Microsoft SQL from A to Z

Intro to SELECT Statements

An **instance** is an installation of a SQL server. Each has a unique name on the network.

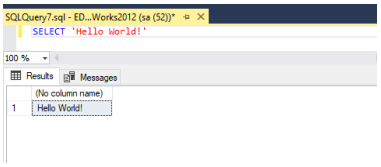
Database is an organized collection of data.

Schema is a collection of database objects associated with one particular database username.

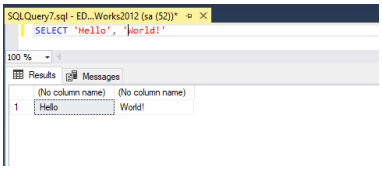
Literal SELECT statement

A statement that does not directly query a particular table and return columns, but instead returns the results of a string or mathematical expression.

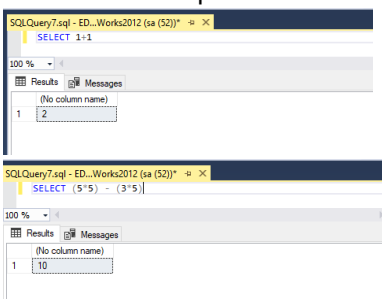
A string SELECT.



To use two columns.



Mathematical expressions.

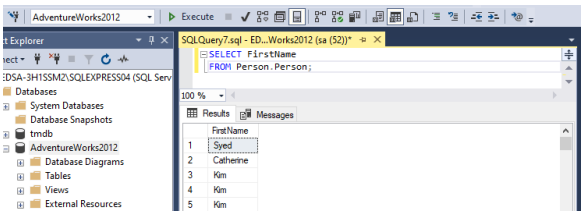


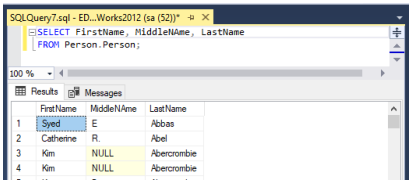
Basic SELECT statement

A statement that does directly query a particular table and return columns

General form:

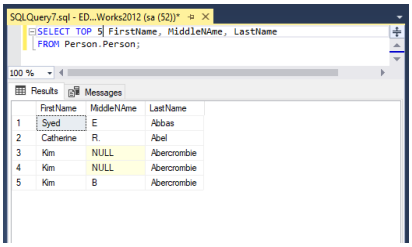
We do not specify the Database name in the form **AdventureWorks2012.Person.Person** because we are connected to the exact database.



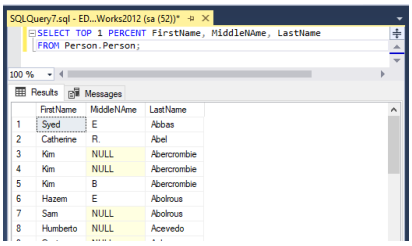


Top operator is used to limit the amount of rows displayed.

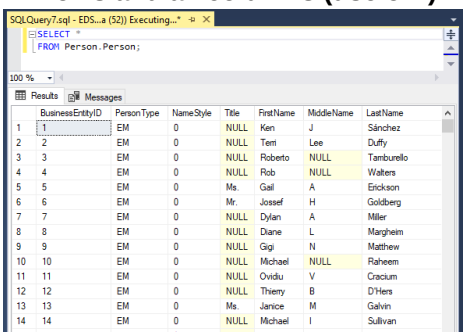
Top 5 rows.



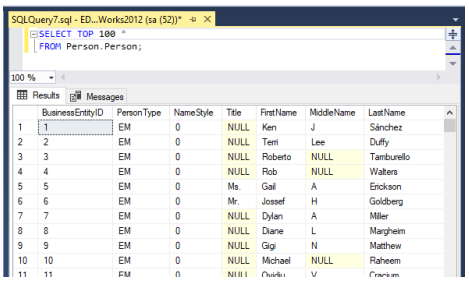
Top 1% rows. (200 rows in total)



All rows and all columns (use of *)



Use ***** with the TOP operator (100 rows)

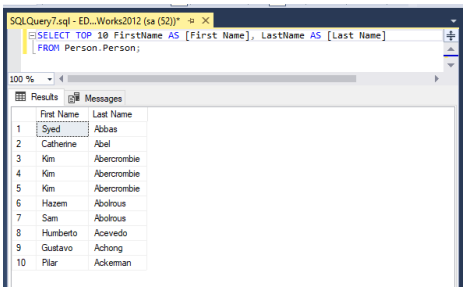


	BusinessEntityID	PersonType	NameStyle	Title	FirstName	MiddleName	LastName
1	1	EM	0	NULL	Ken	J	Sánchez
2	2	EM	0	NULL	Terri	Lee	Duffy
3	3	EM	0	NULL	Roberto	NULL	Tamburello
4	4	EM	0	NULL	Rob	NULL	Walters
5	5	EM	0	Ms.	Gail	A	Erickson
6	6	EM	0	Mr.	Jossef	H	Goldberg
7	7	EM	0	NULL	Dylan	A	Miller
8	8	EM	0	NULL	Diane	L	Margheim
9	9	EM	0	NULL	Gigi	N	Matthew
10	10	EM	0	NULL	Michael	NULL	Raheem

ALIAS is used to change the column name as viewed in the results panel, but do not change the column name in the table.

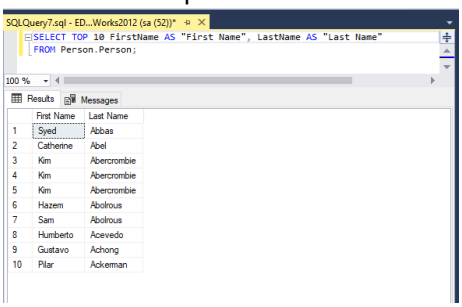
To create spaces in column names and use of ALIAS.

Use of square brackets []



	First Name	Last Name
1	Syed	Abbas
2	Catherine	Abel
3	Kim	Abercrombie
4	Kim	Abercrombie
5	Kim	Abercrombie
6	Hazem	Abolrous
7	Sam	Abolrous
8	Humberto	Acevedo
9	Gustavo	Achong
10	Pilar	Ackerman

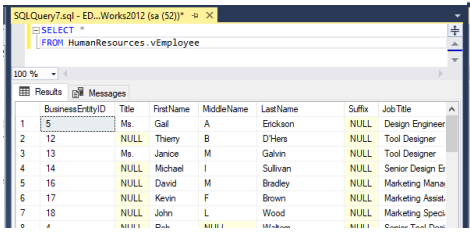
Use of double quotes “”



	First Name	Last Name
1	Syed	Abbas
2	Catherine	Abel
3	Kim	Abercrombie
4	Kim	Abercrombie
5	Kim	Abercrombie
6	Hazem	Abolrous
7	Sam	Abolrous
8	Humberto	Acevedo
9	Gustavo	Achong
10	Pilar	Ackerman

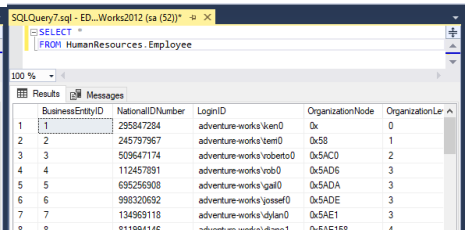
A **view** is a virtual table based on the result set of an SQL statement.
Use of SQL functions, WHERE, and JOIN statements to view and present the data as if the data were coming from one single table.

A view.



	BusinessEntityID	Title	FirstName	MiddleName	LastName	Suffix	JobTitle
1	5	Ms.	Gail	A	Erickson	NULL	Design Engineer
2	12	NULL	Thierry	B	D'Hers	NULL	Tool Designer
3	13	Ms.	Janice	M	Galvin	NULL	Tool Designer
4	14	NULL	Michael	I	Sullivan	NULL	Senior Design Er
5	16	NULL	David	M	Bradley	NULL	Marketing Mana
6	17	NULL	Kevin	F	Brown	NULL	Marketing Assist
7	18	NULL	John	L	Wood	NULL	Marketing Speci

A table.

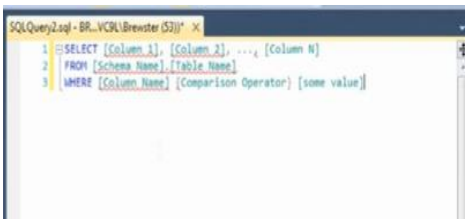


	BusinessEntityID	NationalIDNumber	LoginID	OrganizationNode	OrganizationLe
1	1	295847284	adventure-works\ken0	0x	0
2	2	245797967	adventure-works\terri0	0x58	1
3	3	509647174	adventure-works\roberto0	0x5AC0	2
4	4	112457891	adventure-works\rob0	0x5AD6	3
5	5	695256908	adventure-works\gail0	0x5ADA	3
6	6	998320692	adventure-works\jossef0	0x5ADE	3
7	7	134969118	adventure-works\dylan0	0x5AE1	3

Filtering with the WHERE clause

The **WHERE** clause is used to filter records (or rows of data).

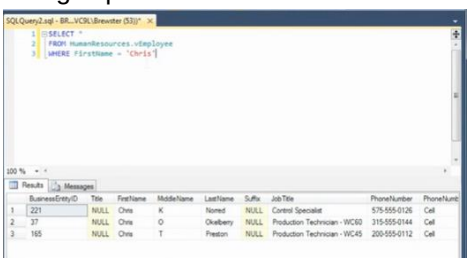
General form:



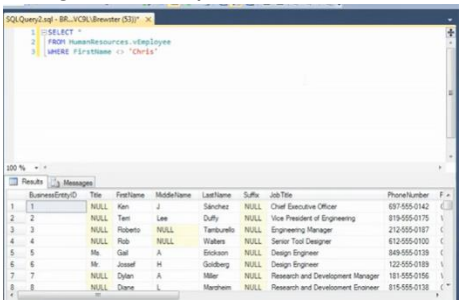
SQL comparison operators.

Operator	Description
=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<>	Not equal to

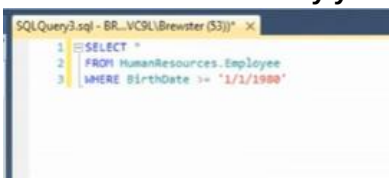
Using Equal to ==



Using the Not equal to <>



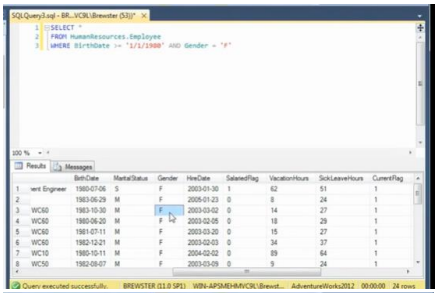
Filter dates the same way you would represent strings in SQL.

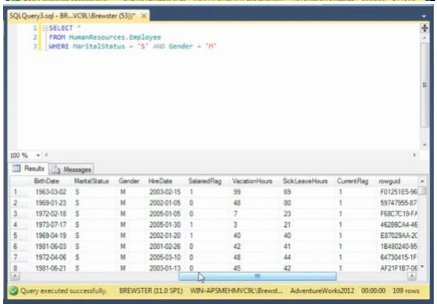


SQL logical operator

AND (conjunction)

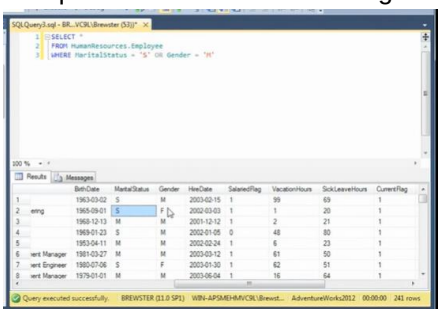
It requires that the filtering criteria is true and returns a result to meet the condition.





OR (disjunction)

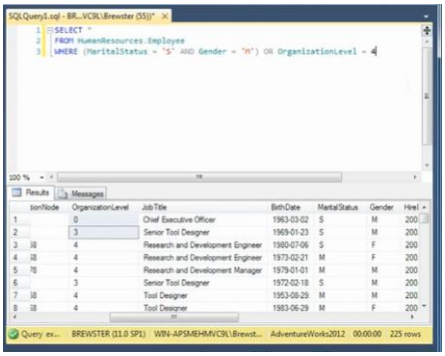
It requires that one of the filtering criteria is true and returns a result to meet the condition.

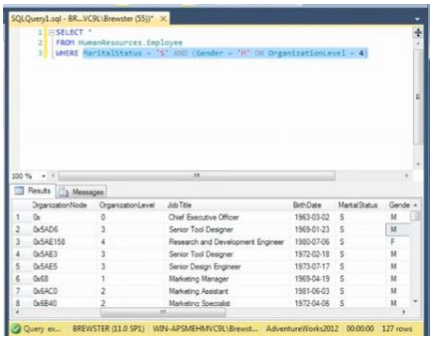


Conjunction

The logical conjunction is used to check that two conditions are true.

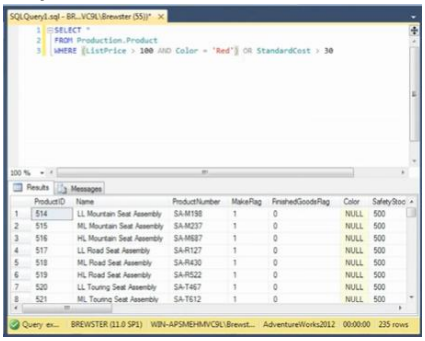
The use of parentheses changes the meaning of the expression when using logical operators.

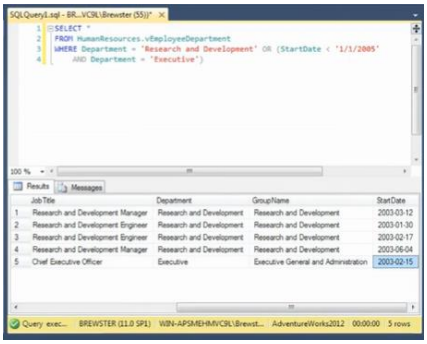


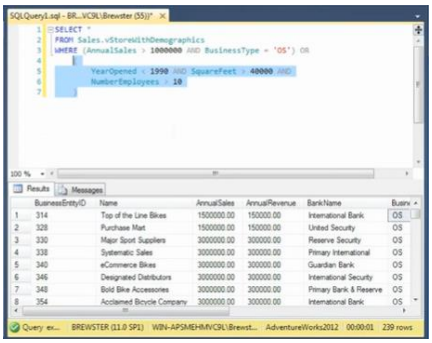


Use of logical and comparison operators

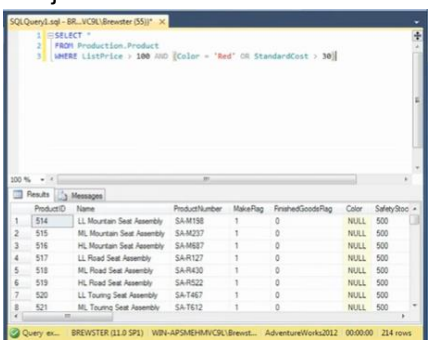
Disjunction





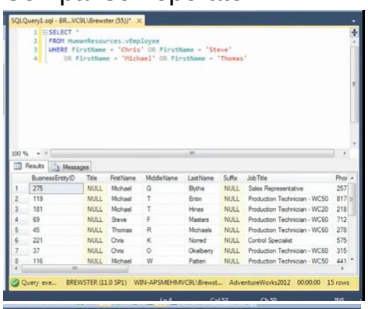


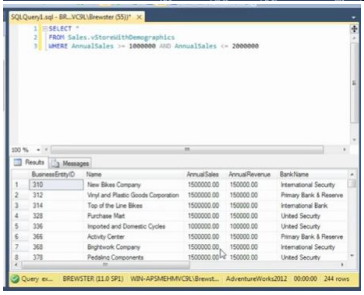
Conjunction



Comparison operators, IN, BETWEEN

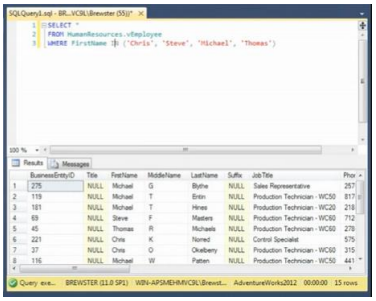
Comparison operator



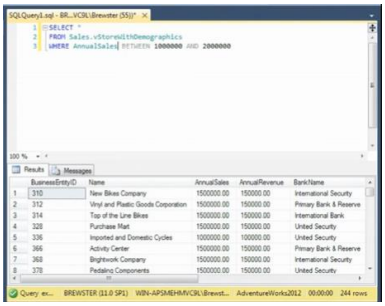


IN

Allows you specify multiple values in a WHERE clause in the form of a list.



BETWEEN



Wildcard operator and characters

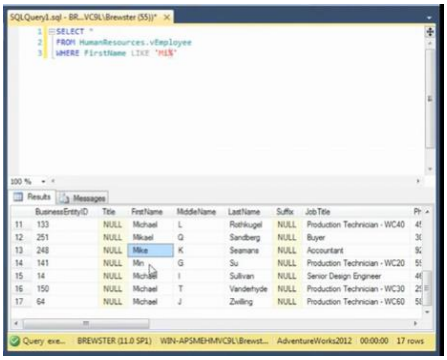
Wildcard Characters in SQL Server

Symbol	Description	Example
%	Represents zero or more characters	bl% finds bl, black, blue, and blob
_	Represents a single character	h_t finds hot, hat, and hit
[]	Represents any single character within the brackets	h[oa]t finds hot and hat, but not hit
^	Represents any character not in the brackets	h[^oa]t finds hit, but not hot and hat
-	Represents a range of characters	c[a-b]t finds cat and cbt

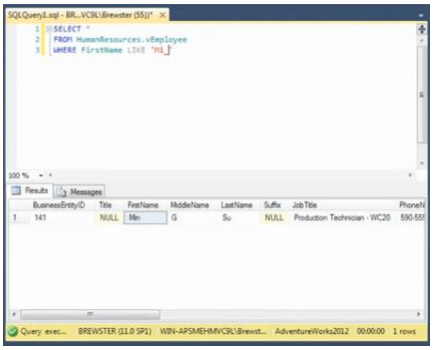
Here are some examples showing different LIKE operators with '%' and '_' wildcards:

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that starts with "a"
WHERE CustomerName LIKE '%a'	Finds any values that ends with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%_%'	Finds any values that starts with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that starts with "a" and ends with "o"

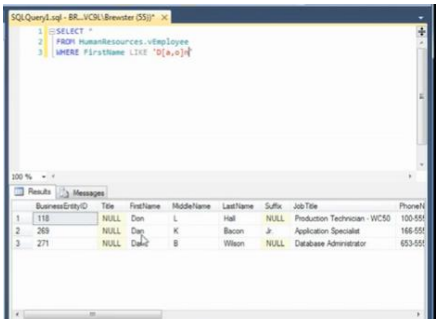
% represents 0 to multiple characters

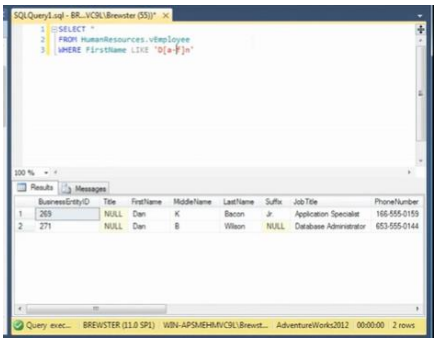


_ (underscore) is used for a single character

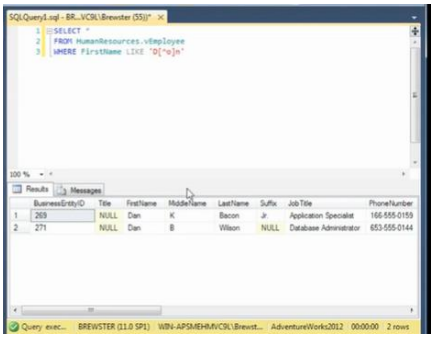


[] to find specific characters or a range





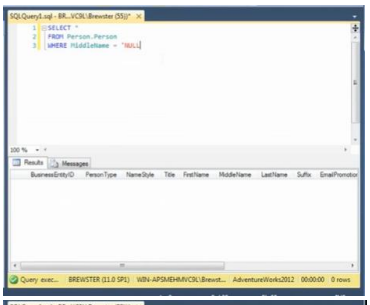
Not a specific character (exclude ‘o’)

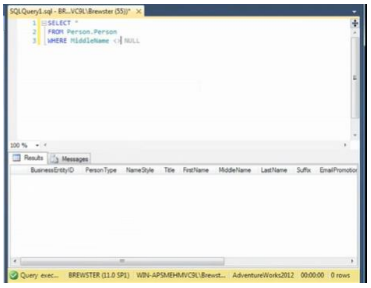


NULL - means nothing, not a blank space.

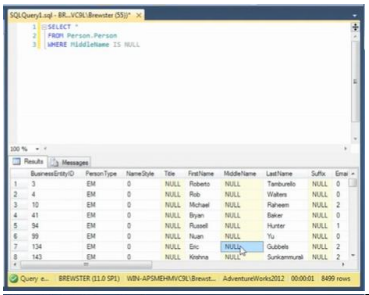
Filter on NULL.

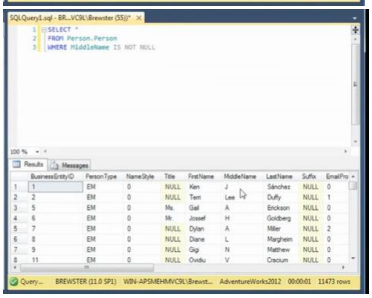
This sees NULL as a value. Not how to use NULL.





How to use NULL

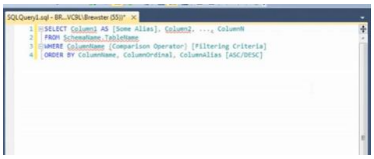




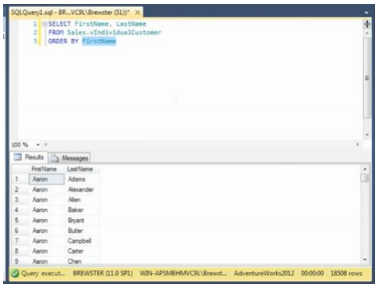
Sorting data with the ORDER BY clause

The ORDER BY clause is used to sort results in ascending or descending order.

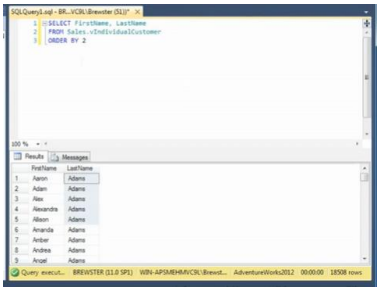
General form



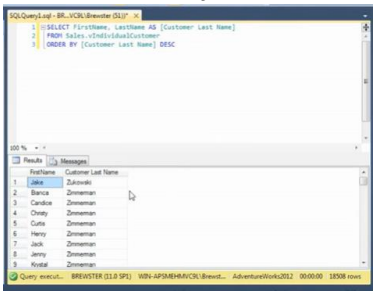
In ascending order (automatic)



Order columns by the second column (ordinal position). **NO OTHER CLAUSE DOES THIS.**

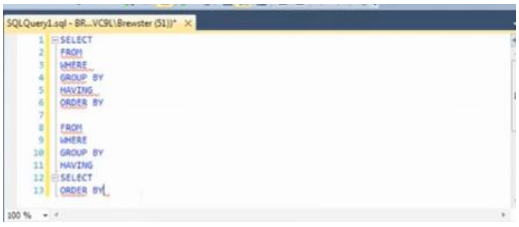


Order columns by the alias

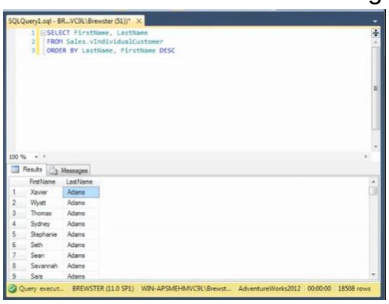


How SQL evaluates the clauses.

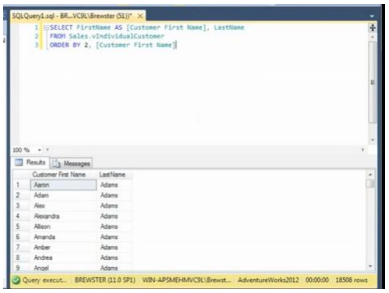
- First 6 is how we write the clauses.
- Second 6 is how SQL evaluates the clauses.



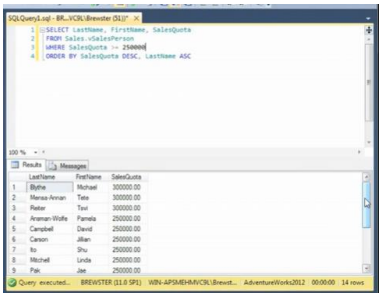
Sort the LastName in ascending order and FirstName is descending order.



Sort the LastName and FirstName in ascending order.



Sort SalesQuota DESC and LastName ASC where SalesQuota >= 250000



Querying multiple tables with JOINS

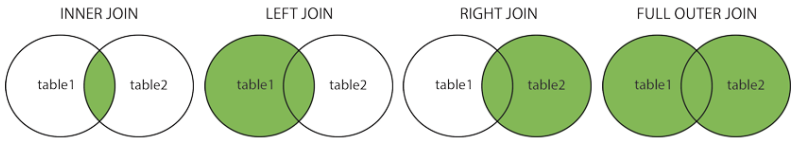
A JOIN is a means for combining columns from one (self-join) or more tables by using values common to each.

Foreign keys indicate how we can JOIN tables.

Different Types of SQL JOINs

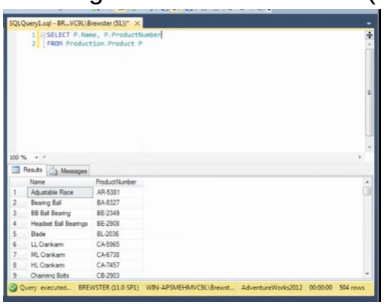
Here are the different types of the JOINs in SQL:

- **(INNER) JOIN**: Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN**: Returns all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN**: Returns all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN**: Returns all records when there is a match in either left or right table

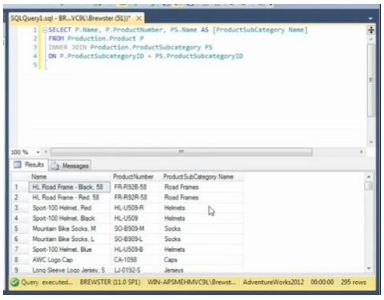


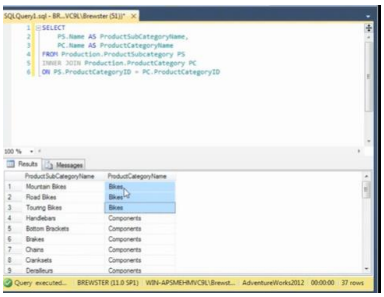
INNER JOIN
An intersection between two tables.

Creating an alias for the table. (Production.Product is now P)

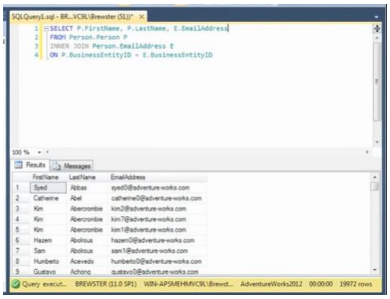


Use INNER join to connect the rows where there is a matching column.

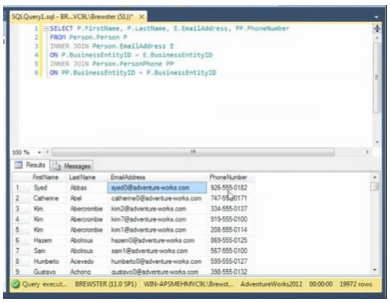




Use one INNER JOIN



Use two INNER JOINS

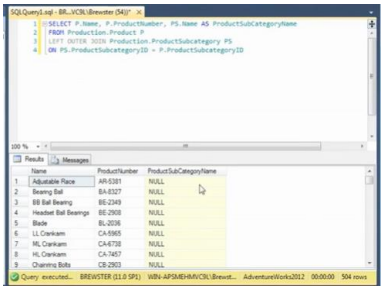


LEFT OUTER JOIN

All information for table A and an intersection between two tables .

LEFT OUTER JOIN

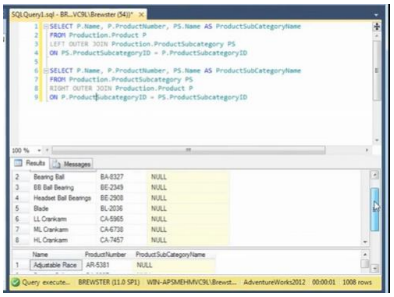
There is NULL under the ProductSubCategoryName as there is no match between the two tables for the ProductSubCategoryID column.



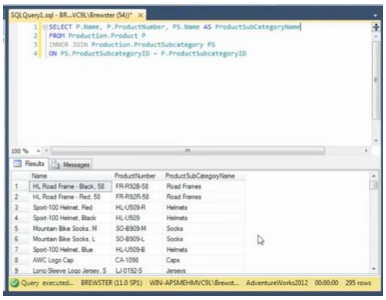
RIGHT OUTER JOIN

All information for table B and an intersection between two tables .

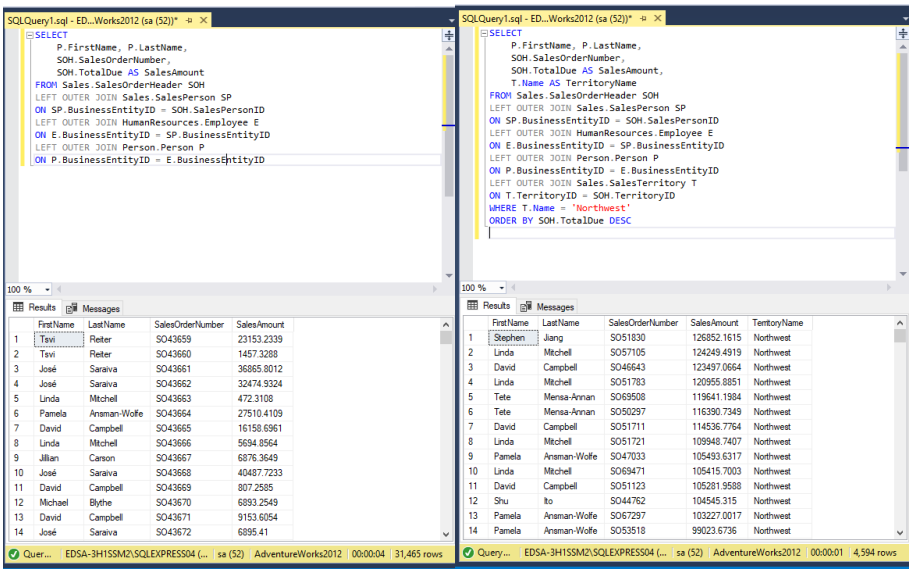
RIGHT OUTER JOIN (follows above example from LEFT OUTER JOIN)



INNER JOIN (follows above example from LEFT OUTER JOIN)



Example: LEFT OUTER JOIN



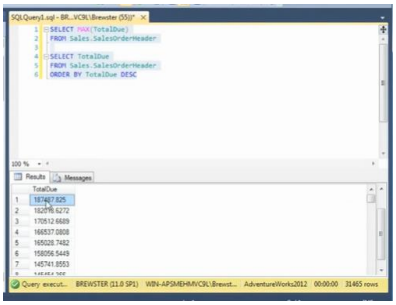
Aggregate Functions

An **aggregate function** performs a calculation on a set of values, and returns a single value.

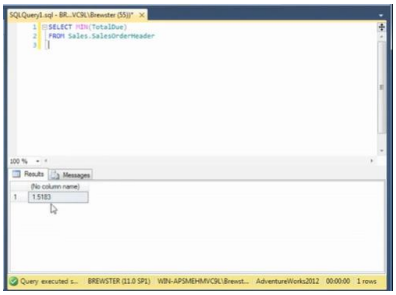
Except for COUNT , **aggregate functions** ignore null values.

Aggregate functions (COUNT, MAX, MIN, SUM, AVG) are often used with the GROUP BY clause of the SELECT statement to group the result-set by one or more columns.

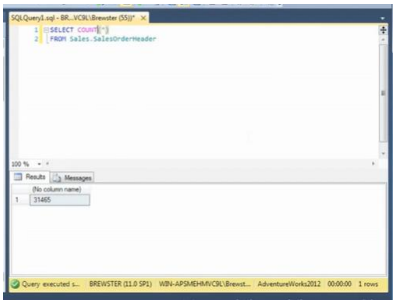
MAX

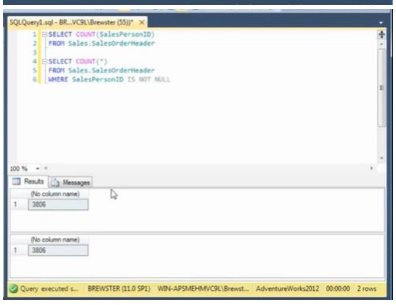


MIN

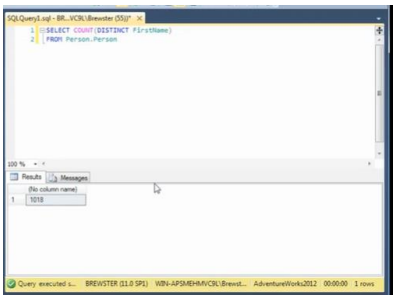


COUNT

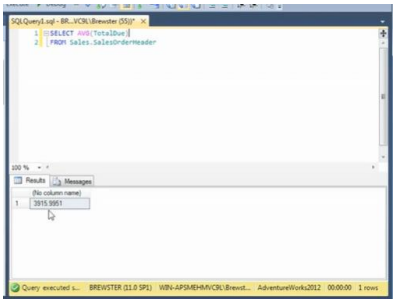




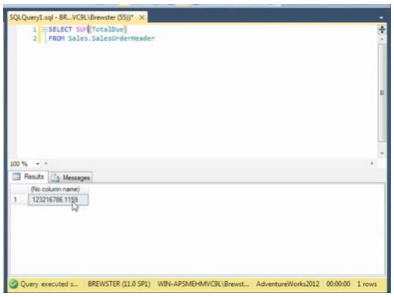
Use **COUNT** to get unique values

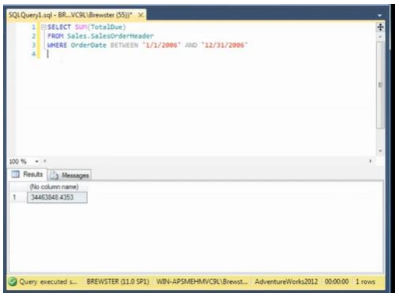


AVG



SUM



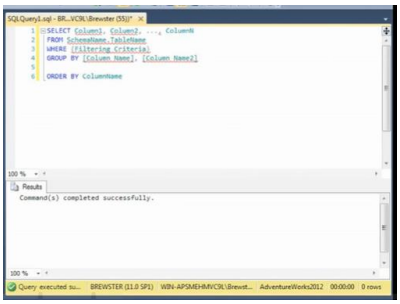


Grouping Data with the GROUP BY Clause

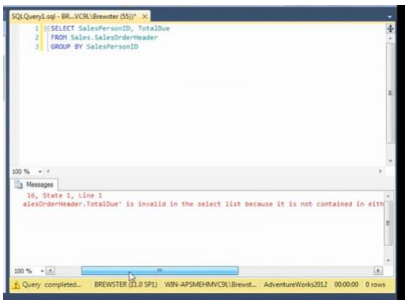
The **GROUP** BY clause is a SQL command that is used to **group** rows that have the same values. The **GROUP** BY clause is used in the SELECT statement.

Optionally it is used in conjunction with aggregate functions to produce summary reports from the database. That's what it does, summarizing data from the database

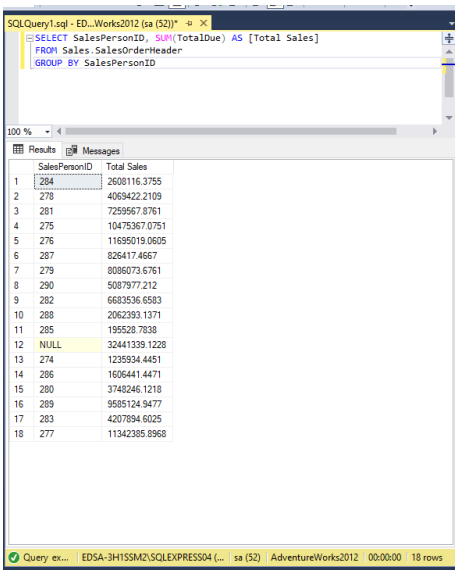
General form



DO NOT DO.

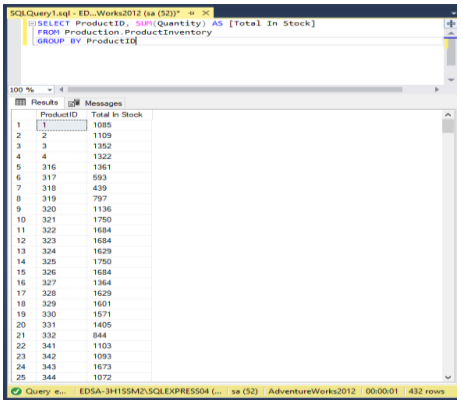


SQL requires that every column in the SELECT clause be included in either an aggregate function or in the GROUP BY clause.

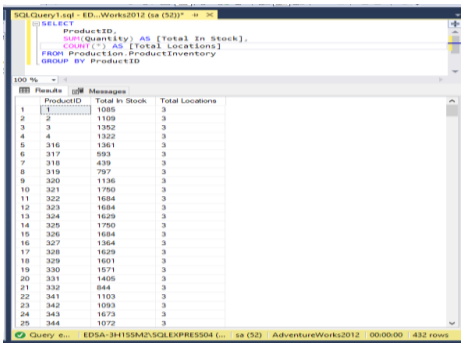


	SalesPersonID	Total Sales
1	284	2608116.3755
2	278	4069422.2109
3	281	7259567.8761
4	275	10475367.0751
5	276	11695019.0605
6	287	826417.4667
7	279	8086073.6761
8	290	5087977.212
9	282	6683536.6583
10	288	2062393.1371
11	285	195528.7838
12	NULL	32441339.1228
13	274	1235934.4451
14	286	1606441.4471
15	280	3748246.1218
16	289	9585124.9477
17	283	4207894.6025
18	277	11342385.8968

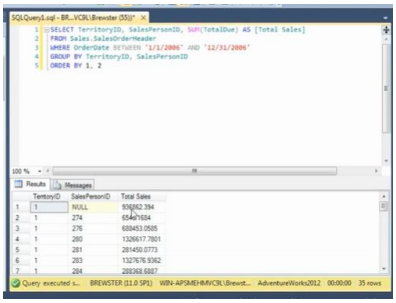
One aggregate function.



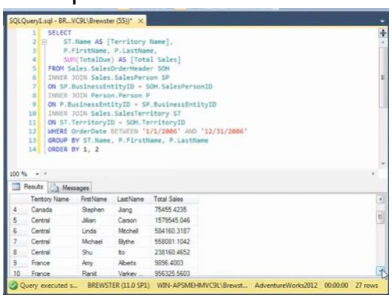
Multiple aggregate functions.



Multiple clauses



Example



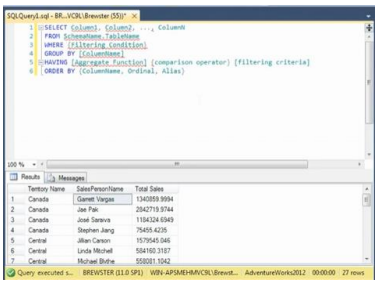
Filtering Groups with the HAVING Clause

A **HAVING** clause in **SQL** specifies that an **SQL** SELECT statement should only return rows where aggregate values meet the specified conditions.

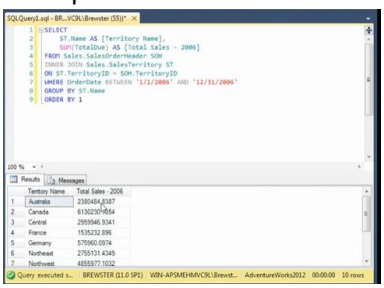
HAVING filters records that work on summarized GROUP BY results. **HAVING** applies to summarized group records, whereas WHERE applies to individual records.

Only the groups that meet the HAVING criteria will be returned.

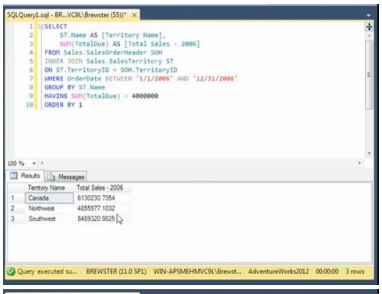
General form:

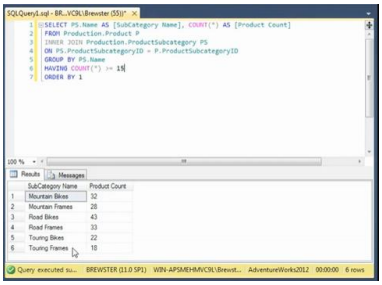


Example

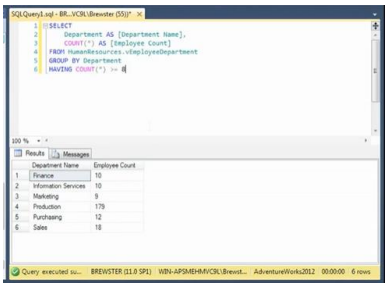


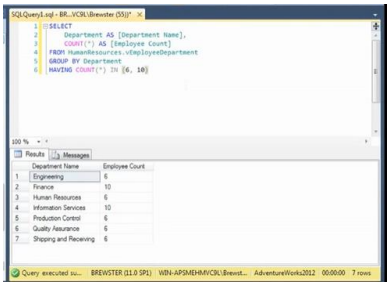
Use HAVING

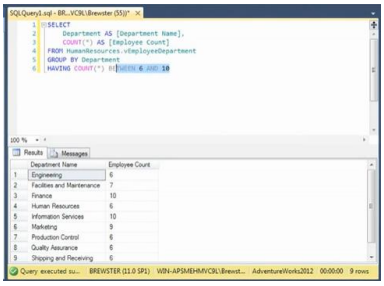




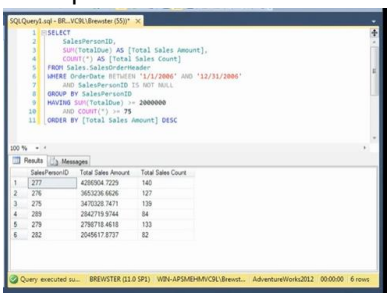
3 examples of the HAVING function with aggregate functions and operators.


SQLQuery1.sql - BR...VC9L\Brewster (55))*
SELECT
Department AS [Department Name],
COUNT(*) AS [Employee Count]
FROM HumanResources.vEmployeeDepartment
GROUP BY Department
HAVING COUNT(*) >= 6
100 %
Results
Messages
Department Name
Employee Count
1 Finance 10
2 Information Services 10
3 Marketing 9
4 Production 179
5 Purchasing 12
6 Sales 18
Query executed su... BREWSTER (11.0 SP1) WIN-APSMEHMVC9L\Brewst... AdventureWorks2012 00:00:00 6 rows


SQLQuery1.sql - BR...VC9L\Brewster (55))*
SELECT
Department AS [Department Name],
COUNT(*) AS [Employee Count]
FROM HumanResources.vEmployeeDepartment
GROUP BY Department
HAVING COUNT(*) IN (6, 10)
100 %
Results
Messages
Department Name
Employee Count
1 Engineering 6
2 Finance 10
3 Human Resources 6
4 Information Services 10
5 Production Control 6
6 Quality Assurance 6
7 Shipping and Receiving 6
Query executed su... BREWSTER (11.0 SP1) WIN-APSMEHMVC9L\Brewst... AdventureWorks2012 00:00:00 7 rows


SQLQuery1.sql - BR...VC9L\Brewster (55))*
SELECT
Department AS [Department Name],
COUNT(*) AS [Employee Count]
FROM HumanResources.vEmployeeDepartment
GROUP BY Department
HAVING COUNT(*) BETWEEN 6 AND 10
100 %
Results
Messages
Department Name
Employee Count
1 Engineering 6
2 Facilities and Maintenance 7
3 Finance 10
4 Human Resources 6
5 Information Services 10
6 Marketing 9
7 Production Control 6
8 Quality Assurance 6
9 Shipping and Receiving 6
Query executed su... BREWSTER (11.0 SP1) WIN-APSMEHMVC9L\Brewst... AdventureWorks2012 00:00:00 9 rows

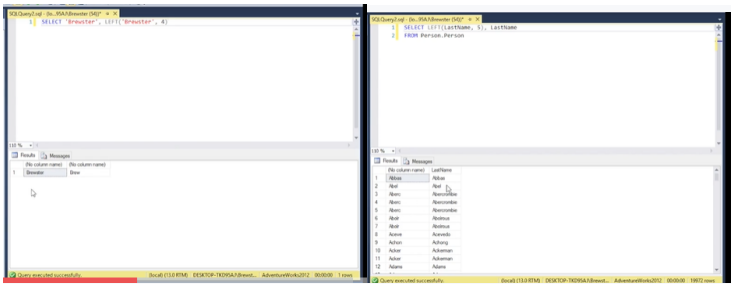
Example


SQLQuery1.sql - BR...VC9L\Brewster (55))*
SELECT
SalesPersonID,
SUM(TotalDue) AS [Total Sales Amount],
COUNT(*) AS [Total Sales Count]
FROM Sales.SalesOrderHeader
WHERE OrderDate BETWEEN '1/1/2006' AND '12/31/2006'
AND SalesPersonID IS NOT NULL
GROUP BY SalesPersonID
HAVING SUM(TotalDue) >= 2000000
AND COUNT(*) >= 75
ORDER BY [Total Sales Amount] DESC
100 %
Results
Messages
SalesPersonID
Total Sales Amount
Total Sales Count
1 277 4286904.7229 140
2 276 3653236.6626 127
3 275 3470328.7471 139
4 289 2842719.5744 84
5 279 2798718.4618 133
6 282 2045617.8737 82
Query executed su... BREWSTER (11.0 SP1) WIN-APSMEHMVC9L\Brewst... AdventureWorks2012 00:00:00 6 rows

Using SQL Server Functions

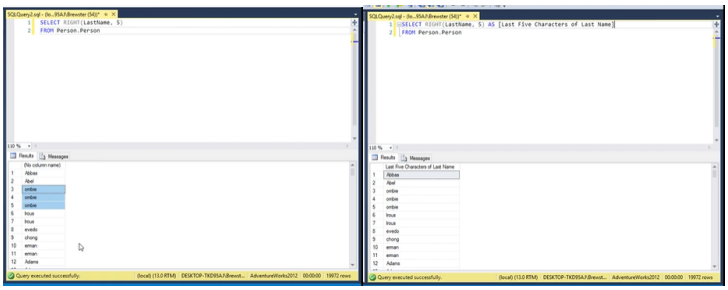
LEFT function

To pass a string and only return an x amount of the LEFT most characters within that string.



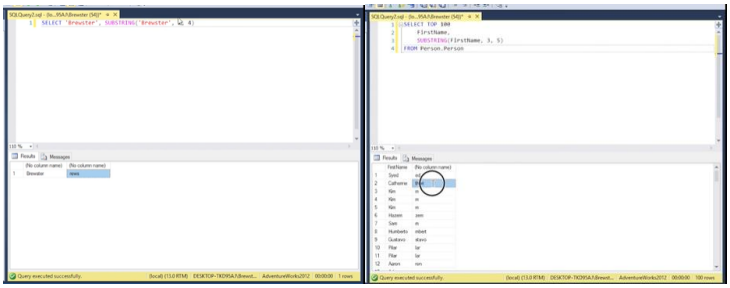
RIGHT function

To pass a string and only return an x amount of the RIGHT most characters within that string.



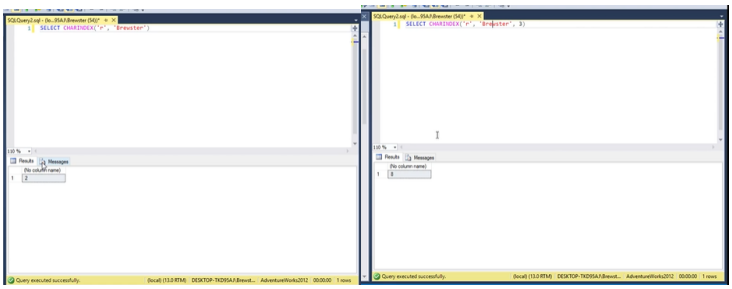
SUBSTRING function

It extracts a **substring** with a specified length starting from a location in an input string.
The length is a positive integer that specifies the number of characters of the **substring** to be returned.



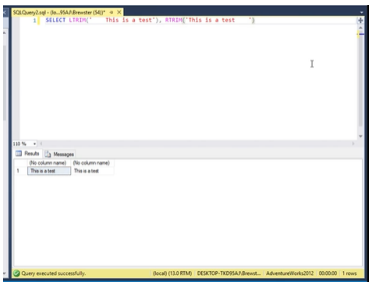
CHARINDEX function

The CHARINDEX() function searches for a substring in a string, and returns the position. If the substring is not found, this function returns 0.
It has a 3rd argument that specifies the starting position.



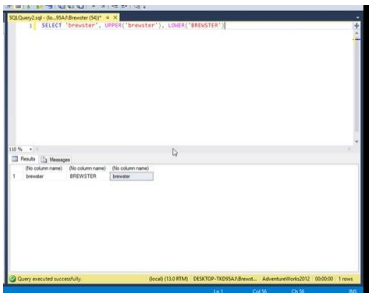
LTRIM function
The LTRIM function removes all space characters from the left-hand side of a string.

RTRIM function
The RTRIM function removes all space characters from the right-hand side of a string.

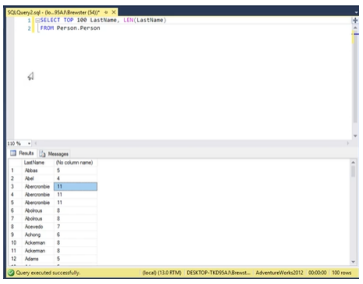


UPPER function
It is used to capitalize all characters in a string.

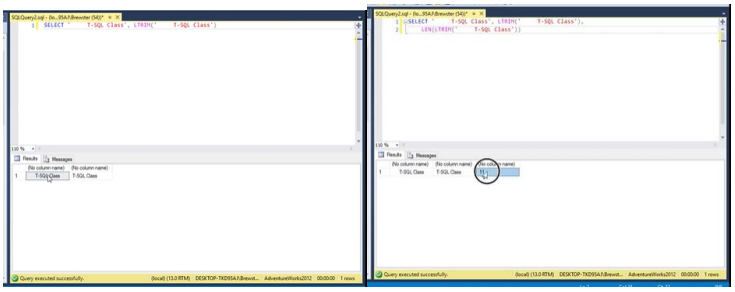
LOWER function
Converts a string to lower-case.



LEN function
Find the length of a string.

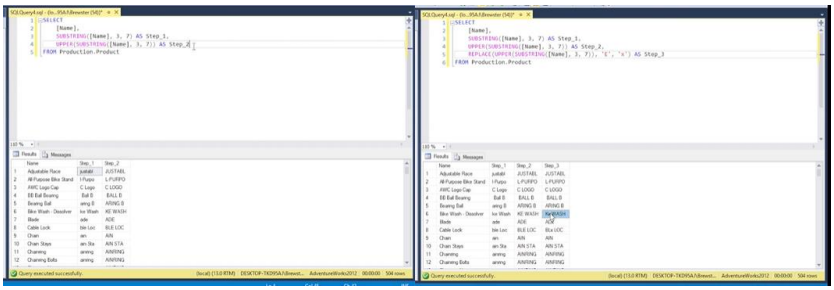


Nested functions



REPLACE function

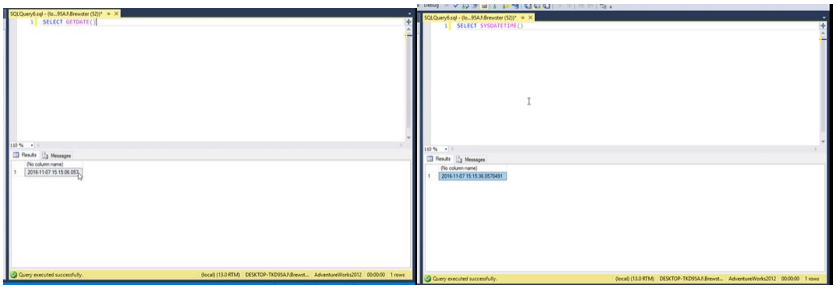
The **REPLACE**() function replaces all occurrences of a substring within a string, with a new substring. Note: The search is case-insensitive.



Date functions

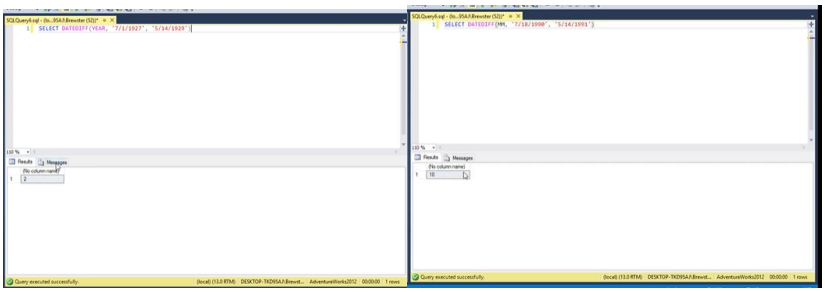
GETDATE function

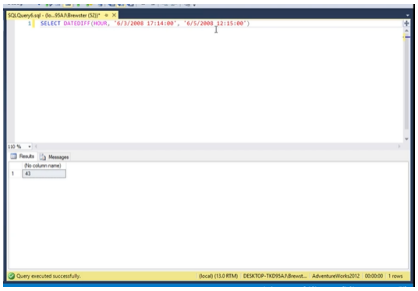
It gives the current date. (use **SYSDATETIME()** for precision)



DATEDIFF function

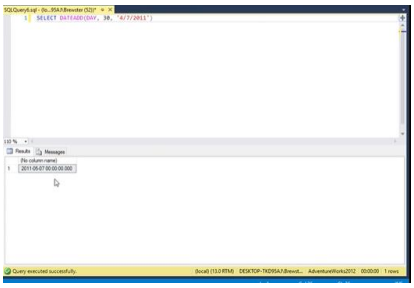
Returns the difference between two dates.

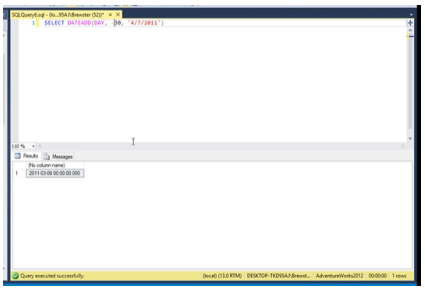




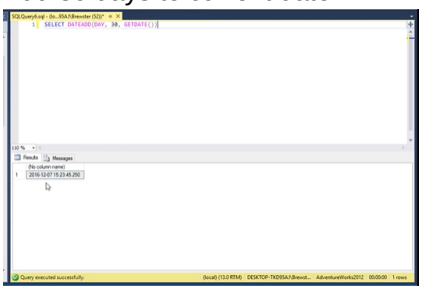
DATEADD function

To add a specified value to a specified date part of a date.





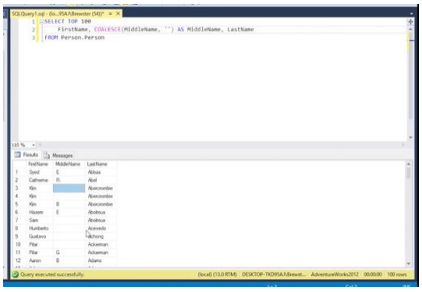
Add 30 days to current date



NULL handling functions

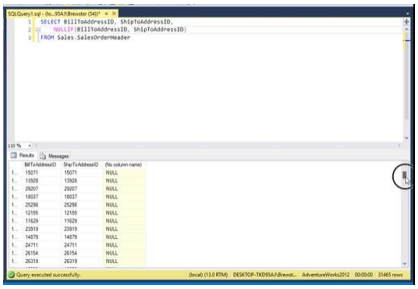
COALESCE function

Take any cell with NULL and convert it to a blank space.

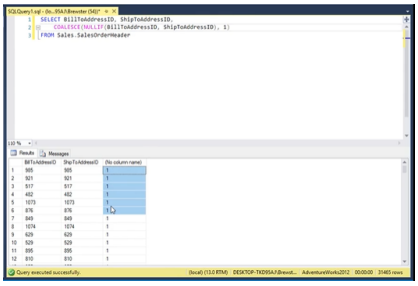


NULLIF function

Returns **NULL if** two expressions are equal, otherwise it returns the first expression.



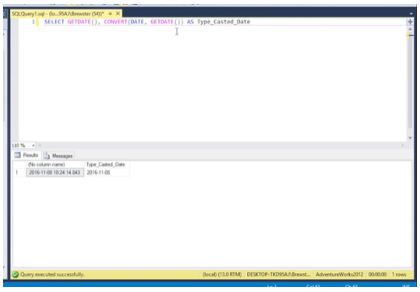
COALESCE and NULLIF



SQL Server Data Types and Type Casting

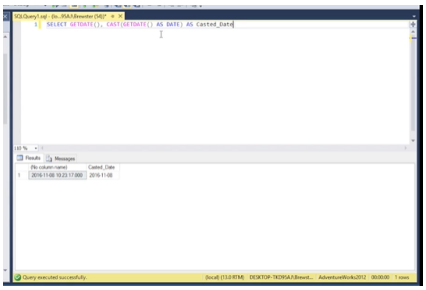
CONVERT

The **CONVERT** function **converts** an expression from one datatype to another datatype.
Can be one of the following: bigint, int, smallint, tinyint, bit, decimal, numeric, money, float etc.



CAST

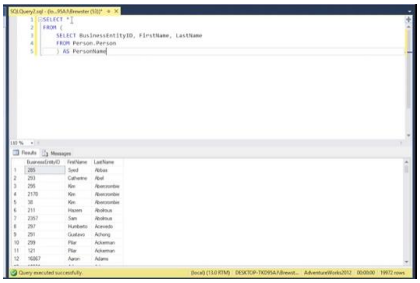
The CAST function in SQL converts data from one data type to another, such as from numeric data into character string data.



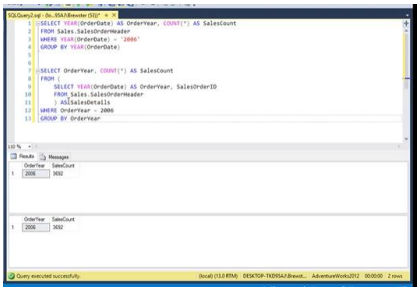
Working with Table Expressions

Derived table

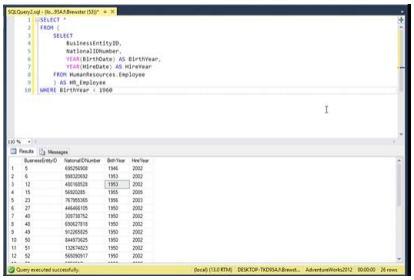
The **SQL Derived Table** is nothing but a Subquery used in the From Clause.
It has to have a name.

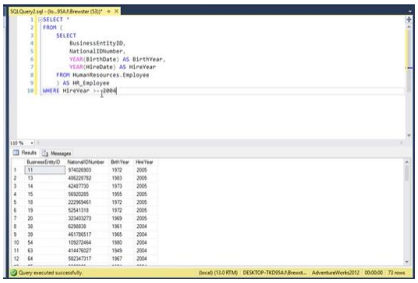


Example to reduce redundancy



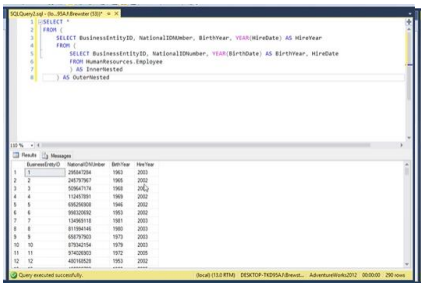
Example

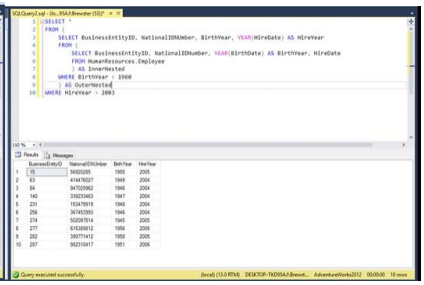




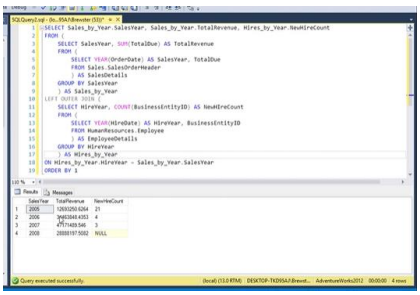
Nested derived tables

These work inside out.

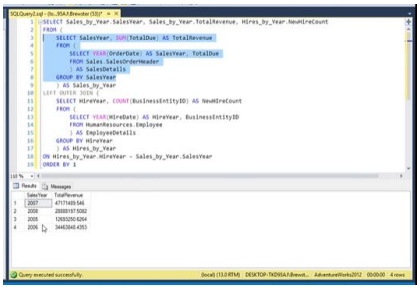




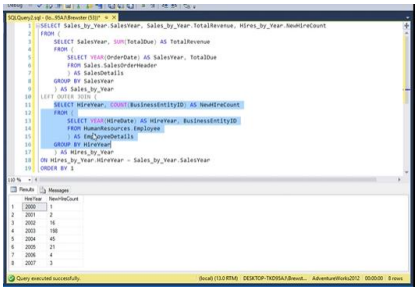
Using derived tables with JOINS



Part One



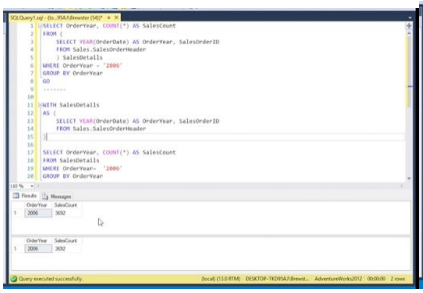
Part Two

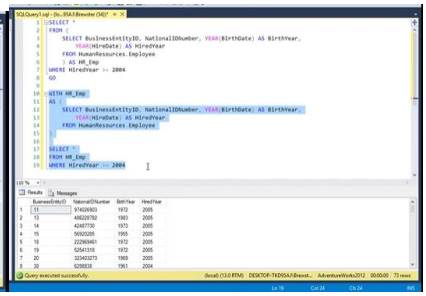


Working with CTEs (Common Table Expression)

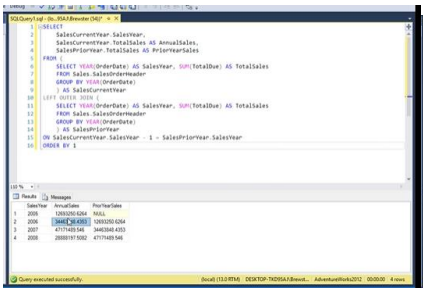
A **CTE** (Common Table Expression) is a temporary result set that you can reference within another SELECT, INSERT, UPDATE, or DELETE statement.

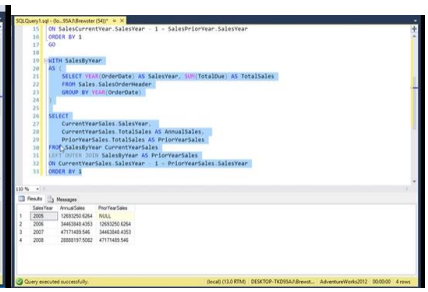
A derived table vs. CTE



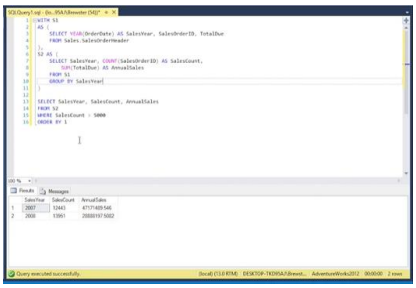


To reduce redundancy using a CTE after a derived table.





Nested CTEs

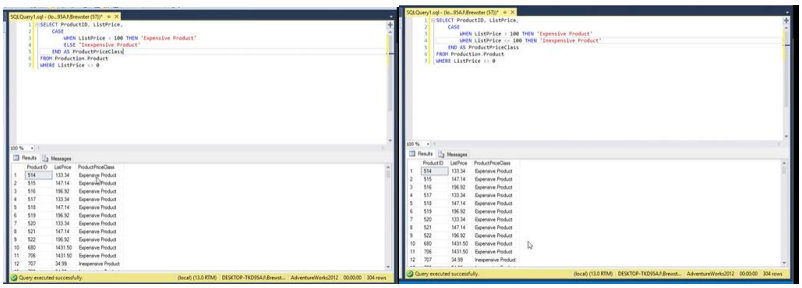


CASE statements

The CASE statement goes through conditions and returns a value when the first condition is met (like an IF-THEN-ELSE statement).

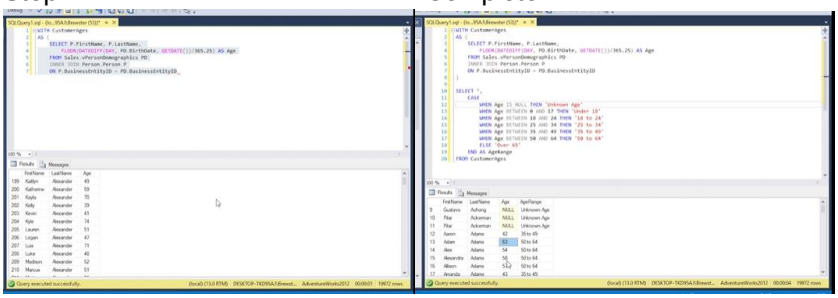
So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause.

If there is no ELSE part and no conditions are true, it returns NULL.



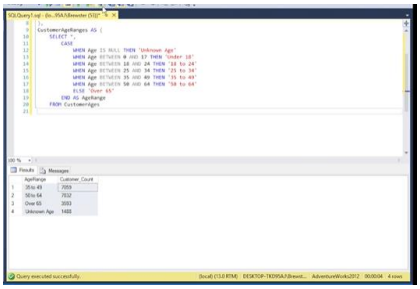
Example
FLOOR function
Returns an integer after removing the decimal point.

Step 1 Complete

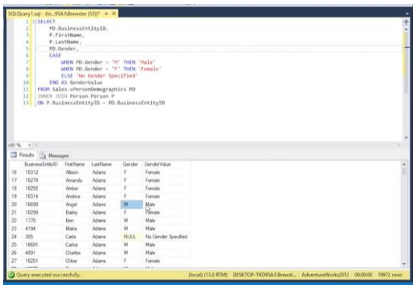


Group by age and count.

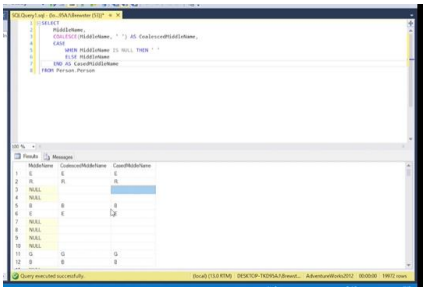
Place the last block in a second CTE.



Example where you convert ‘M’ or ‘F’ to ‘Male’ or ‘Female’.

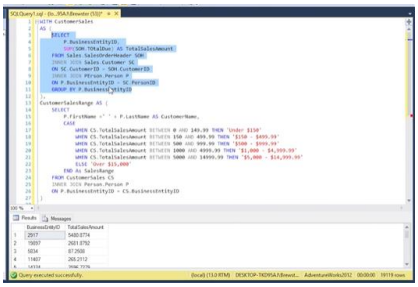


Taking advantage of NULL handling functions

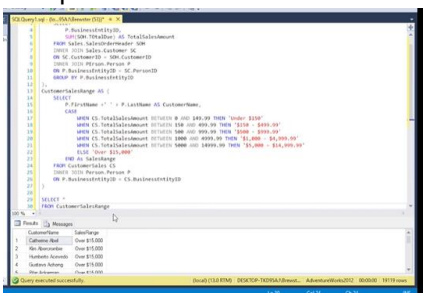


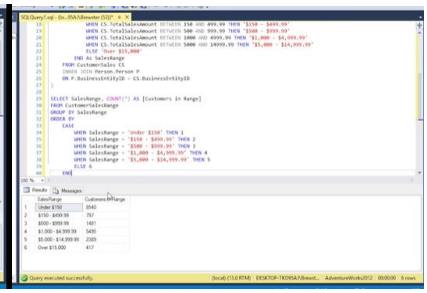
Example used with an ORDER BY clause

Part One



Complete



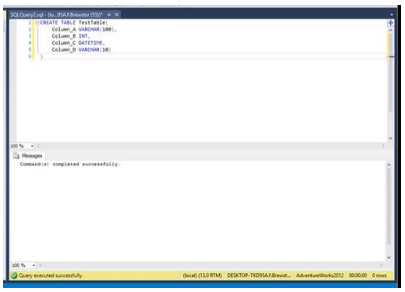


Creating Tables and Inserting/Updating Data

CREATE TABLE

Creating a basic **table** involves naming the **table** and defining its columns and each column's data type.
The **SQL CREATE TABLE** statement is used to **create** a new **table**.

General form

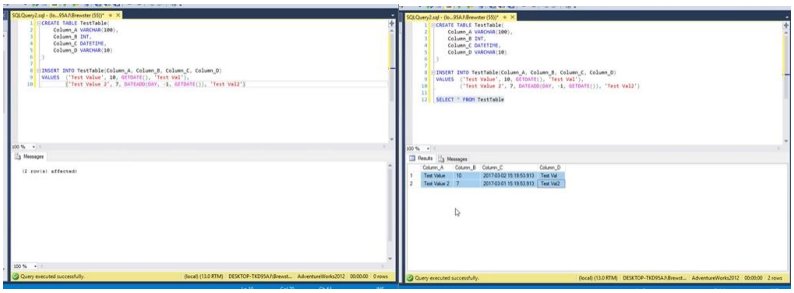


INSERT INTO

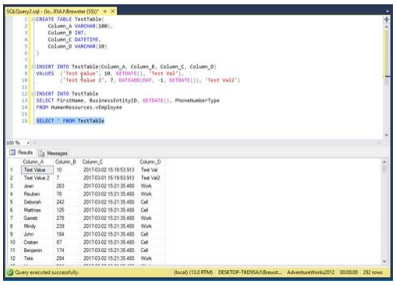
The INSERT INTO statement is used to add new data to a database.
The INSERT INTO statement adds a new record to a table.

Can contain values for some or all of its columns.
Can be combined with a SELECT to insert records.

Manually inserting data.



Use SELECT statement

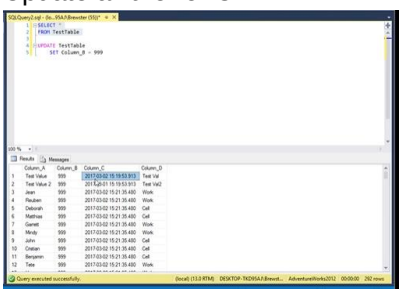


UPDATE

An **SQL UPDATE** statement changes the data of one or more records in a table.
Either all the rows can be **updated**, or a subset may be chosen using a condition.

Query is used to modify the existing records in a table.
You can use the WHERE clause with the **UPDATE** query to **update** the selected rows, otherwise all the rows would be affected.

Update all the rows



Update a row