Import a Sample Database

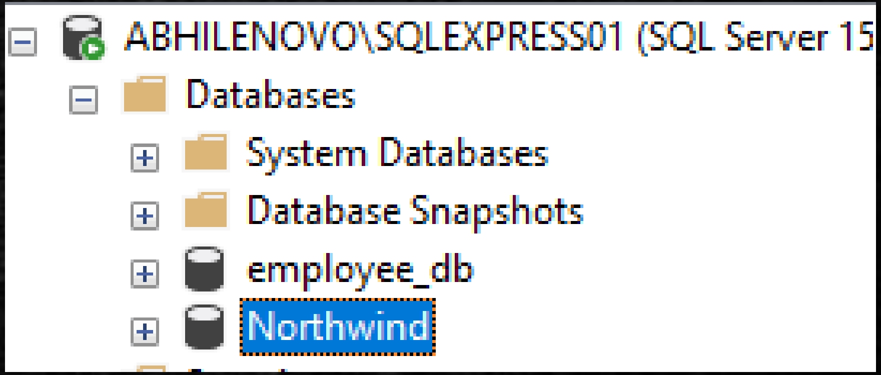
Import the Northwind database from Microsoft

Northwind and pubs sample databases for Microsoft SQL Server

The Northwind and Pubs databases are available for free by Microsoft and can be downloaded and used in any SQL Server

https://github.com/microsoft/sql-server-samples/tree/master/samples/databases/northwind-pubs

View the Raw SQL of northwind, copy it to query window and run it.


```
⊟ ABHILENOVO\SQLEXPRESS01 (SQL Server 15
   ⊟ Databases
      ⊞ System Databases
      ⊞ Database Snapshots
      ⊞ employee_db
      ⊞ Northwind
```

SQL Basic Aggregate Functions

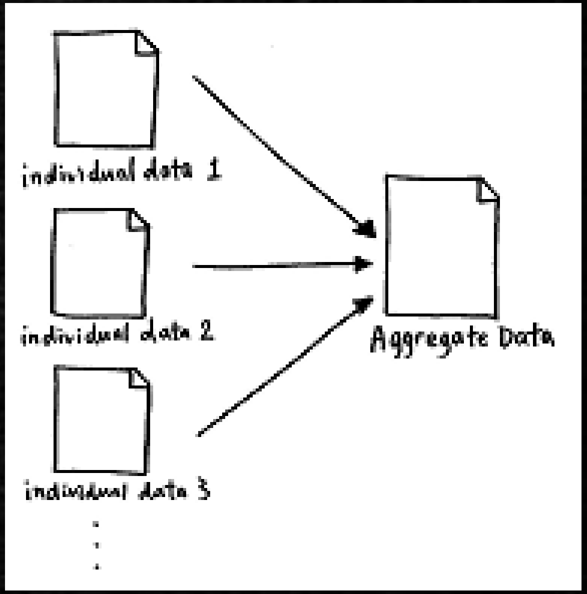
Basic Aggregate Operations: MIN, MAX, SUM, AVG, COUNT

What are Aggregate Functions in SQL?

An aggregate function allows you to perform a calculation on a set of values to return a single scalar value.

The most commonly used SQL aggregate functions:

MIN – gets the minimum value in a set of values.
MAX – gets the maximum value in a set of values.
SUM – calculates the sum of values.
AVG – calculates the average of a set of values.
COUNT – counts rows in a specified table or view.



SQL MIN Aggregate Function

MIN is used to get the minimum or smallest value of a specified column or expression.

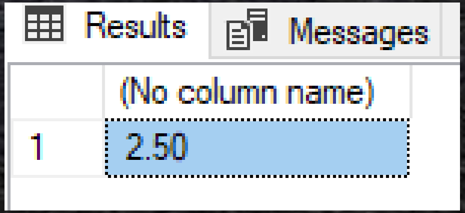
MIN ignores NULL values from the table.

The Syntax is

```
SELECT MIN column(s)
FROM table_name(s)
[WHERE conditions];
```

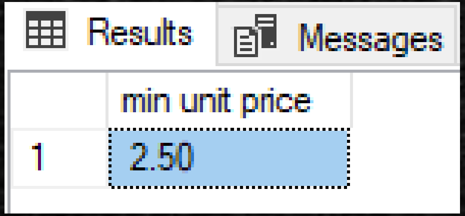
SQL MIN Aggregate Function Examples

```
SELECT
    MIN(unitprice)
FROM
 Northwind.dbo.products;
```



```
    (No column name)
1   2.50
```

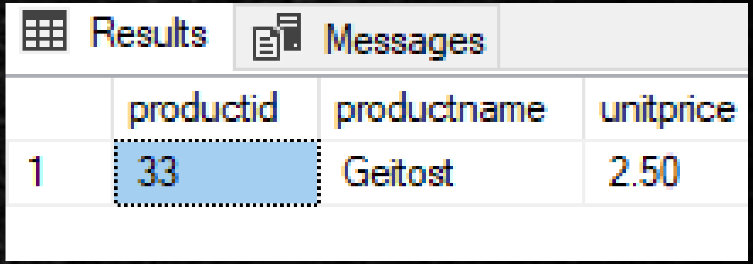
```
SELECT
    MIN(unitprice) AS 'min unit price'
FROM
    products;
```



```
    min unit price
1   2.50
```

SQL MIN Aggregate Function Examples

```
-- Using a subquery that uses the MIN() function

SELECT
    productid, productname, unitprice
FROM
    products
WHERE
    unitprice = (SELECT MIN(unitprice) FROM products);

--Will be equal to

SELECT
    productid, productname, unitprice
FROM
    products
WHERE
    unitprice = 2.50;
```



	productid	productname	unitprice
1	33	Geitost	2.50

SQL MAX Aggregate Function

MAX is used to get the maximum or largest value of a specified column or expression.

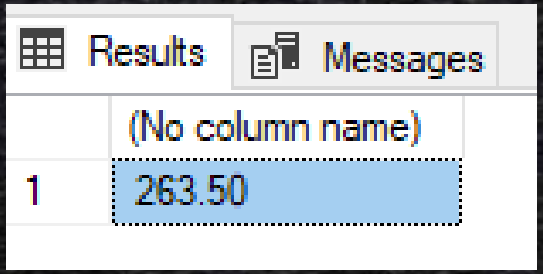
MIN ignores NULL values from the table.

The Syntax is

```
SELECT MAX column(s)
FROM table_name(s)
[WHERE conditions];
```

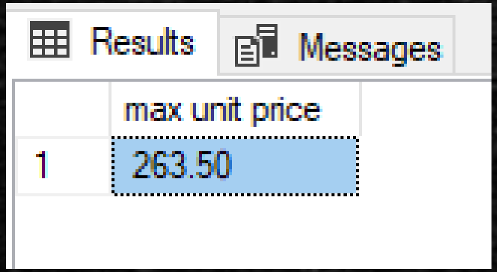
SQL MAX Aggregate Function Examples

```
SELECT
    MAX(unitprice)
FROM
    products;
```



```
   (No column name)
1  263.50
```

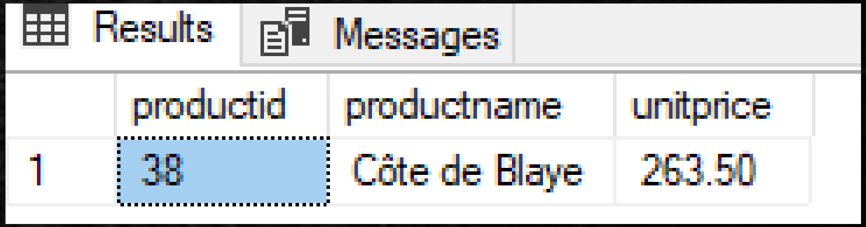
```
SELECT
    MAX(unitprice) AS 'max unit price'
FROM
    products;
```



```
   max unit price
1  263.50
```

SQL MAX Aggregate Function Examples

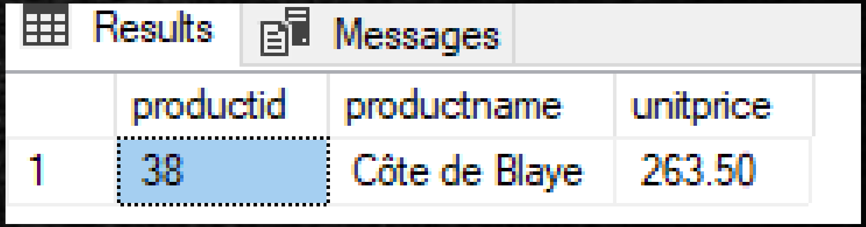
```
-- Using a subquery that uses the MAX() function

SELECT
    productid, productname, unitprice
FROM
    products
WHERE
    unitprice = (SELECT MAX(unitprice) FROM products);

--Will be equal to

SELECT
    productid, productname, unitprice
FROM
    products
WHERE
    unitprice = 263.50;
```



	productid	productname	unitprice
1	38	Côte de Blaye	263.50

SQL MAX Aggregate Function Examples

```
-- Using a subquery that uses the MAX() function

SELECT
    productid, productname, unitprice
FROM
    products
WHERE
    unitprice = (SELECT MAX(unitprice) FROM products);

--Will be equal to

SELECT
    productid, productname, unitprice
FROM
    products
WHERE
    unitprice = 263.50;
```



	productid	productname	unitprice
1	38	Côte de Blaye	263.50

SQL AVG Aggregate Function

AVG is used to get the average value of a specified column or expression.

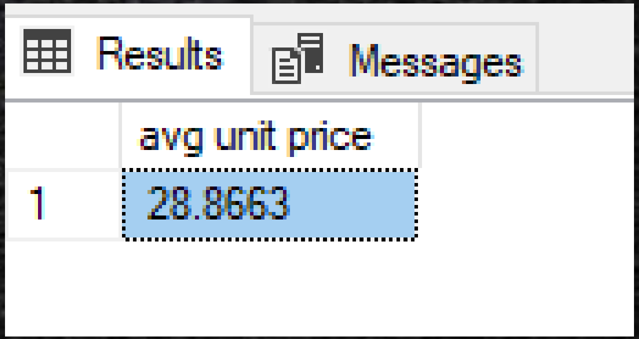
AVG ignores NULL values from the table.

The Syntax is

```
SELECT AVG column(s)
FROM table_name(s)
[WHERE conditions];
```

SQL AVG Aggregate Function Examples

```
SELECT
    AVG(unitprice)
FROM
    products;
```

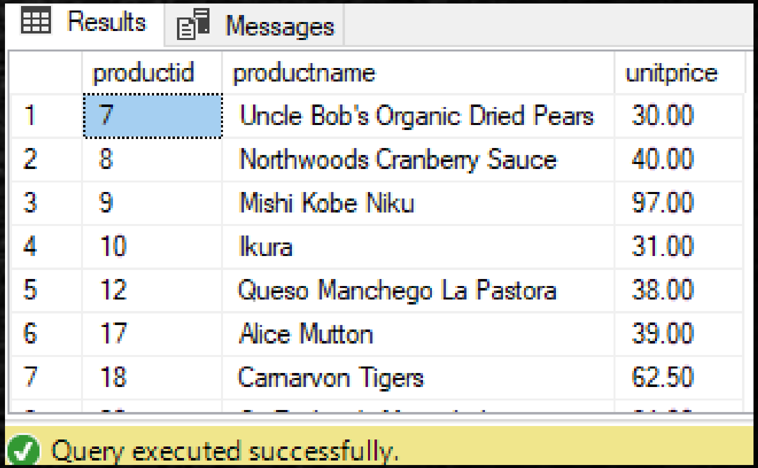


	avg unit price
1	28.8663

```
SELECT
    AVG(unitprice) AS ‘avg unit price'
FROM
    products;
```

SQL AVG Aggregate Function Examples

```
-- Using a subquery that uses the AVG() function

SELECT
    productid, productname, unitprice
FROM
    products
WHERE
    unitprice > (SELECT AVG(unitprice) FROM products);

--Will be equal to

SELECT
    productid, productname, unitprice
FROM
    products
WHERE
    unitprice > 28.8663;
```



	productid	productname	unitprice
1	7	Uncle Bob's Organic Dried Pears	30.00
2	8	Northwoods Cranberry Sauce	40.00
3	9	Mishi Kobe Niku	97.00
4	10	Ikura	31.00
5	12	Queso Manchego La Pastora	38.00
6	17	Alice Mutton	39.00
7	18	Carnarvon Tigers	62.50

Query executed successfully.

SQL SUM Aggregate Function

SUM is used to get the total value of a specified column or expression.

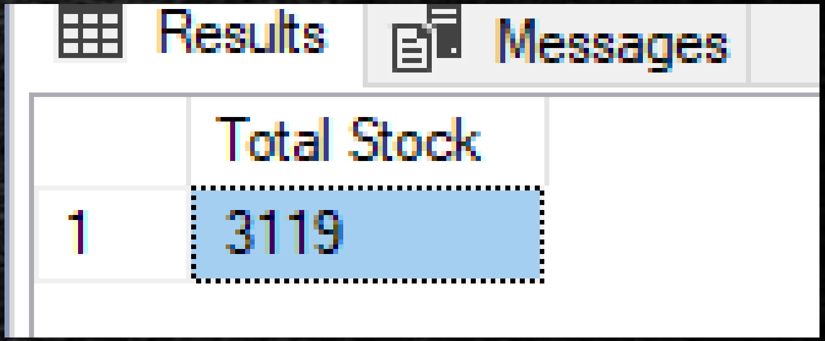
SUM ignores NULL values from the table.

The Syntax is

```
SELECT SUM column
FROM table_name
[WHERE conditions];
```

SQL SUM Aggregate Function Examples

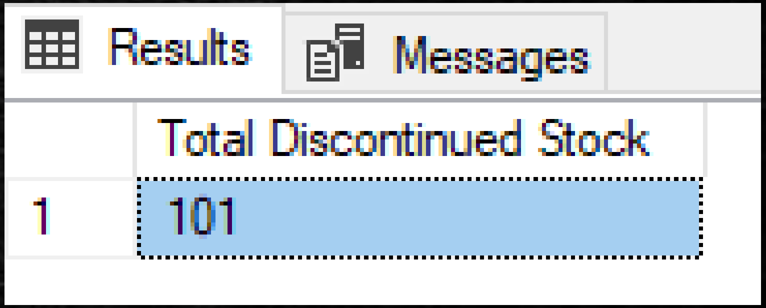
```
SELECT
      SUM(UnitsInStock) AS 'Total Stock'
FROM
      products
```



Results | Messages

	Total Stock
1	3119

```
SELECT
      SUM(UnitsInStock) AS 'Total Discontinued Stock'
FROM
      products
WHERE
      Discontinued = 1
```



Results | Messages

	Total Discontinued Stock
1	101

SQL COUNT Aggregate Function

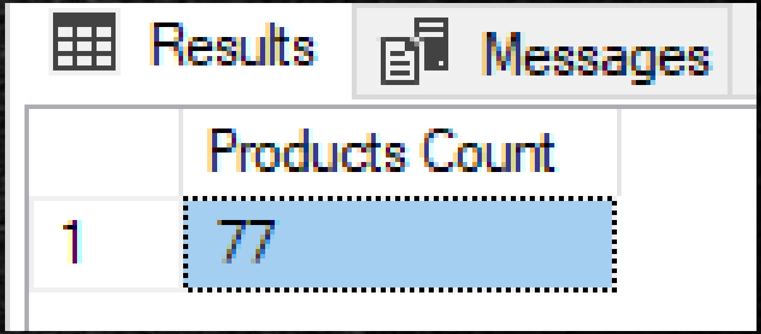
COUNT is used for calculating the total number of rows present in the table.

The Syntax is

```
SELECT COUNT column
FROM table_name
[WHERE conditions];
```

SQL COUNT Aggregate Function Examples

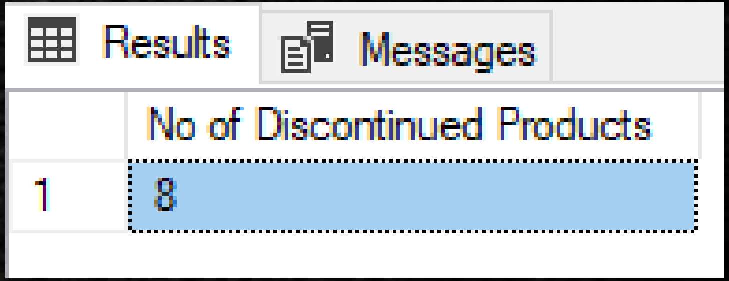
```
SELECT
      COUNT(ProductID) AS 'Products Count'
FROM
      products
```



Results | Messages

	Products Count
1	77

```
SELECT
      COUNT(ProductID) AS 'No of Discontinued Products'
FROM
      products
WHERE
      Discontinued = 1
```



Results | Messages

	No of Discontinued Products
1	8

SQL Server - Basic Clauses

Basic Clauses : DISTINCT, GROUP BY, WHERE, ORDER BY, HAVING, SELECT, GROUPING SETS

What are clauses in SQL?

A clause is just a logical part of an SQL statement

The most commonly used SQL Clauses are:

- DISTINCT
- GROUP BY
- WHERE
- ORDER BY
- HAVING
- SELECT
- GROUPING SETS

DISTINCT Clause

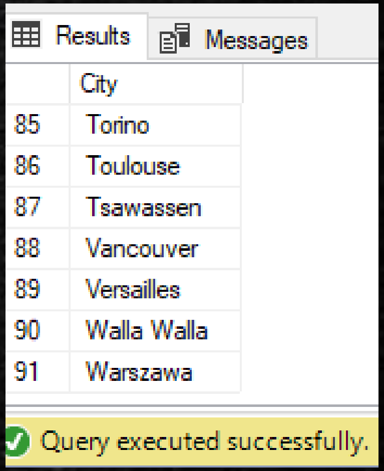
- The result set of a SELECT statement may contain duplicate rows.
- To eliminate the duplicates, use the DISTINCT operator
- We can use the DISTINCT operator in the SELECT statement only.

The syntax is:

```
SELECT DISTINCT column(s)
FROM table_name;
```

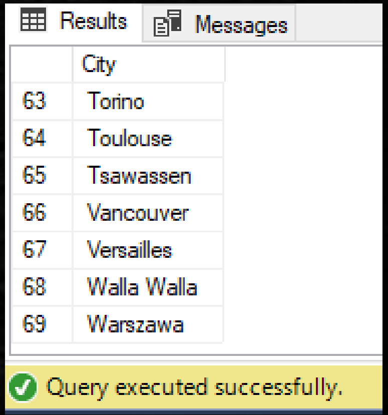
DISTINCT Clause Examples

```
SELECT City FROM Northwind.dbo.Customers

SELECT DISTINCT City FROM Customers

SELECT DISTINCT City, Region FROM Customers

SELECT DISTINCT City, Region FROM Customers
WHERE Country='UK'
```



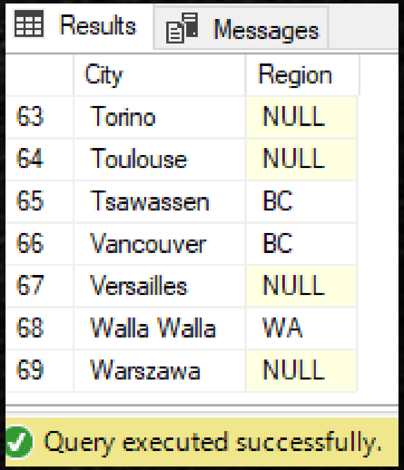
	City
85	Torino
86	Toulouse
87	Tsawassen
88	Vancouver
89	Versailles
90	Walla Walla
91	Warszawa

Query executed successfully.



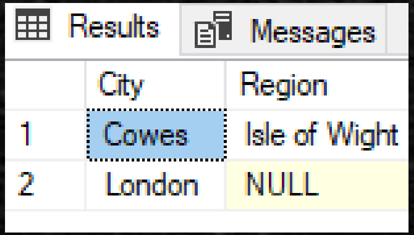
	City
63	Torino
64	Toulouse
65	Tsawassen
66	Vancouver
67	Versailles
68	Walla Walla
69	Warszawa

Query executed successfully.



	City	Region
63	Torino	NULL
64	Toulouse	NULL
65	Tsawassen	BC
66	Vancouver	BC
67	Versailles	NULL
68	Walla Walla	WA
69	Warszawa	NULL

Query executed successfully.



	City	Region
1	Cowes	Isle of Wight
2	London	NULL

GROUP BY Clause

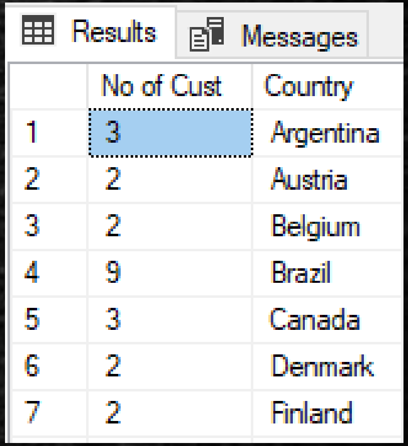
- GROUP BY statement groups rows that have the same values into temporary summary rows
- It is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG())

The syntax is:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

GROUP BY Clause Examples

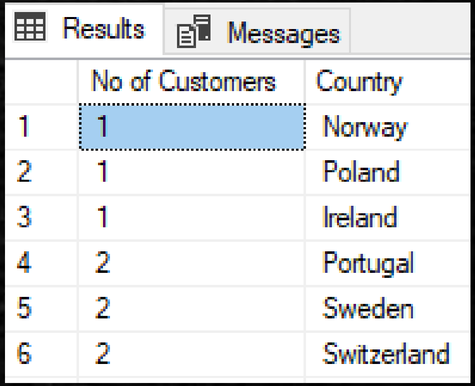
```
SELECT COUNT(CustomerID) AS 'No of Customers', Country
FROM Customers
GROUP BY Country;
```



Results | Messages

	No of Cust	Country
1	3	Argentina
2	2	Austria
3	2	Belgium
4	9	Brazil
5	3	Canada
6	2	Denmark
7	2	Finland

```
SELECT COUNT(CustomerID) AS 'No of Customers', Country
FROM Customers
GROUP BY Country;
ORDER BY COUNT(CustomerID)
```



Results | Messages

	No of Customers	Country
1	1	Norway
2	1	Poland
3	1	Ireland
4	2	Portugal
5	2	Sweden
6	2	Switzerland

WHERE Clause

- The WHERE clause in SQL Server is used to filter records from the table.

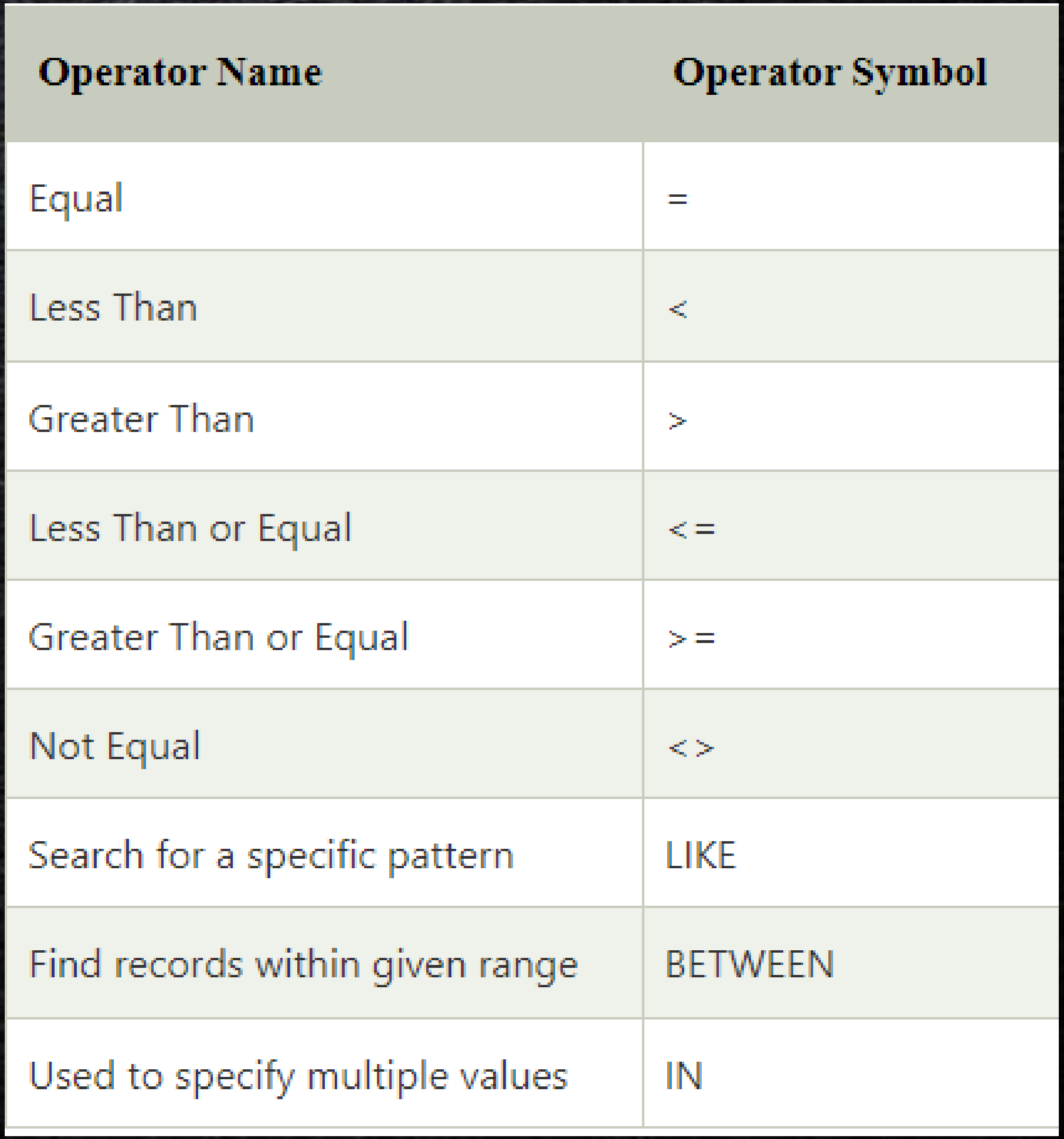
- Often used with SELECT, the WHERE clause can also work with the UPDATE and DELETE query.

The syntax is:

```
SELECT column_name(s)
FROM table_name
WHERE condition;
```

WHERE Clause Operators

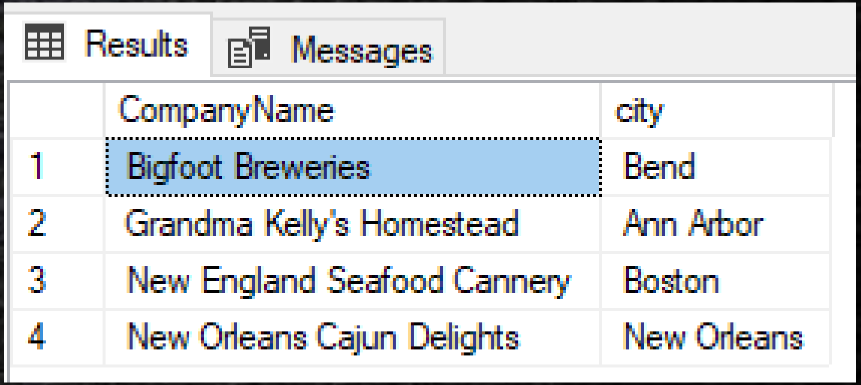
- The WHERE clause also supports these operators to filter the records:



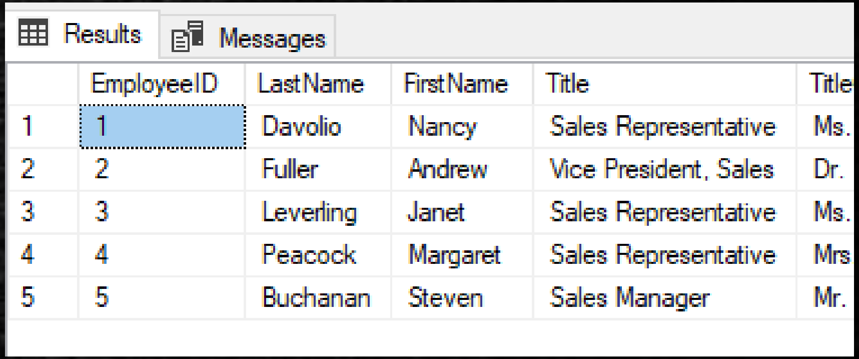
Operator Name	Operator Symbol
Equal	=
Less Than	<
Greater Than	>
Less Than or Equal	<=
Greater Than or Equal	>=
Not Equal	<>
Search for a specific pattern	LIKE
Find records within given range	BETWEEN
Used to specify multiple values	IN

WHERE Clause Examples

```
--Using = operator
-- For string compare use ‘’
SELECT CompanyName, city
  FROM Suppliers
  WHERE Country = 'USA'
  ORDER BY CompanyName;

--Using BETWEEN operator
SELECT * FROM Employees
WHERE EmployeeID BETWEEN 1 AND 5
```


Results | Messages

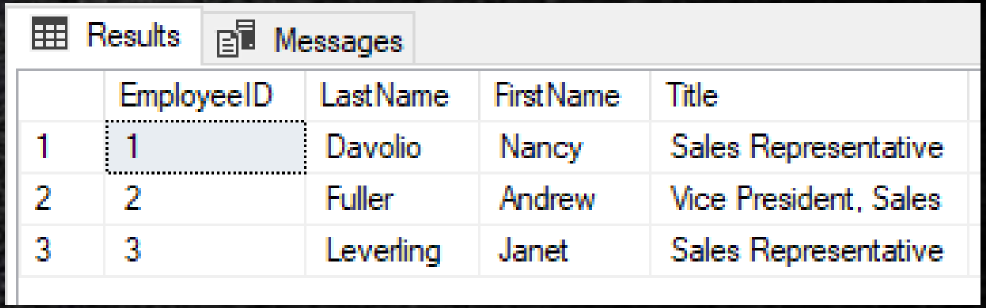
	CompanyName	city
1	Bigfoot Breweries	Bend
2	Grandma Kelly's Homestead	Ann Arbor
3	New England Seafood Cannery	Boston
4	New Orleans Cajun Delights	New Orleans


Results | Messages

	EmployeeID	LastName	FirstName	Title	Title
1	1	Davolio	Nancy	Sales Representative	Ms.
2	2	Fuller	Andrew	Vice President, Sales	Dr.
3	3	Leverling	Janet	Sales Representative	Ms.
4	4	Peacock	Margaret	Sales Representative	Mrs
5	5	Buchanan	Steven	Sales Manager	Mr.

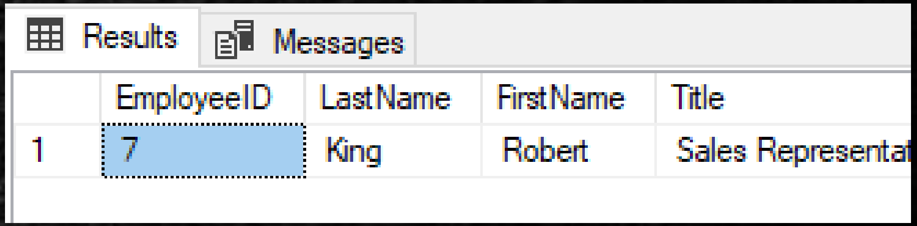
WHERE Clause Examples

```
--Using IN operator
SELECT * FROM Employees
WHERE EmployeeID IN (1,2,3)
```



	EmployeeID	LastName	FirstName	Title
1	1	Davolio	Nancy	Sales Representative
2	2	Fuller	Andrew	Vice President, Sales
3	3	Leverling	Janet	Sales Representative

```
--Using LIKE operator
SELECT * FROM Employees
WHERE FirstName Like 'Robert'
```



	EmployeeID	LastName	FirstName	Title
1	7	King	Robert	Sales Representat

ORDER BY Clause

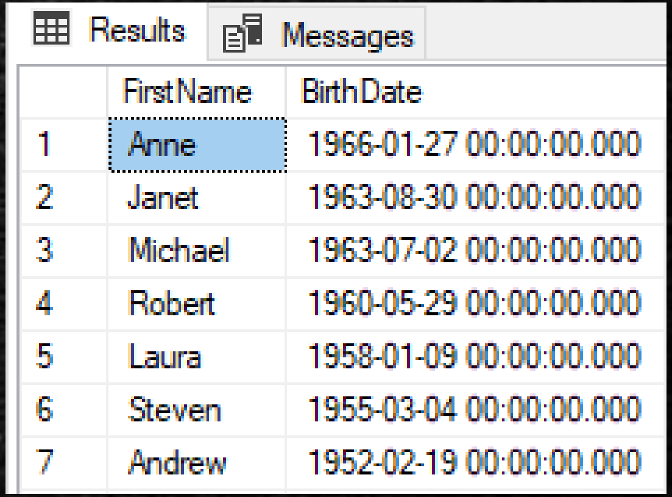
- Used to arrange the table's data in ascending or descending order based on the given column or list of columns.
-
- Often used with SELECT

The syntax is:

```
SELECT column_name(s)
FROM table_name
WHERE conditions
ORDER BY column_name [ASC | DESC];
```

ORDER BY Clause Examples

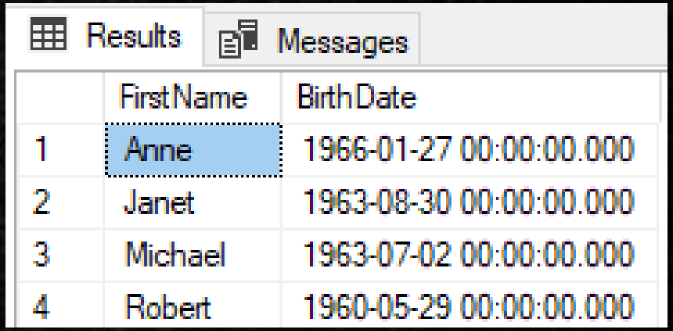
```
SELECT FirstName, BirthDate FROM Employees
ORDER BY BirthDate DESC
```



Results | Messages

	FirstName	BirthDate
1	Anne	1966-01-27 00:00:00.000
2	Janet	1963-08-30 00:00:00.000
3	Michael	1963-07-02 00:00:00.000
4	Robert	1960-05-29 00:00:00.000
5	Laura	1958-01-09 00:00:00.000
6	Steven	1955-03-04 00:00:00.000
7	Andrew	1952-02-19 00:00:00.000

```
--First sort by BD, then by First name
SELECT FirstName, BirthDate FROM Employees
ORDER BY BirthDate DESC,
FirstName ASC;
```



Results | Messages

	FirstName	BirthDate
1	Anne	1966-01-27 00:00:00.000
2	Janet	1963-08-30 00:00:00.000
3	Michael	1963-07-02 00:00:00.000
4	Robert	1960-05-29 00:00:00.000

HAVING Clause

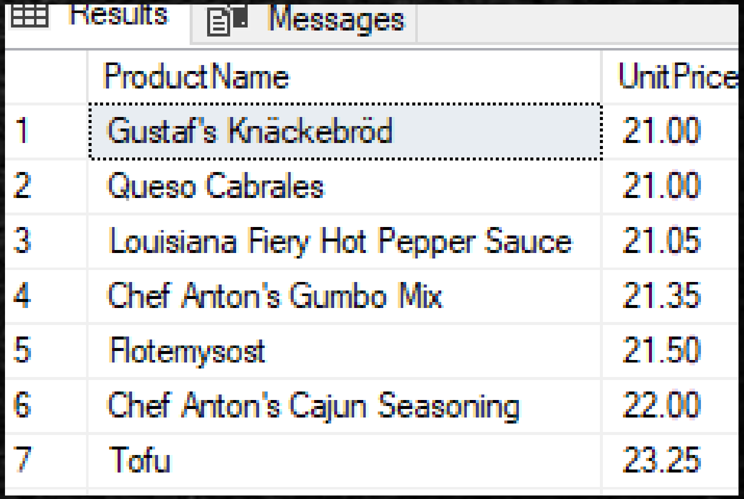
- The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

The syntax is:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

HAVING Clause Examples

```
SELECT ProductName,UnitPrice FROM Products
GROUP BY ProductName, UnitPrice
HAVING AVG(UnitPrice)>20
```



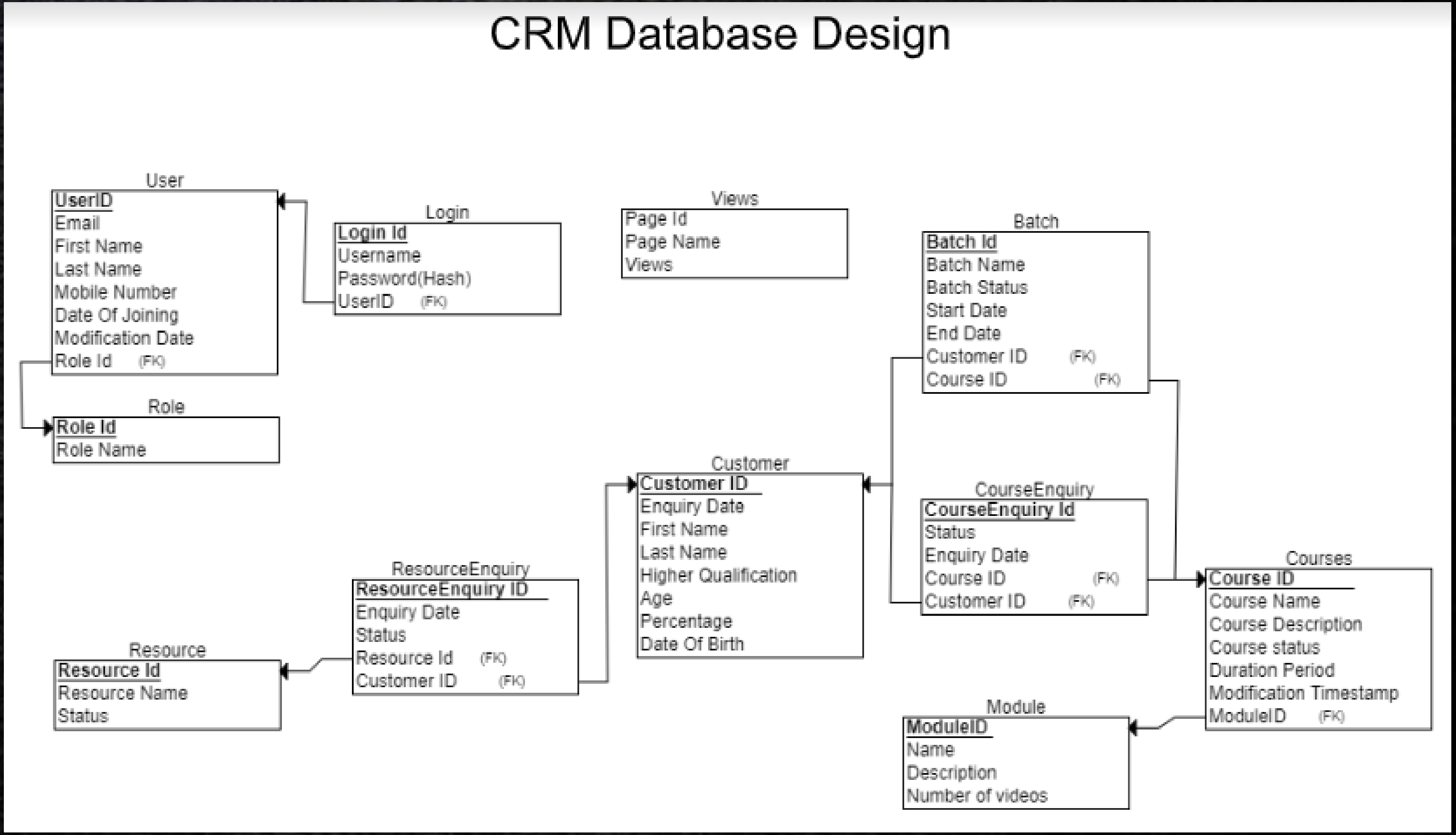
	ProductName	UnitPrice
1	Gustaf's Knäckebröd	21.00
2	Queso Cabrales	21.00
3	Louisiana Fiery Hot Pepper Sauce	21.05
4	Chef Anton's Gumbo Mix	21.35
5	Flotemysost	21.50
6	Chef Anton's Cajun Seasoning	22.00
7	Tofu	23.25

ASSIGNMENT 1

Prepare a database with schema like this:

Fill it with some meaningful data

Try the aggregate and clause queries that we tried today


CRM Database Design
User
UserID
Email
First Name
Last Name
Mobile Number
Date Of Joining
Modification Date
Role Id (FK)
Login
Login Id
Username
Password(Hash)
UserID (FK)
Views
Page Id
Page Name
Views
Batch
Batch Id
Batch Name
Batch Status
Start Date
End Date
Customer ID (FK)
Course ID (FK)
Role
Role Id
Role Name
Customer
Customer ID
Enquiry Date
First Name
Last Name
Higher Qualification
Age
Percentage
Date Of Birth
CourseEnquiry
CourseEnquiry Id
Status
Enquiry Date
Course ID (FK)
Customer ID (FK)
ResourceEnquiry
ResourceEnquiry ID
Enquiry Date
Status
Resource Id (FK)
Customer ID (FK)
Courses
Course ID
Course Name
Course Description
Course status
Duration Period
Modification Timestamp
ModuleID (FK)
Resource
Resource Id
Resource Name
Status
Module
ModuleID
Name
Description
Number of videos