has the following data:

	PurchaseOrderID	PurchaseOrderDetailID	DueDate	OrderQty	ProductID	UnitPrice	LineTotal	ReceivedQty	RejectedQty
1	1	1	2011-04-30 00:00:00.000	4	1	50.26	201.04	3.00	0.00
2	2	2	2011-04-30 00:00:00.000	3	359	45.12	135.36	3.00	0.00
3	2	3	2011-04-30 00:00:00.000	3	360	45.5805	136.7415	3.00	0.00
4	3	4	2011-04-30 00:00:00.000	550	530	16.086	8847.30	550.00	0.00
5	4	5	2011-04-30 00:00:00.000	3	4	57.0255	171.0765	2.00	1.00
6	5	6	2011-05-14 00:00:00.000	550	512	37.086	20397.30	550.00	0.00
7	6	7	2011-05-14 00:00:00.000	550	513	26.5965	14628.075	468.00	0.00
8	7	8	2011-05-14 00:00:00.000	550	317	27.0585	14882.175	550.00	0.00
9	7	9	2011-05-14 00:00:00.000	550	318	33.579	18468.45	550.00	0.00
10	7	10	2011-05-14 00:00:00.000	550	319	46.0635	25334.925	550.00	0.00
11	8	11	2011-05-14 00:00:00.000	3	403	47.4705	142.4115	3.00	0.00
12	8	12	2011-05-14 00:00:00.000	3	404	45.3705	136.1115	3.00	0.00
13	8	13	2011-05-14 00:00:00.000	3	405	49.644	148.932	3.00	0.00
14	8	14	2011-05-14 00:00:00.000	3	406	45.3705	136.1115	3.00	0.00
15	8	15	2011-05-14 00:00:00.000	3	407	43.2705	129.8115	3.00	0.00
16	9	16	2011-12-28 00:00:00.000	3	422	47.523	142.569	3.00	0.00
17	9	17	2011-12-28 00:00:00.000	3	423	45.423	136.269	3.00	0.00
18	9	18	2011-12-28 00:00:00.000	3	424	49.6965	149.0895	3.00	0.00

We are interested only in the 5 columns-ProductID, UnitPrice, OrderQty, LineTotal, RejectedQty. The above query can re-written now as follows,

select ProductID,UnitPrice,OrderQty,LineTotal,RejectedQty from [Purchasing].[PurchaseOrderDetail]

⊞ Results ▨ Messages

	ProductID	UnitPrice	OrderQty	LineTotal	RejectedQty
1	1	50.26	4	201.04	0.00
2	359	45.12	3	135.36	0.00
3	360	45.5805	3	136.7415	0.00
4	530	16.086	550	8847.30	0.00
5	4	57.0255	3	171.0765	1.00
6	512	37.086	550	20397.30	0.00
7	513	26.5965	550	14628.075	0.00
8	317	27.0585	550	14882.175	0.00
9	318	33.579	550	18468.45	0.00
10	319	46.0635	550	25334.925	0.00
11	403	47.4705	3	142.4115	0.00
12	404	45.3705	3	136.1115	0.00
13	405	49.644	3	148.932	0.00
14	406	45.3705	3	136.1115	0.00
15	407	43.2705	3	129.8115	0.00
16	422	47.523	3	142.569	0.00

From the above output, you can see the Total Price(LineTotal) calculated based on the Price per Unit(UnitPrice) and the Orders received(OrderQty). **The requirement is, you need to find the Amount lost due to the rejected quantity (RejectedQty).** You have the Price per Unit(UnitPrice) and the quantities Rejected(RejectedQty). To find the Amount lost due to the rejected quantities, you need to multiply UnitPrice * RejectedQty and create them as a calculated field. This can be done using the below query,

select ProductID,UnitPrice,OrderQty,LineTotal,RejectedQty,
    (UnitPrice*RejectedQty) as LossAmount
from [Purchasing].[PurchaseOrderDetail]

	ProductID	UnitPrice	OrderQty	LineTotal	RejectedQty	LossAmount	
16	422	47.523	3	142.569	0.00	0.000000	
17	423	45.423	3	136.269	0.00	0.000000	
18	424	49.6965	3	149.0895	0.00	0.000000	
19	425	45.423	3	136.269	0.00	0.000000	
20	426	43.323	3	129.969	0.00	0.000000	
21	320	47.4705	3	142.4115	0.00	0.000000	
22	321	42.798	3	128.394	0.00	0.000000	
23	322	25.4205	60	1525.23	0.00	0.000000	
24	438	41.223	3	123.669	0.00	0.000000	
25	439	45.4965	3	136.4895	0.00	0.000000	
26	440	41.223	3	123.669	0.00	0.000000	
27	441	39.123	3	117.369	0.00	0.000000	
28	941	62.9895	550	34644.225	82.00	5165.139000	⬅
29	320	45.3705	3	136.1115	0.00	0.000000	
30	321	40.488	3	121.464	0.00	0.000000	
31	322	26.3655	60	1581.93	0.00	0.000000	
32	504	48.762	3	146.286	3.00	146.286000	⬅
33	497	34.188	3	102.564	0.00	0.000000	
34	323	50.2635	3	150.7905	0.00	0.000000	

You can see from the above output that the Amount lost(LossAmount) due to rejected quantity(RejectedQty), got created as a calculated field.
For example, 82 rejected quantities each with a unit price of $62.9895, lead to a loss of $5165.139.

SCENARIO 29

Arrange rows in groups

Consider the table [Production].[ProductInventory]. It has the following data:

	ProductID	LocationID	Shelf	Bin	Quantity	rowguid
1	1	1	A	1	408	47A24246-6C43-48EB-968F-025738A8
2	1	6	B	5	324	D4544D7D-CAF5-46B3-AB22-5718DC(
3	1	50	A	5	353	BFF7DC60-96A8-43CA-81A7-D6D2ED
4	2	1	A	2	427	F407C07A-CA14-4684-A02C-608BD00
5	2	6	B	1	318	CA1FF2F4-48FB-4960-8D92-3940B633
6	2	50	A	6	364	D38CFBEE-6347-47B1-B033-0E278C(
7	3	1	A	7	585	E18A519B-FB5E-4051-874C-58CD584
8	3	6	B	9	443	3C860C96-15FF-4DF4-91D7-B237FF6
9	3	50	A	10	324	1339E5E3-1F8E-4B82-A447-A8666A2(
10	4	1	A	6	512	6BEAF0A0-971A-4CE1-96FE-692807D
11	4	6	B	10	422	2C82427A-63F1-4877-A1F6-A27B4D2(
12	4	50	A	11	388	FD912E69-EFA2-4AB7-82A4-03F5101.
13	316	5	A	11	532	1EE3DBD3-2A7E-47DC-AF99-1B5855
14	316	10	B	1	388	CB2A24D7-9B70-4140-8836-9EB7592
15	316	50	B	8	441	36B375A3-022A-45BF-B425-DBFFAA(
16	317	1	C	1	283	C04FC1CF-1D2B-4480-BA13-64C6EF7
17	317	5	A	1	158	83332A73-48A9-401D-95F4-385C944[
18	317	50	A	21	152	4072C90C-A867-4F64-882F-EC45ADA
19	318	1	C	2	136	F287EFD3-CCC5-4344-9F4A-E588BBF
20	318	5	A	2	171	B62222F8 90B5 4DA1 BEE1 453AA19

In the above output, you can see the ProductID's having different "Quantity" based on the "LocationID".

Now the requirement is to find the lowest Quantity for each ProductID in the Inventory.

For example, the lowest quantity for ProductID- 1 is 324.

Similarly, the lowest quantity for ProductID-2 is 318.

Likewise, find the lowest Quantity for each ProductID in the Inventory.

To find the lowest quantity for each ProductID, you need to *group* the "ProductID" together using the **GROUP BY** clause.The above query can be re-written as follows,

```
select ProductID, MIN(Quantity) from [Production].[ProductInventory]
GROUP BY ProductID
```

	ProductID	(No column name)
1	1	324
2	2	318
3	3	324
4	4	388
5	316	388
6	317	152
7	318	132
8	319	184
9	320	283
10	321	540
11	322	475
12	323	513
13	324	476
14	325	540
15	326	475
16	327	408

Now you can see in the output, the lowest quantity for each ProductID.

SCENARIO 30

Filter Groups based on condition

Consider the table [Production].[ProductInventory]. It has the following data:

select * from [Production].[ProductInventory]

Similar to previous scenario, in the above output, you can see the ProductID's having different "Quantity" based on the "LocationID".

Now the requirement is, if the location count (LocationID) is less than 3 for a ProductID, then need to find the lowest stock quantity (Quantity) for only that ProductID.

For example, consider ProductID-1, it has 3 locations(LocationID) and you are not required to find the lowest quantity in Stock for the ProductID-1.

Consider ProductID-507, it has only 2 locations and you need to find the lowest quantity in stock for the ProductID-507.Likewise, you need to find the lowest stock quantity (Quantity) for ProductID's that has less than 3 locations (LocationID).

```sql
select * from [Production].[ProductInventory] where ProductID = 507
```

The above output shows that the ProductID-507 has only 2 locations.

Now back to the requirement, to find the lowest stock quantity(Quantity) for ProductID's that has less than 3 locations(LocationID), first you need to group the ProductID's using the **GROUP BY** clause and then filter the GROUP using the **HAVING** clause. This can be done using the below query,

```sql
select ProductID,
    MIN(Quantity) as LowestQuantity,
    count(locationID) as Locations from [Production].[ProductInventory]
GROUP BY ProductID
HAVING count(locationID) < 3
```