(T4)討論Select搭配Aggregation和GroupBy。Count、Sum、Avg、Min、Max  
CourseGUID: e48417fc-9db5-4e99-822c-706c5ccef6cc  
=======================================================================  
(T4)討論Select搭配Aggregation和GroupBy。Count、Sum、Avg、Min、Max  
=======================================================================  
0. What to learn

-----------

1. Northwind Database

1.1. 1st way : Restore database from bak file

1.2. 2nd way : Restore database from sql file

1.3. Northwind Database Diagram

-----------

2. Select

3. Group By

4. Where V.S. Having  
=======================================================================

0. What to learn

1.

Select (DISTINCT/TOP n)..

WHERE...(AND/OR)...

ORDER BY ...(ASC/DESC)

2.

SQL Operator

Reference:

<https://www.w3schools.com/sql/sql_operators.asp>

<https://www.tutorialspoint.com/sql/sql-wildcards.htm>

<https://docs.microsoft.com/en-us/sql/t-sql/language-elements/wildcard-character-s-to-match-transact-sql>

<https://docs.microsoft.com/en-us/sql/t-sql/language-elements/wildcard-character-s-not-to-match-transact-sql>

2.1.

SQL Comparison Operators

2.1.1. =    : Equal to

2.1.2. >     : Greater than

2.1.3. <     : Less than

2.1.4. >=    : Greater than or equal to

2.1.5. <=    : Less than or equal to

2.1.6. != or <>    : Not equal to

2.2.

SQL Logical Operators

2.2.1. ALL     : TRUE if all of the subquery values meet the condition

2.2.2. SOME    : TRUE if any of the subquery values meet the condition

2.2.3. ANY     : TRUE if any of the subquery values meet the condition

2.2.4. AND     : TRUE if all the conditions separated by AND is TRUE

2.2.5. OR      : TRUE if any of the conditions separated by OR is TRUE

2.2.6. BETWEEN : TRUE if the operand is within the range of comparisons.

2.2.7. IN      : TRUE if the operand is equal to one of a list of expressions

2.2.8. EXISTS  : TRUE if the subquery returns one or more records

2.2.9. LIKE    : TRUE if the operand matches a pattern

2.2.10. NOT    : Displays a record if the condition(s) is NOT TRUE

2.3.

SQL Wildcard Operators

2.3.1. %    : The percent sign means matches zero or more characters.

2.3.2. \_    : The underscore (\_) means matches one character

2.3.3. []   : Matches any single character within the specified range between brackets []

2.3.4. [^]  : Matches any single character that is NOT within the range between brackets []

3.

- In order to use aggregate, we need Group By

- aggregate include Count(), Sum(), avg(), Min(), Max().

4.

WHERE V.S. Having

4.1.

WHERE and HAVING can be used together.

Aggregates includes Count, Sum, Avg, Min and Max.

Aggregates must be happened after Group By,

because Aggregates need Group By to perform.

4.2.

WHERE operator filters the rows before Group by.

Thus, WHERE operator CAN NOT be used to filter Aggregates.

HAVING operator filters the Group after Group by.

Thus, HAVING operator CAN filter Aggregates.

4.3.

HAVING is slower than WHERE.

Try to use WHERE to replace HAVING if possible.

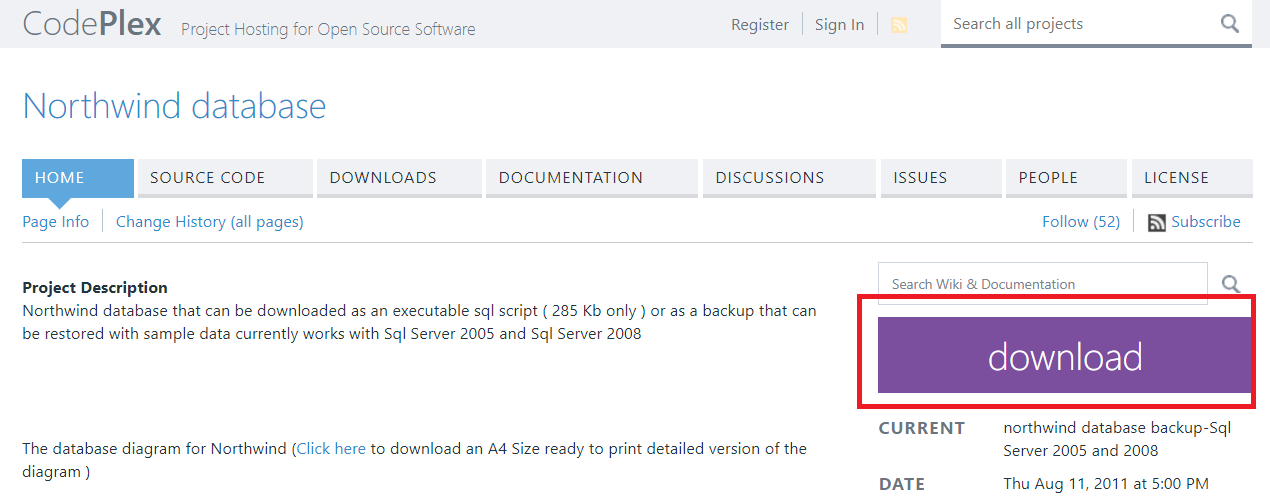
1. Northwind Database

There are 2 ways to re-store database.

1.1. 1st way : Restore database from bak file

Go the the following link and download the Northwind Database

<https://northwinddatabase.codeplex.com/>



Press the download button and get the file "Northwind.bak.zip"

Unzip it and get the file "Northwind.bak"

If you don't have any software to unzip the file,

you may use 7-zip (<http://www.7-zip.org/download.html>)

It is a very nice and free software.

In SQL server

--> Database --> Restore Database...

--> Device --> ... -->

-->Add --> paste the bak file path.

E.g.

D:\0\_MyDocument\Desktop\Northwind.bak\Northwind.bak

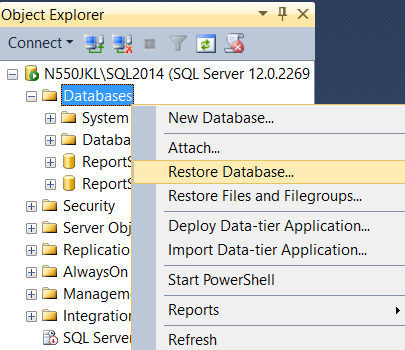
**-**-> OK

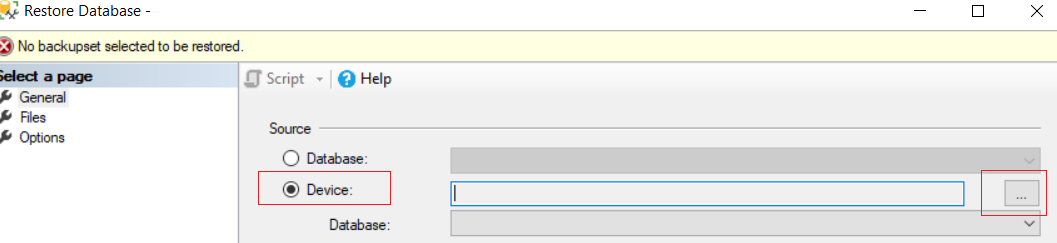
--> Verify Backup Media

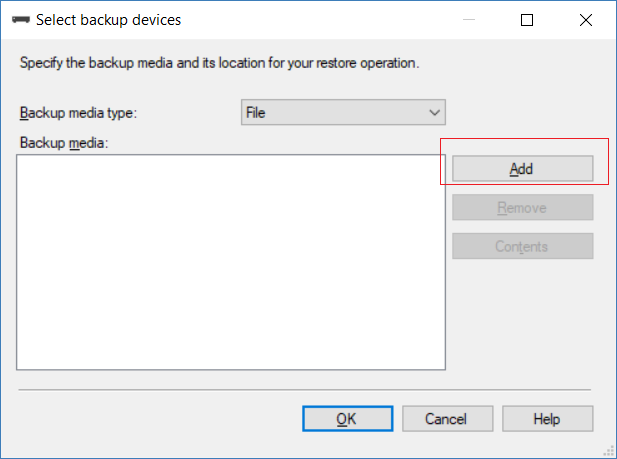
--> Files Tab --> Make sure the path is what you want.

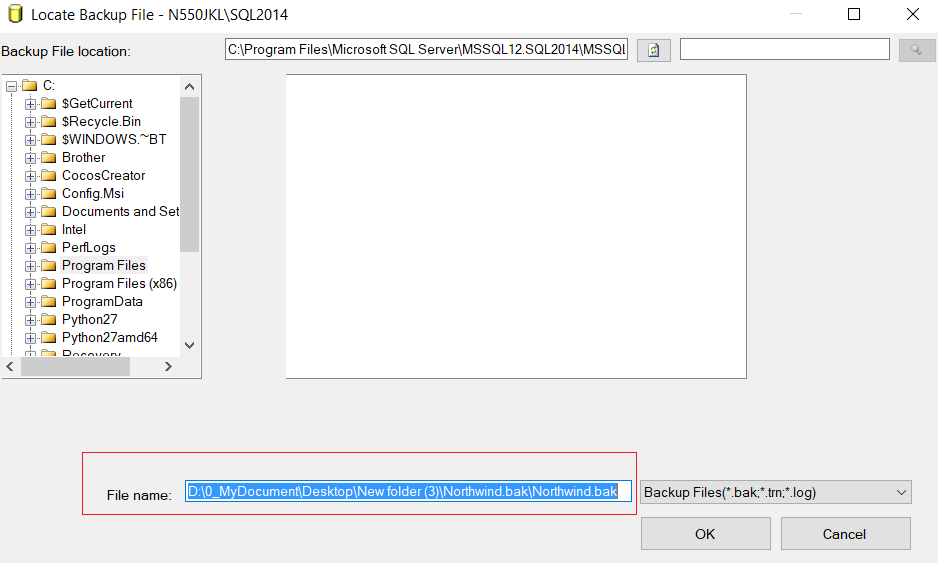
--> Script --> New Query Editor Window

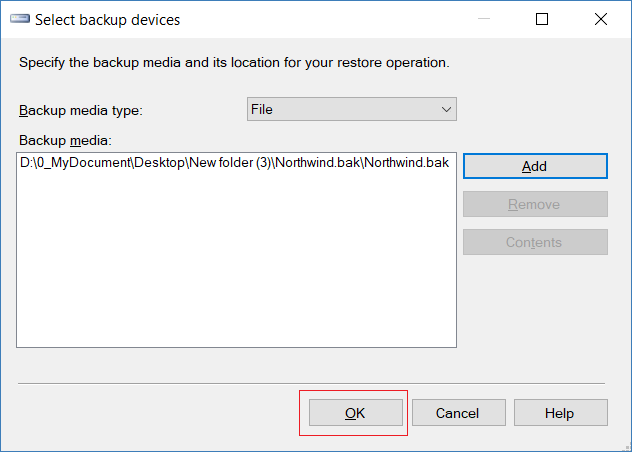
--> Execute the query

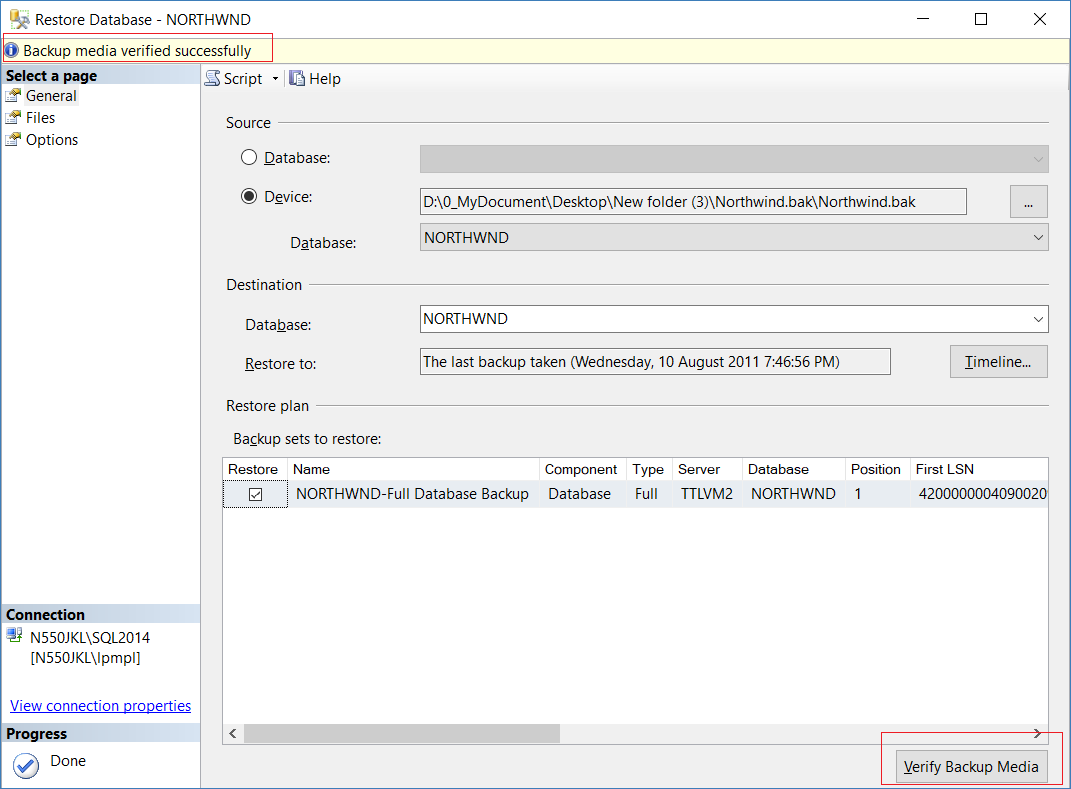


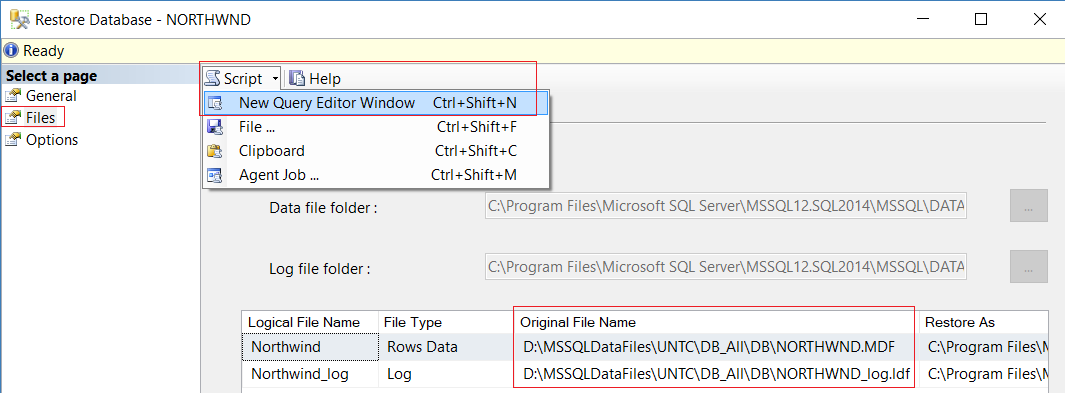


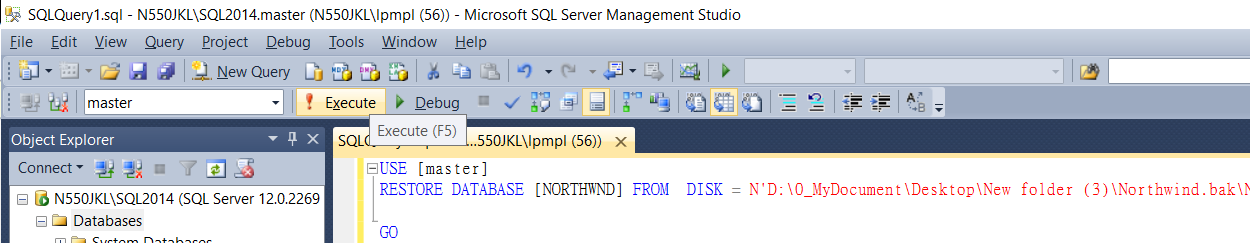








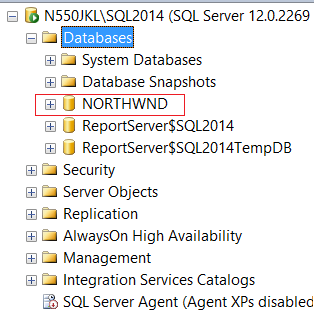




USE [master]

RESTORE DATABASE [NORTHWND] FROM  DISK = N'D:\0\_MyDocument\Desktop\New folder (3)\Northwind.bak\Northwind.bak' WITH  FILE = 1,  MOVE N'Northwind' TO N'C:\Program Files\Microsoft SQL Server\MSSQL12.SQL2014\MSSQL\DATA\NORTHWND.MDF',  MOVE N'Northwind\_log' TO N'C:\Program Files\Microsoft SQL Server\MSSQL12.SQL2014\MSSQL\DATA\NORTHWND\_log.ldf',  NOUNLOAD,  STATS = 5

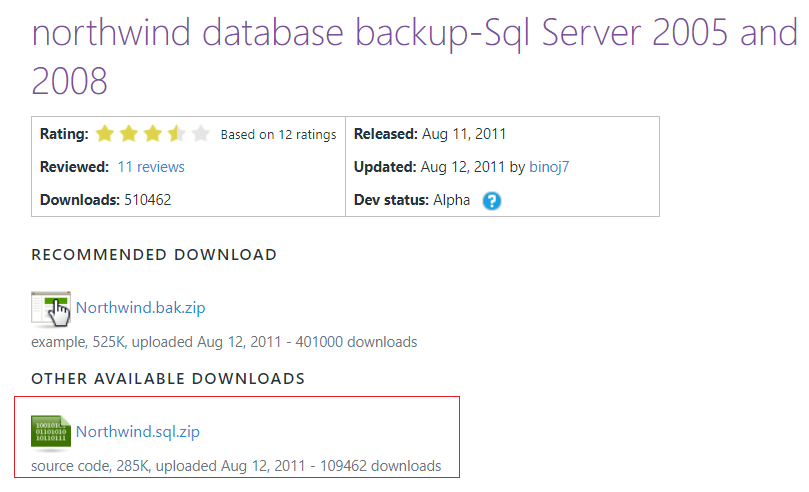
GO



1.2. 2nd way : Restore database from sql file

Go to the following link to download sql file

<https://northwinddatabase.codeplex.com/releases/view/71634>



Download [Northwind.sql.zip](https://northwinddatabase.codeplex.com/downloads/get/269239) and unzip it.

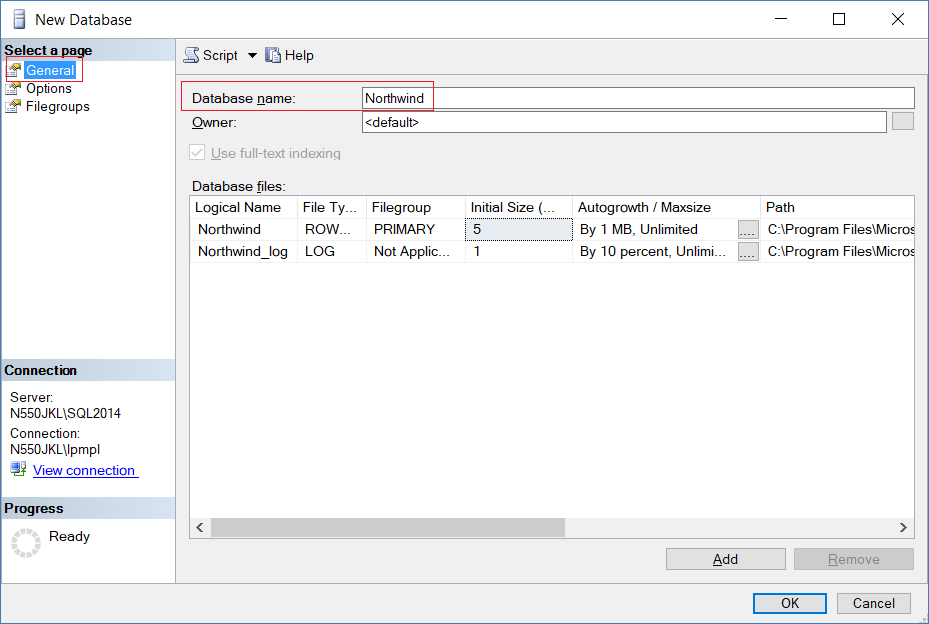
You will get the file called "northwind.sql"

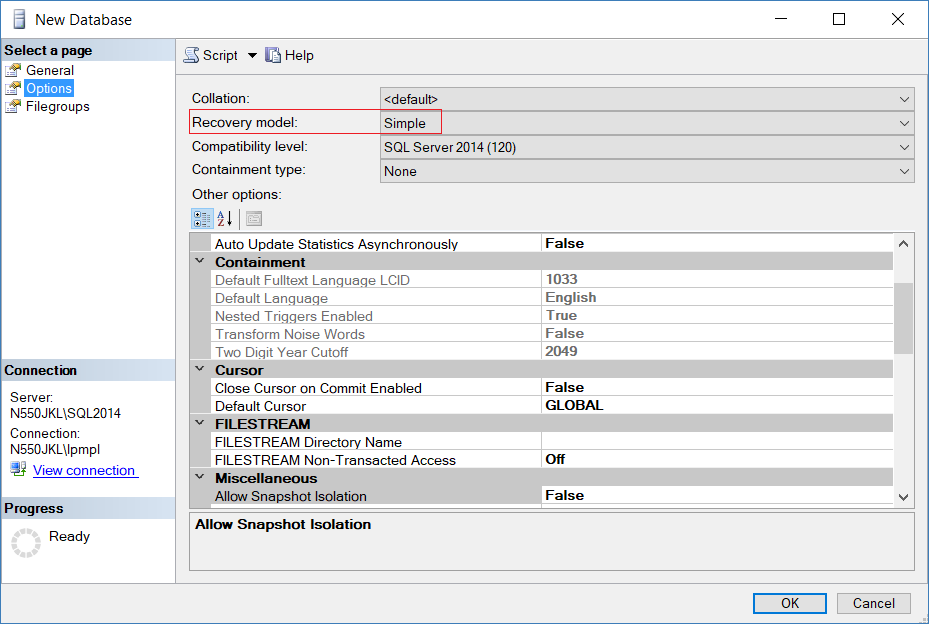
In SQL server

--> Database --> New Database

--> General Tab --> Database Name : **Northwind**

--> Option Tab --> Recovery Model : **Simple**





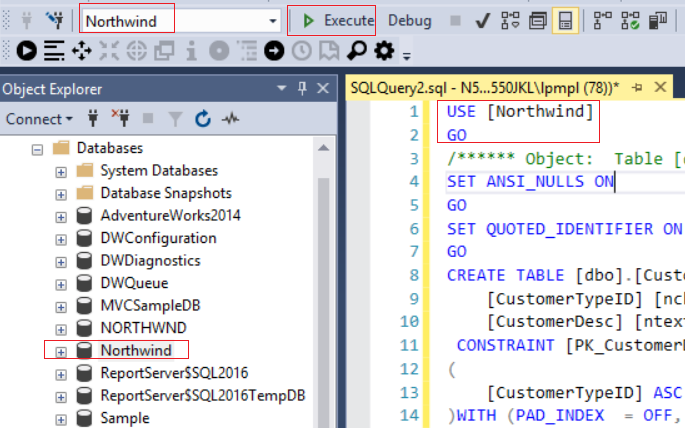
-->

Use Notepad++ to open the file "northwind.sql"

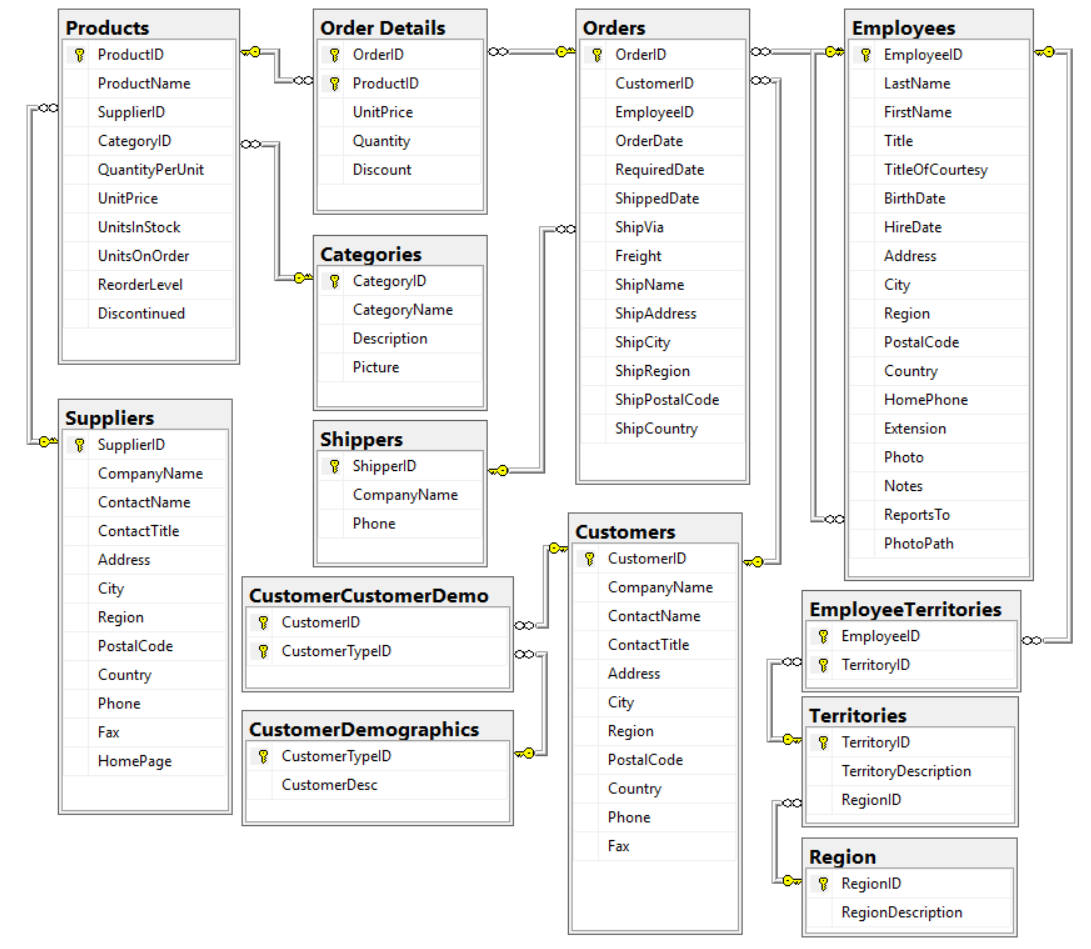
You may download Notepad++ here (<https://notepad-plus-plus.org/>)

Copy the code from "northwind.sql" into new query window.

Execute the query.



1.3. Northwind Database Diagram



========================================================

2. Select

--============================================================================================

--T004\_01\_Select

--============================================================================================

/\*

What to learn

1.

Select (DISTINCT/TOP n)..

WHERE...(AND/OR)...

ORDER BY ...(ASC/DESC)

2.

SQL Operator

Reference:

<https://www.w3schools.com/sql/sql_operators.asp>

<https://www.tutorialspoint.com/sql/sql-wildcards.htm>

<https://docs.microsoft.com/en-us/sql/t-sql/language-elements/wildcard-character-s-to-match-transact-sql>

<https://docs.microsoft.com/en-us/sql/t-sql/language-elements/wildcard-character-s-not-to-match-transact-sql>

2.1.

SQL Comparison Operators

2.1.1. =    : Equal to

2.1.2. >     : Greater than

2.1.3. <     : Less than

2.1.4. >=    : Greater than or equal to

2.1.5. <=    : Less than or equal to

2.1.6. != or <>    : Not equal to

2.2.

SQL Logical Operators

2.2.1. ALL     : TRUE if all of the subquery values meet the condition

2.2.2. SOME    : TRUE if any of the subquery values meet the condition

2.2.3. ANY     : TRUE if any of the subquery values meet the condition

2.2.4. AND     : TRUE if all the conditions separated by AND is TRUE

2.2.5. OR      : TRUE if any of the conditions separated by OR is TRUE

2.2.6. BETWEEN : TRUE if the operand is within the range of comparisons.

2.2.7. IN      : TRUE if the operand is equal to one of a list of expressions

2.2.8. EXISTS  : TRUE if the subquery returns one or more records

2.2.9. LIKE    : TRUE if the operand matches a pattern

2.2.10. NOT    : Displays a record if the condition(s) is NOT TRUE

2.3.

SQL Wildcard Operators

2.3.1. %    : The percent sign means matches zero or more characters.

2.3.2. \_    : The underscore (\_) means matches one character

2.3.3. []   : Matches any single character within the specified range between brackets []

2.3.4. [^]  : Matches any single character that is NOT within the range between brackets []

\*/

--============================================================================================

--T004\_01\_01

--Select

--------------------------------------------------

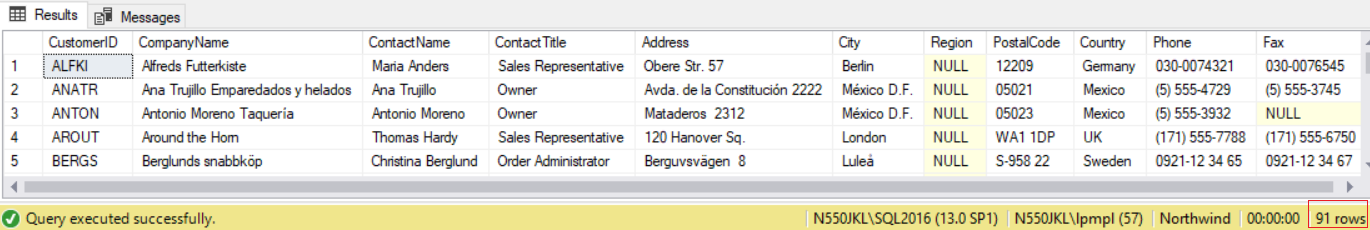
--T004\_01\_01\_01

USE Northwind;

GO -- Run the prvious command and begins new batch

SELECT  \*

FROM    [dbo].[Customers];



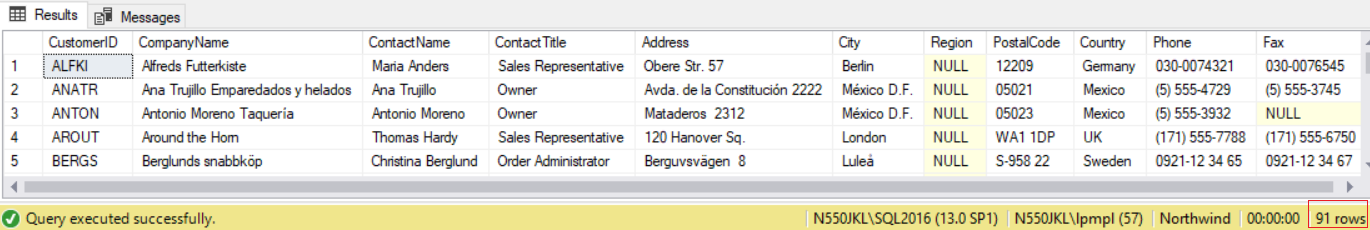
--------------------------------------------------

--T004\_01\_01\_02

SELECT  \*

FROM    [Northwind].[dbo].[Customers];

--[DatabaseName].[SchemaName].[TableName]



--============================================================================================

--T004\_01\_02

--Select (DISTINCT/TOP n)..

--------------------------------------------------

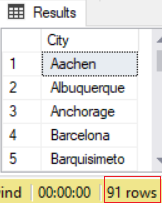
--T004\_01\_02\_01

USE Northwind;

GO -- Run the prvious command and begins new batch

SELECT  [Customers].[City]

FROM    [dbo].[Customers];



--------------------------------------------------

--T004\_01\_02\_02

USE Northwind;

GO -- Run the prvious command and begins new batch

SELECT  DISTINCT

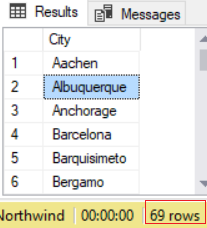
        [Customers].[City]

FROM    [dbo].[Customers];

/\*

DISTINCT filter all the repeated [Customers].[City]

\*/



--------------------------------------------------

--T004\_01\_03

USE Northwind;

GO -- Run the prvious command and begins new batch

SELECT  DISTINCT TOP 1

        [Customers].[City]

FROM    [dbo].[Customers];

/\*

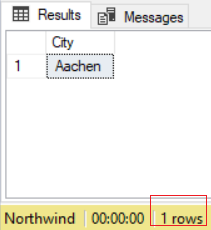
1.

DISTINCT filter all the repeated [Customers].[City]

2.

TOP 1  only get the first one record.

\*/



--------------------------------------------------

--T004\_01\_02\_04

USE Northwind;

GO -- Run the prvious command and begins new batch

SELECT  DISTINCT TOP 10

        [Customers].[City]

FROM    [dbo].[Customers];

/\*

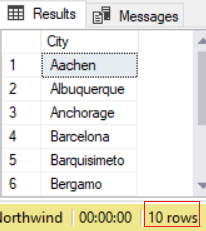
1.

DISTINCT filter all the repeated [Customers].[City]

2.

TOP 10  only get the first ten records.

\*/



--============================================================================================

--T004\_01\_03

--SQL Wildcard Operators

--Select (DISTINCT/TOP n)..

--WHERE...(AND/OR)...

--Reference: <https://www.tutorialspoint.com/sql/sql-wildcards.htm>

--------------------------------------------------

--T004\_01\_03\_01

USE Northwind;

GO -- Run the prvious command and begins new batch

SELECT  \*

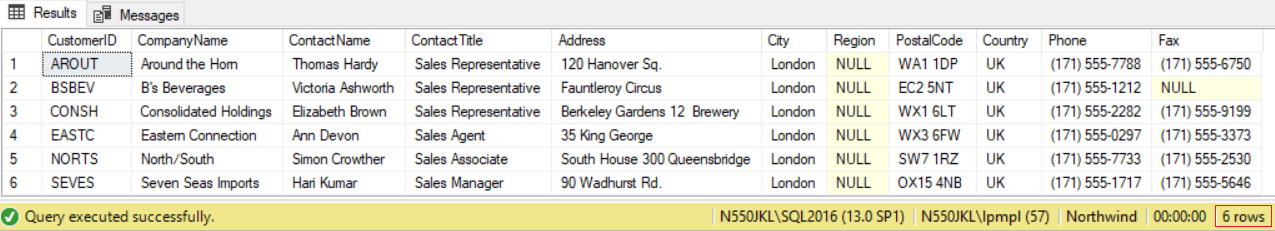
FROM    [dbo].[Customers]

WHERE   [dbo].[Customers].[City] LIKE 'Lo%';

/\*

Finds the customers whoes City name start with 'Lo'.

\*/



--------------------------------------------------

--T004\_01\_03\_02

USE Northwind;

GO -- Run the prvious command and begins new batch

SELECT  \*

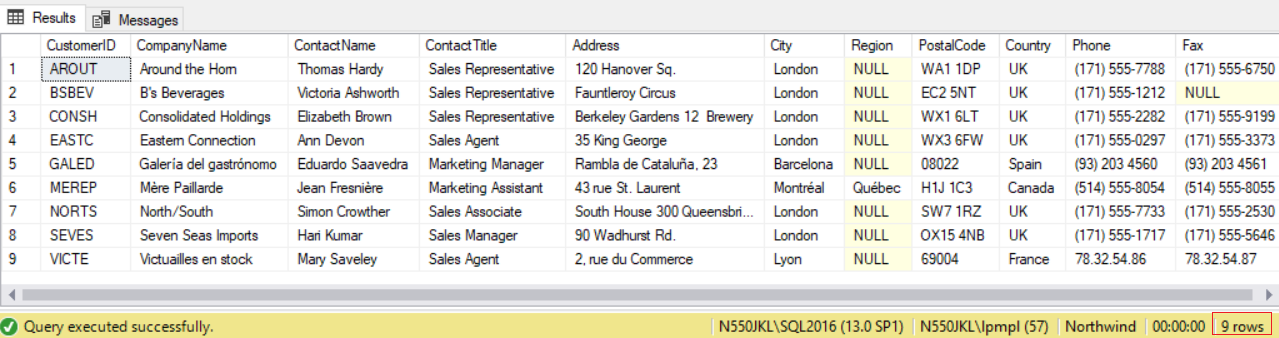
FROM    [dbo].[Customers]

WHERE   [dbo].[Customers].[City] LIKE '%on%';

/\*

Finds the customers whoes City name have 'on' in any position.

\*/



--------------------------------------------------

--T004\_01\_03\_03

USE Northwind;

GO -- Run the prvious command and begins new batch

SELECT  \*

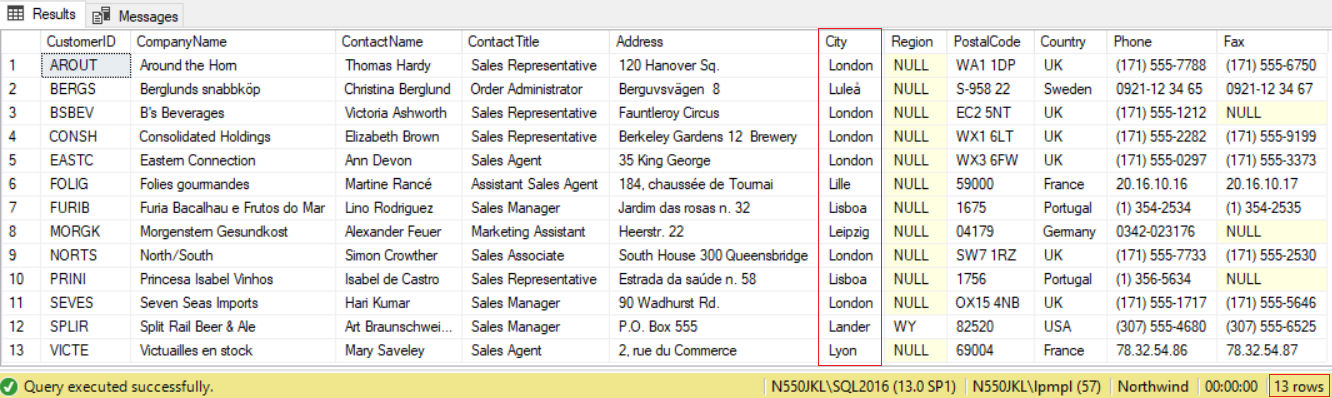
FROM    [dbo].[Customers]

WHERE   [dbo].[Customers].[City] LIKE 'L\_%\_%';

/\*

Finds the customers whoes City name start with 'L' and are at least 3 characters in length.

\*/



--------------------------------------------------

--T004\_01\_03\_04

USE Northwind;

GO -- Run the prvious command and begins new batch

SELECT  \*

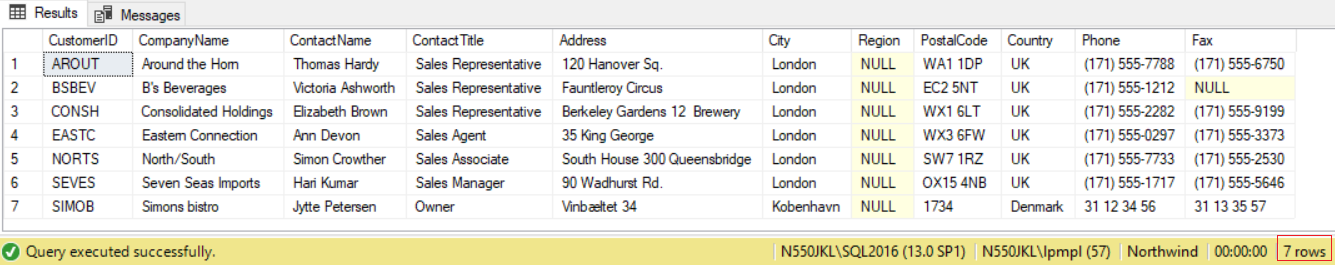
FROM    [dbo].[Customers]

WHERE   [dbo].[Customers].[City] LIKE '\_o%n';

/\*

Finds the customers whoes City name have 'o' in the second position and end with a 'n'.

\*/



--------------------------------------------------

--T004\_01\_03\_05

USE Northwind;

GO -- Run the prvious command and begins new batch

SELECT  \*

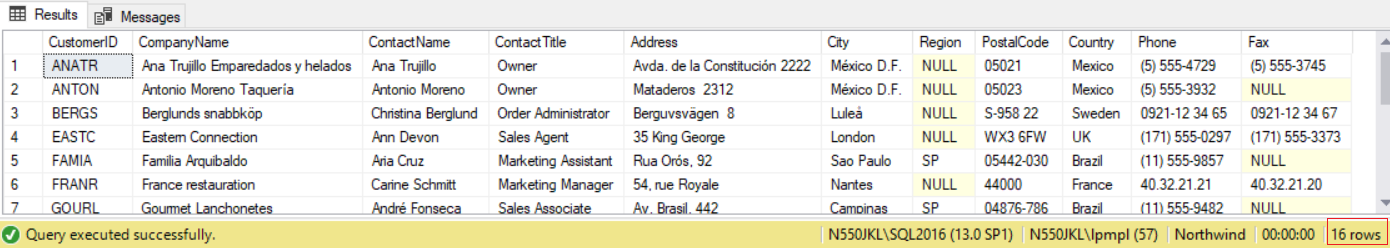
FROM    [dbo].[Customers]

WHERE   [dbo].[Customers].[ContactName] LIKE '[ABC]%';

/\*

Finds the customers whoes [ContactName] start with 'A' or 'B' or 'C'.

\*/



--------------------------------------------------

--T004\_01\_03\_06

USE Northwind;

GO -- Run the prvious command and begins new batch

SELECT  \*

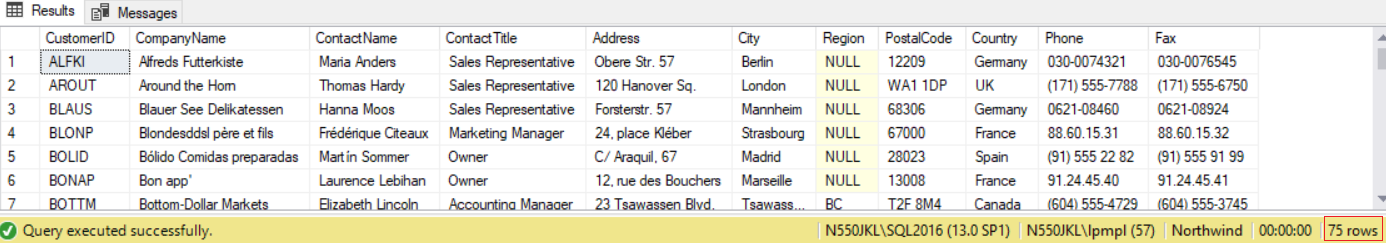
FROM    [dbo].[Customers]

WHERE   [dbo].[Customers].[ContactName] LIKE '[^ABC]%';

/\*

Finds the customers whoes [ContactName] does NOT start with 'A' or 'B' or 'C'.

\*/



--------------------------------------------------

--T004\_01\_03\_07

USE Northwind;

GO -- Run the prvious command and begins new batch

SELECT  \*

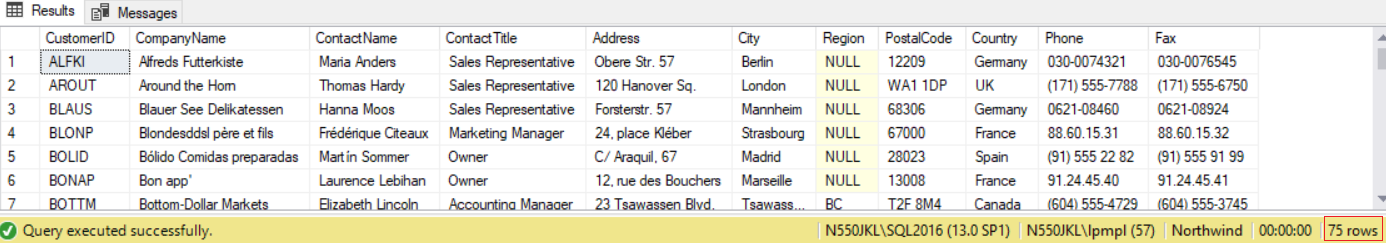
FROM    [dbo].[Customers]

WHERE   [dbo].[Customers].[ContactName] NOT LIKE '[ABC]%';

/\*

Finds the customers whoes [ContactName] does NOT start with 'A' or 'B' or 'C'.

\*/



--------------------------------------------------

--T004\_01\_03\_08

USE Northwind;

GO -- Run the prvious command and begins new batch

SELECT  \*

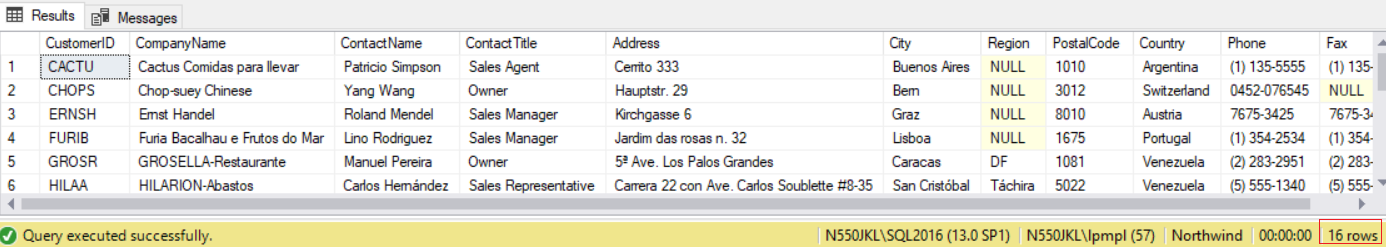
FROM    [dbo].[Customers]

WHERE   [dbo].[Customers].[PostalCode] LIKE '[0-9][0-9][0-9][0-9]';

/\*

Finds the customers whoes [PostalCode] have only 4 digits which are between 0 and 9.

\*/



--============================================================================================

--T004\_01\_04

--Select (DISTINCT/TOP n)..

--WHERE...(AND/OR)...

--ORDER BY ...(ASC/DESC)

--------------------------------------------------

--T004\_01\_04\_01

USE Northwind;

GO -- Run the prvious command and begins new batch

SELECT  \*

FROM    [dbo].[Customers]

WHERE   [dbo].[Customers].[ContactName] LIKE '[ABC]%'

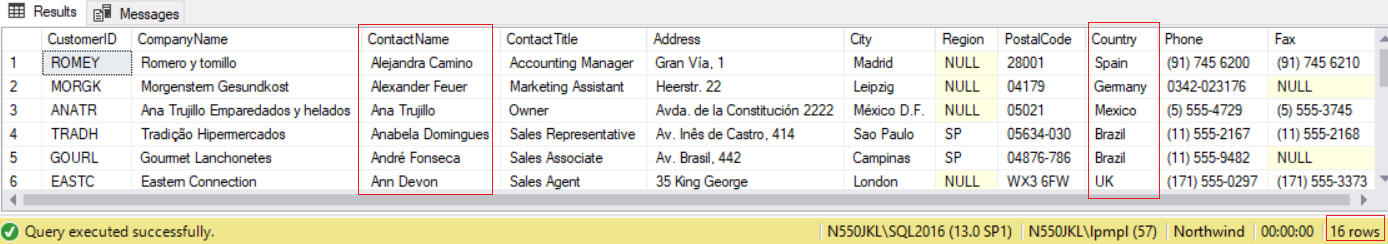
ORDER BY [dbo].[Customers].[ContactName], [dbo].[Customers].[Country];

/\*

Finds the customers whoes [ContactName] start with 'A' or 'B' or 'C'.

Order by the [ContactName] firstly, then order by the [Country] secondly.

\*/



--------------------------------------------------

--T004\_01\_04\_02

USE Northwind;

GO -- Run the prvious command and begins new batch

SELECT  \*

FROM    [dbo].[Customers]

WHERE   [dbo].[Customers].[ContactName] LIKE '[ABC]%'

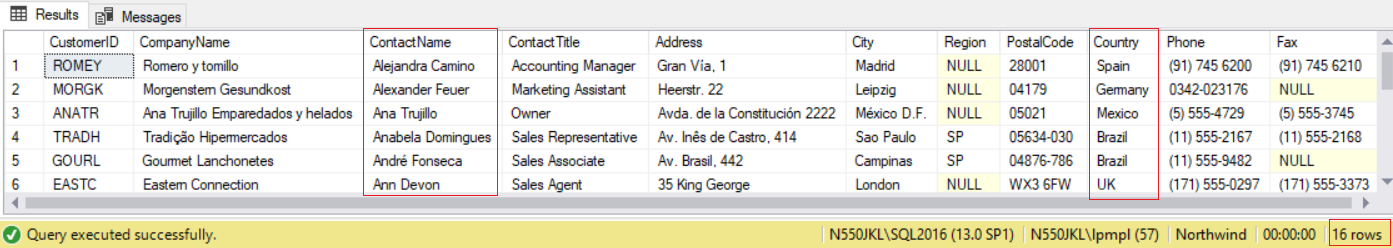
ORDER BY [dbo].[Customers].[ContactName] ASC, [dbo].[Customers].[Country] DESC;

/\*

Finds the customers whoes [ContactName] start with 'A' or 'B' or 'C'.

Order by the [ContactName] ascending firstly, then order by the [Country] descending secondly.

\*/



========================================================

3. Group By

--============================================================================================

--T004\_02\_GroupBy

--============================================================================================

/\*

What to learn

- In order to use aggregate, we need Group By

- aggregate include Count(), Sum(), avg(), Min(), Max().

\*/

--============================================================================================

--T004\_02\_01

--Aggregate : Min(), Max(), avg()

USE Northwind;

GO -- Run the prvious command and begins new batch

SELECT  ProductID ,

        ( ( [UnitPrice] - [Discount] ) \* [Quantity] ) AS TotalSales

FROM    [Northwind].[dbo].[Order Details];

SELECT  [ProductID] ,

        [ProductName] ,

        [SupplierID] ,

        [CategoryID] ,

        [QuantityPerUnit] ,

        [UnitPrice] ,

        [UnitsInStock] ,

        [UnitsOnOrder] ,

        [ReorderLevel] ,

        [Discontinued]

FROM    [Northwind].[dbo].[Products];

SELECT  MIN([UnitPrice])

FROM    [Northwind].[dbo].[Products];

SELECT  MAX([UnitPrice])

FROM    [Northwind].[dbo].[Products];

SELECT  AVG([UnitPrice])

FROM    [Northwind].[dbo].[Products];

SELECT  MIN(( [UnitPrice] - [Discount] ) \* [Quantity]) AS TotalSalesOfTheProductOfTheOrder

FROM    [Northwind].[dbo].[Order Details];

SELECT  MAX(( [UnitPrice] - [Discount] ) \* [Quantity]) AS TotalSalesOfTheProductOfTheOrder

FROM    [Northwind].[dbo].[Order Details];

GO -- Run the prvious command and begins new batch

/\*

1.

1.1.

-- SELECT  MIN([UnitPrice])

The cheapest product unit price.

1.2.

-- SELECT  MAX([UnitPrice])

The most expensive product unit price.

1.3.

-- SELECT  AVG([UnitPrice])

The average product unit price.

2.

--SELECT  MIN(( [UnitPrice] - [Discount] ) \* [Quantity]) AS TotalSalesOfTheProductOfTheOrder

--SELECT  MAX(( [UnitPrice] - [Discount] ) \* [Quantity]) AS TotalSalesOfTheProductOfTheOrder

2.1.

The total sales of each Product of each OrderID is

( [UnitPrice] - [Discount] ) \* [Quantity])

2.2.

--SELECT  MIN(( [UnitPrice] - [Discount] ) \* [Quantity]) AS TotalSalesOfTheProductOfTheOrder

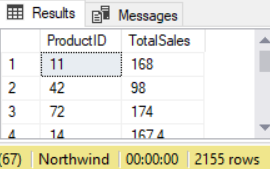
The cheapest total sales of each Product of each OrderID

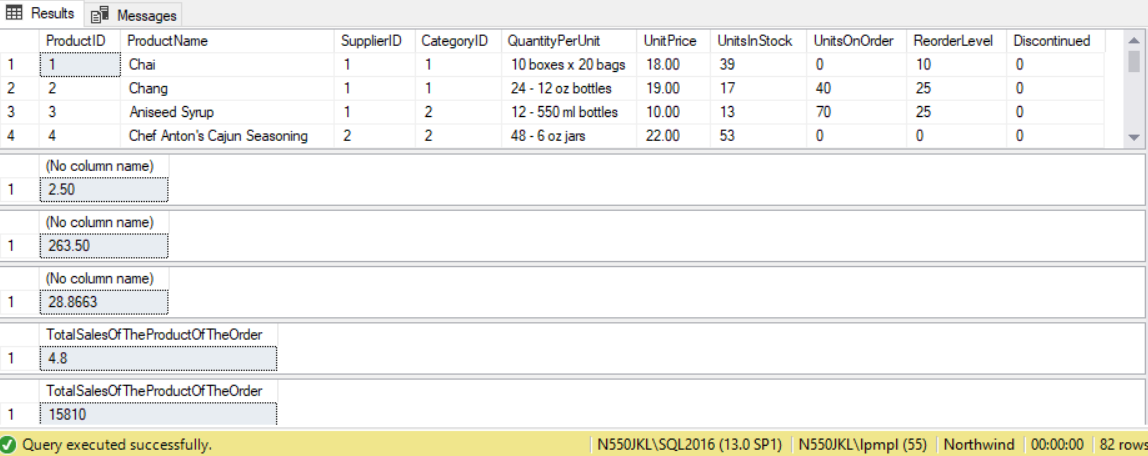
2.3.

--SELECT  MAX(( [UnitPrice] - [Discount] ) \* [Quantity]) AS TotalSalesOfTheProductOfTheOrder

The most expensive total sales of each Product of each OrderID

\*/





--============================================================================================

--T004\_02\_02

--Aggregate : Min(SUM()) cause ERROR

USE Northwind;

GO -- Run the prvious command and begins new batch

SELECT  MIN(SUM(( [UnitPrice] - [Discount] ) \* [Quantity])) AS TotalSalesOfTheOrder

FROM    [Northwind].[dbo].[Order Details];

GO -- Run the prvious command and begins new batch

/\*

1.

-- SELECT  MIN(SUM(( [UnitPrice] - [Discount] ) \* [Quantity])) AS TotalSalesOfTheOrder

1.1.

The total sales of each Product of each OrderID is

( [UnitPrice] - [Discount] ) \* [Quantity])

1.2.

The total sales of each OrderID is

SUM(( [UnitPrice] - [Discount] ) \* [Quantity])

1.3.

--SELECT  MIN(SUM(( [UnitPrice] - [Discount] ) \* [Quantity])) AS TotalSalesOfTheOrder

The cheapest sales of the OrderID.

2.

Output ERROR message

--Msg 130, Level 15, State 1, Line 60

--Cannot perform an aggregate function on an expression containing an aggregate or a subquery.

\*/

--============================================================================================

--T004\_02\_03

--Aggregate : SUM(), COUNT()

USE Northwind;

GO -- Run the prvious command and begins new batch

SELECT  [OrderID] ,

        [ProductID] ,

        [UnitPrice] ,

        [Quantity] ,

        [Discount]

FROM    [Northwind].[dbo].[Order Details];

SELECT  [OrderID] ,

        SUM(( [UnitPrice] - [Discount] ) \* [Quantity]) AS TotalSalesOfTheOrder ,

        COUNT([ProductID]) AS NumberOfProductIDInTheSale

FROM    [Northwind].[dbo].[Order Details]

GROUP BY [OrderID]

ORDER BY [OrderID];

GO -- Run the prvious command and begins new batch

/\*

1.

-- COUNT([ProductID]) AS NumberOfProductIDInTheSale

Each OrderID has several ProductID.

Thus, number of productID in the sales is COUNT([ProductID])

2.

-- SUM(( [UnitPrice] - [Discount] ) \* [Quantity]) AS TotalSalesOfTheOrder

2.1.

The total sales of each Product of each OrderID is

( [UnitPrice] - [Discount] ) \* [Quantity])

2.2.

The total sales of each OrderID is

SUM(( [UnitPrice] - [Discount] ) \* [Quantity])

3.

In order to use aggregate, we need Group By

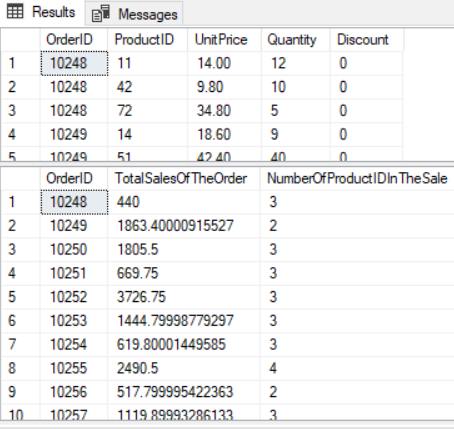
aggregate include Count(), Sum(), avg(), Min(), Max().

3.1.

[OrderID] is the only thing without aggregate in the SELECT clause.

Thus, we need to GROUP BY [OrderID]

\*/



--============================================================================================

--T004\_02\_04

--Aggregate : SUM()

--Different between WHERE and HAVING

--Ch11\_04\_01

USE Northwind;

GO -- Run the prvious command and begins new batch

SELECT  [OrderID] ,

        [ProductID] ,

        SUM(( [UnitPrice] - [Discount] ) \* [Quantity]) AS TotalSalesByProduct

FROM    [Northwind].[dbo].[Order Details]

WHERE   OrderID BETWEEN 10249 AND 10328

GROUP BY [OrderID] ,

        [ProductID]

--HAVING  OrderID BETWEEN 10249 AND 10328

ORDER BY [OrderID];

--Ch11\_04\_02

SELECT  [OrderID] ,

        [ProductID] ,

        SUM(( [UnitPrice] - [Discount] ) \* [Quantity]) AS TotalSalesByProduct

FROM    [Northwind].[dbo].[Order Details]

--WHERE OrderID BETWEEN 10249 AND 10328

GROUP BY [OrderID] ,

        [ProductID]

HAVING  OrderID BETWEEN 10249 AND 10328

ORDER BY [OrderID];

GO -- Run the prvious command and begins new batch

/\*

1.

1.1.

WHERE filters rows before aggregation (GROUP BY)

and can be used with - Select, Insert, Update statements.

1.2.

HAVING filters groups after the aggregations (GROUP BY)

and can only be used with the Select statement

2.

--WHERE OrderID BETWEEN 10249 AND 10328

and

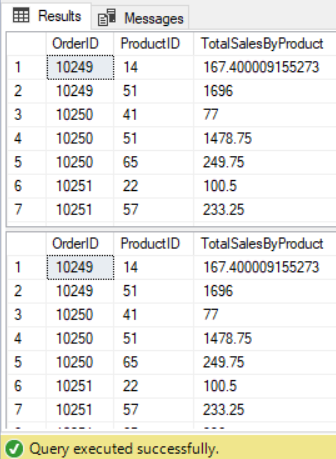
--HAVING  OrderID BETWEEN 10249 AND 10328

both will get the same result.

However, try to eliminate rows that you don't want as early as possible.

Thus, in this case, using WHERE is better than using HAVING.

\*/



--============================================================================================

--T004\_02\_05

--Aggregate : SUM()

--Aggregation (GROUP BY) can not be used in the WHERE clause.

--but it is ok in HAVING clause.

USE Northwind;

GO -- Run the prvious command and begins new batch

SELECT  [OrderID] ,

        [ProductID] ,

        SUM(( [UnitPrice] - [Discount] ) \* [Quantity]) AS TotalSalesByProduct

FROM    [Northwind].[dbo].[Order Details]

--WHERE SUM(( [UnitPrice] - [Discount] ) \* [Quantity]) > 500   --Error

GROUP BY [OrderID] ,

        [ProductID]

HAVING  SUM(( [UnitPrice] - [Discount] ) \* [Quantity]) > 500

ORDER BY [OrderID];

GO -- Run the prvious command and begins new batch

/\*

1.

1.1.

WHERE filters rows before aggregation (GROUP BY)

and can be used with - Select, Insert, Update statements.

1.2.

HAVING filters groups after the aggregations (GROUP BY)

and can only be used with the Select statement

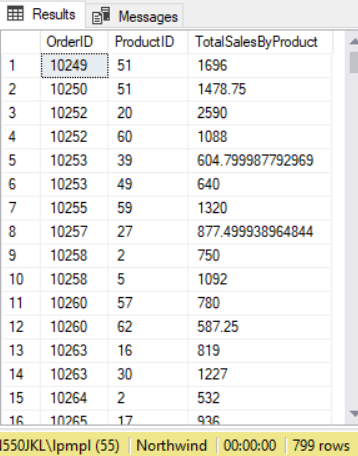
2.

aggregation (GROUP BY) can be used in Having clause.

However, aggregation (GROUP BY) can not be used in the WHERE clause,

unless aggregation (GROUP BY) is in a sub query with a HAVING clause.

\*/



--============================================================================================

--T004\_02\_06

--Aggregate : SUM()

--WHERE with HAVING

USE Northwind;

GO -- Run the prvious command and begins new batch

SELECT  [OrderID] ,

        [ProductID] ,

        SUM(( [UnitPrice] - [Discount] ) \* [Quantity]) AS TotalSalesByProduct

FROM    [Northwind].[dbo].[Order Details]

WHERE   OrderID BETWEEN 10249 AND 10328

GROUP BY [OrderID] ,

        [ProductID]

HAVING  SUM(( [UnitPrice] - [Discount] ) \* [Quantity]) > 500

ORDER BY [OrderID];

GO -- Run the prvious command and begins new batch

/\*

1.

1.1.

WHERE filters rows before aggregation (GROUP BY)

and can be used with - Select, Insert, Update statements.

1.2.

HAVING filters groups after the aggregations (GROUP BY)

and can only be used with the Select statement

2.

aggregation (GROUP BY) can be used in Having clause.

However, aggregation (GROUP BY) can not be used in the WHERE clause,

unless aggregation (GROUP BY) is in a sub query with a HAVING clause.

3.

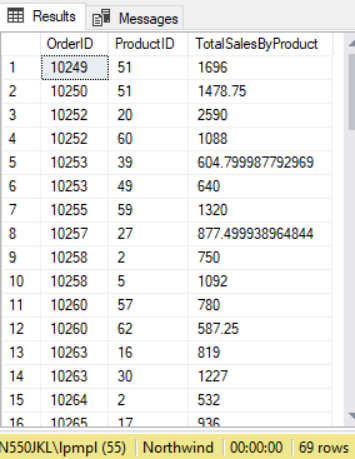
In this case,

it will execute WHERE clause,

then execute   aggregation (GROUP BY, SUM()),

then execute   HAVING clause.

\*/



========================================================

4. Where V.S. Having

--============================================================================================

--T004\_03\_Where\_Having

--============================================================================================

/\*

1.

WHERE V.S. Having

1.1.

WHERE and HAVING can be used together.

Aggregates includes Count, Sum, Avg, Min and Max.

Aggregates must be happened after Group By,

because Aggregates need Group By to perform.

1.2.

WHERE operator filters the rows before Group by.

Thus, WHERE operator CAN NOT be used to filter Aggregates.

HAVING operator filters the Group after Group by.

Thus, HAVING operator CAN filter Aggregates.

1.3.

HAVING is slower than WHERE.

Try to use WHERE to replace HAVING if possible.

\*/

--=========================================================

--T004\_03\_01

--Create Sample Data

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'StoreSales' ) )

    BEGIN

        TRUNCATE TABLE dbo.StoreSales;

        DROP TABLE StoreSales;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE StoreSales

(

  Id INT IDENTITY(1, 1)

         PRIMARY KEY ,

  StoreName NVARCHAR(100) ,

  Sales MONEY

);

GO -- Run the previous command and begins new batch

INSERT  INTO StoreSales

VALUES  ( 'Store01', 350 );

INSERT  INTO StoreSales

VALUES  ( 'Store02', 690 );

INSERT  INTO StoreSales

VALUES  ( 'Store01', 700 );

INSERT  INTO StoreSales

VALUES  ( 'Store02', 150 );

INSERT  INTO StoreSales

VALUES  ( 'Store05', 880 );

INSERT  INTO StoreSales

VALUES  ( 'Store02', 860 );

INSERT  INTO StoreSales

VALUES  ( 'Store03', 210 );

INSERT  INTO StoreSales

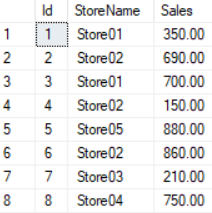
VALUES  ( 'Store04', 750 );

GO -- Run the previous command and begins new batch

SELECT  \*

FROM    StoreSales;

GO -- Run the previous command and begins new batch



--=========================================================

--T004\_03\_02

SELECT  StoreName ,

        SUM(Sales) AS Total

FROM    StoreSales

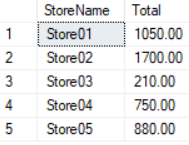
GROUP BY StoreName;

GO -- Run the previous command and begins new batch

/\*

GROUP BY StoreName will get the total sales of each store,

\*/



--=========================================================

--T004\_03\_03

SELECT  StoreName ,

        SUM(Sales) AS Total

FROM    StoreSales

GROUP BY StoreName

HAVING  SUM(Sales) > 1100;

GO -- Run the previous command and begins new batch

/\*

GROUP BY StoreName will get the total sales of each store,

HAVING operator filter the Group

which otal sales of each store is greater than 1100.

\*/



--=========================================================

--T004\_03\_04

SELECT  StoreName ,

        SUM(Sales) AS Total

FROM    StoreSales

WHERE   StoreName IN ( 'Store01', 'Store02' )

GROUP BY StoreName;

GO -- Run the previous command and begins new batch

/\*

WHERE operator filter the rows before Group by.

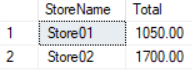
--WHERE StoreName in ('Store01', 'Store02')

Only get the Store01 and Store02 rows.

Then GROUP BY StoreName will

get the total sales of Store01 and Store02

\*/



--=========================================================

--T004\_03\_05

SELECT  StoreName ,

        SUM(Sales) AS Total

FROM    StoreSales

GROUP BY StoreName

HAVING  StoreName IN ( 'Store01', 'Store02' );

GO -- Run the previous command and begins new batch

/\*

HAVING operator filter the Group after Group by.

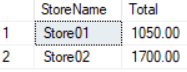
--HAVING StoreName in ('Store01', 'Store02')

Only get the Store01 and Store02 Groups.

Then GROUP BY StoreName will

get the total sales of Store01 and Store02.

\*/



--=========================================================

--T004\_03\_06

--Clean up

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'StoreSales' ) )

    BEGIN

        TRUNCATE TABLE dbo.StoreSales;

        DROP TABLE StoreSales;

    END;

GO -- Run the previous command and begins new batch