

Lesson 2: SQL Basics (Part C)

December 27, 2016 © We Can Code IT. All Rights Reserved.

Audience

- Those who are new to relational DBs and SQL.

Objective

To understand basic ***SQL statements***.

To create simple ***Queries: SELECT, DISTINCT, Sorting & Ordering***.

To go beyond simple columns: ***Concatenation, Mathematical, Dates, Null***.

To understand how to ***filter data using the WHERE clause.***

Software & Resources Needed

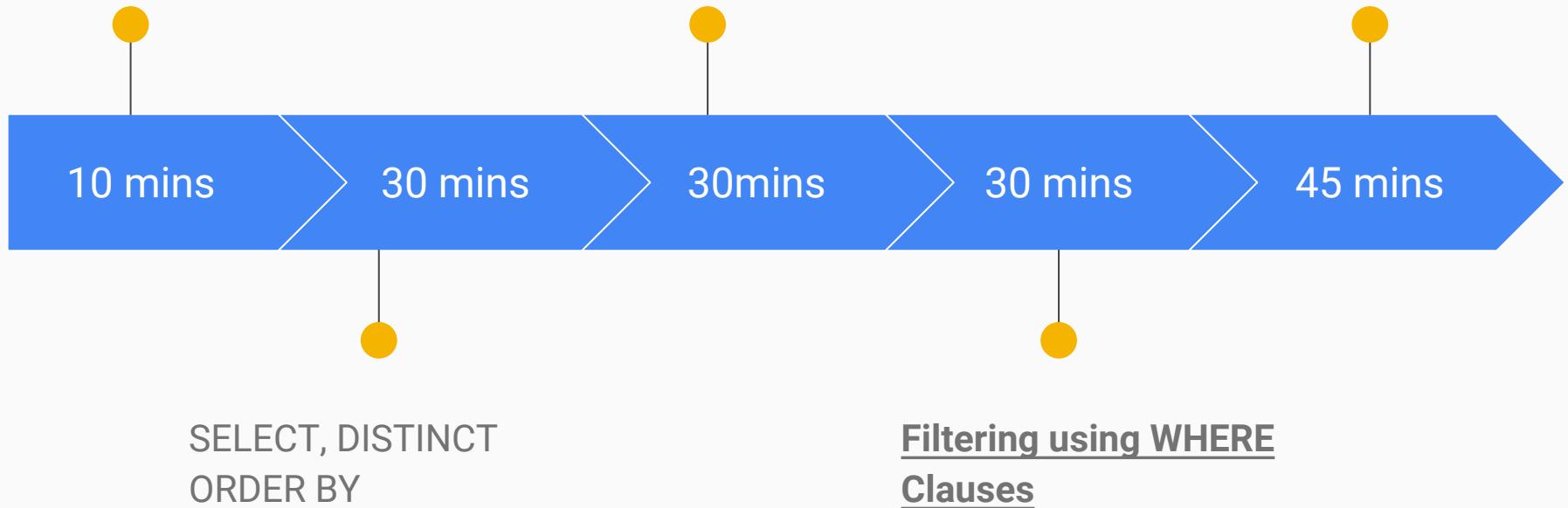
- Database Server with Interface
 - MS SQL Server with SQL Server Management Studio
- OR
- MySQL with MySQL Workbench
(part of MySQL install)
- OR
- Online (sqlfiddle.com)
- Data
 - Test data on Github repo.
 - Github repo for information

Overview of Day 2

Basic SQL Queries (SELECT)

Expressions, Casting, NULL

Practice

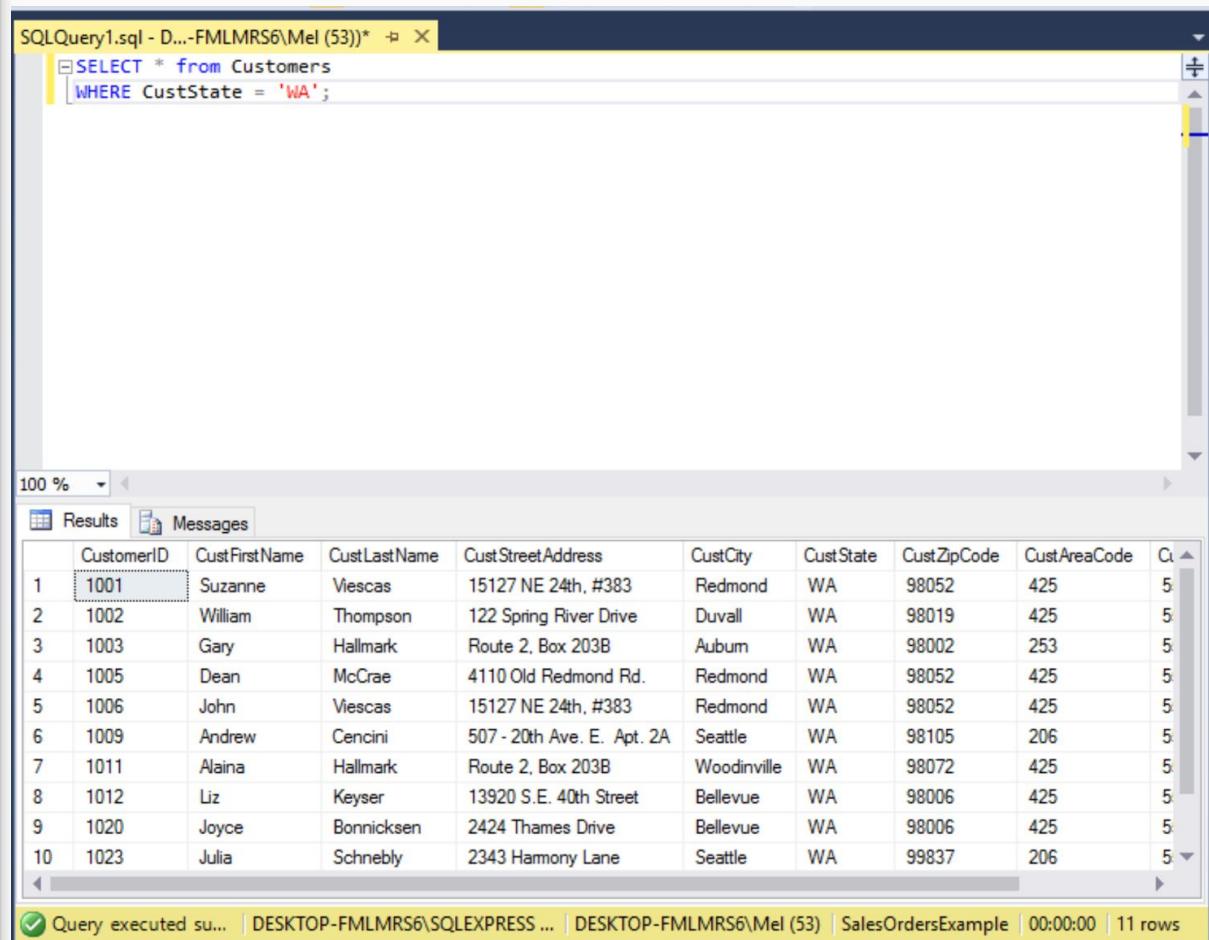


Day 2, Part C: Filtering Data

WHERE Clause

Filtering data with the WHERE clause

- The WHERE clause contains the search **condition** (think conditional statement) of the query.
- It's how you filter the data returned by the SELECT statement.



A screenshot of the SQL Server Management Studio (SSMS) interface. The top bar shows the title "SQLQuery1.sql - D...-FMLMRS6\Mel (53)*". The query window contains the following SQL code:

```
SELECT * from Customers  
WHERE CustState = 'WA';
```

The results pane displays a table with 11 rows of customer data, filtered by the condition where CustState is 'WA'. The columns are CustomerID, CustFirstName, CustLastName, CustStreetAddress, CustCity, CustState, CustZipCode, CustAreaCode, and CustPhone. The data includes customers from Redmond, Duvall, Auburn, and Seattle.

CustomerID	CustFirstName	CustLastName	CustStreetAddress	CustCity	CustState	CustZipCode	CustAreaCode	CustPhone
1001	Suzanne	Viescas	15127 NE 24th, #383	Redmond	WA	98052	425	503-333-0491
1002	William	Thompson	122 Spring River Drive	Duvall	WA	98019	425	425-555-0144
1003	Gary	Hallmark	Route 2, Box 203B	Auburn	WA	98002	253	425-555-0145
1005	Dean	McCrae	4110 Old Redmond Rd.	Redmond	WA	98052	425	425-555-0146
1006	John	Viescas	15127 NE 24th, #383	Redmond	WA	98052	425	425-555-0147
1009	Andrew	Cencini	507 - 20th Ave. E. Apt. 2A	Seattle	WA	98105	206	206-555-0148
1011	Alaina	Hallmark	Route 2, Box 203B	Woodinville	WA	98072	425	425-555-0149
1012	Liz	Keyser	13920 S.E. 40th Street	Bellevue	WA	98006	425	425-555-0150
1020	Joyce	Bonnicksen	2424 Thames Drive	Bellevue	WA	98006	425	425-555-0151
1023	Julia	Schnebly	2343 Harmony Lane	Seattle	WA	99837	206	206-555-0152

At the bottom, a status bar indicates: "Query executed successfully | DESKTOP-FMLMRS6\SQLEXPRESS... | DESKTOP-FMLMRS6\Mel (53) | SalesOrdersExample | 00:00:00 | 11 rows".

What can you compare?

- You can compare literals, columns, or a combination of these. (Think of columns like variable names you'd find in C# or Java.)

WHERE 1 = 1;

WHERE some_col = 1;

WHERE some_col = another_col;

Use caution when comparing strings

- You may get data that you aren't expecting when comparing strings.
- This has to deal with the collation of the database.
- A doesn't always equal a.
- Character sets may not compare equally either.
- Usually DB Admins handle the default collation for the database.
- Read more here: <https://mariadb.com/kb/en/sql-99/character-strings-and-collations/> and <http://dev.mysql.com/doc/refman/5.7/en/charset-general.html>

Predicates

Predicates in a WHERE Clause

- Predicates are like operators in an *if statement*.
- There are 18 predicates in SQL, we will discuss 5, the most used ones.
- Comparison (=, <>, >, <, >=, <=)
- LIKE
- BETWEEN
- IN
- IS NULL

Comparison Predicates: “=”

- Equal to.
- When you want a precise match.
- CustState = 'WA'
- CustLastName = 'Hallmark'
- CustomerID = 1001;
- CustAreaCode = 425;

SQLQuery1.sql - D...-FMLMRS6\Mel (53)* X

```
SELECT * from Customers
WHERE CustState = 'WA';
```

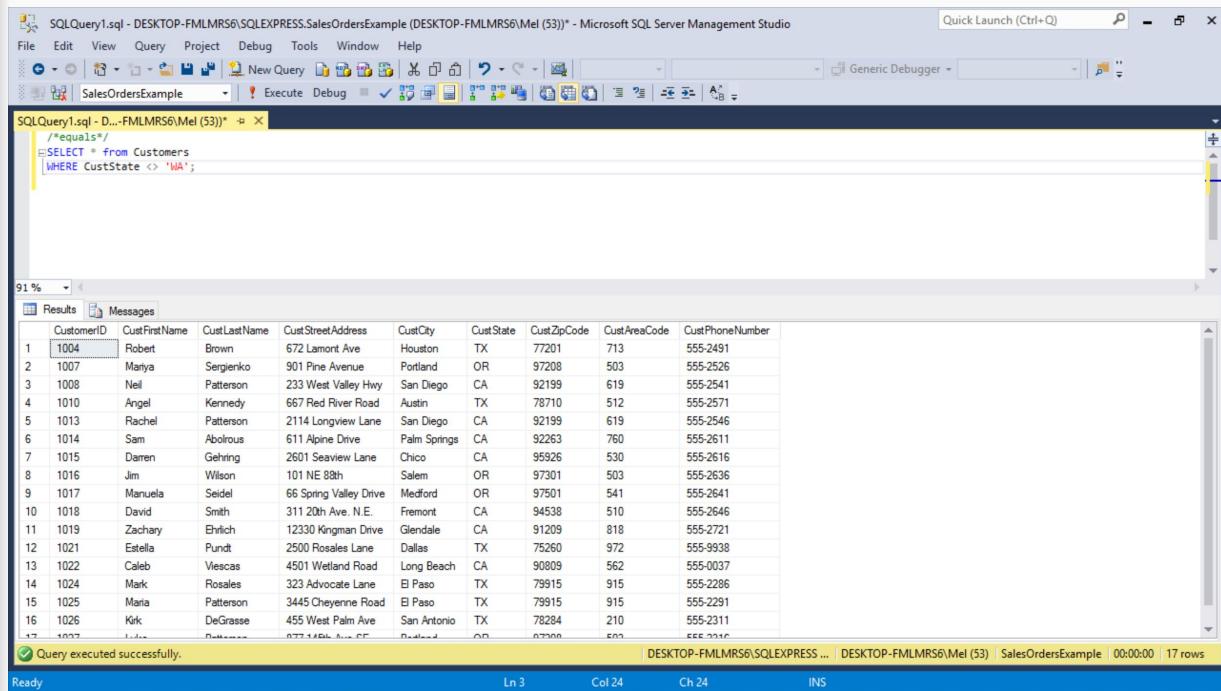
100 %

	CustomerID	CustFirstName	CustLastName	CustStreetAddress	CustCity	CustState	CustZipCode	CustAreaCode	Ci
1	1001	Suzanne	Viescas	15127 NE 24th, #383	Redmond	WA	98052	425	5
2	1002	William	Thompson	122 Spring River Drive	Duvall	WA	98019	425	5
3	1003	Gary	Hallmark	Route 2, Box 203B	Auburn	WA	98002	253	5
4	1005	Dean	McCrae	4110 Old Redmond Rd.	Redmond	WA	98052	425	5
5	1006	John	Viescas	15127 NE 24th, #383	Redmond	WA	98052	425	5
6	1009	Andrew	Cencini	507 - 20th Ave. E. Apt. 2A	Seattle	WA	98105	206	5
7	1011	Alaina	Hallmark	Route 2, Box 203B	Woodinville	WA	98072	425	5
8	1012	Liz	Keyser	13920 S.E. 40th Street	Bellevue	WA	98006	425	5
9	1020	Joyce	Bonnicksen	2424 Thames Drive	Bellevue	WA	98006	425	5
10	1023	Julia	Schnebly	2343 Harmony Lane	Seattle	WA	99837	206	5

Query executed su... | DESKTOP-FMLMRS6\SQLEXPRESS ... | DESKTOP-FMLMRS6\Mel (53) | SalesOrdersExample | 00:00:00 | 11 rows

Comparison Predicates: “<>”

- Not equal to.
- When you want a precise match that isn't equal to something.
- CustState <> 'WA'
- CustLastName <> 'Hallmark'
- CustomerID <> 1001;
- CustAreaCode <> 425;



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery1.sql' is open, displaying the following T-SQL code:

```
/*!=equals*/
SELECT * from Customers
WHERE CustState <> 'WA';
```

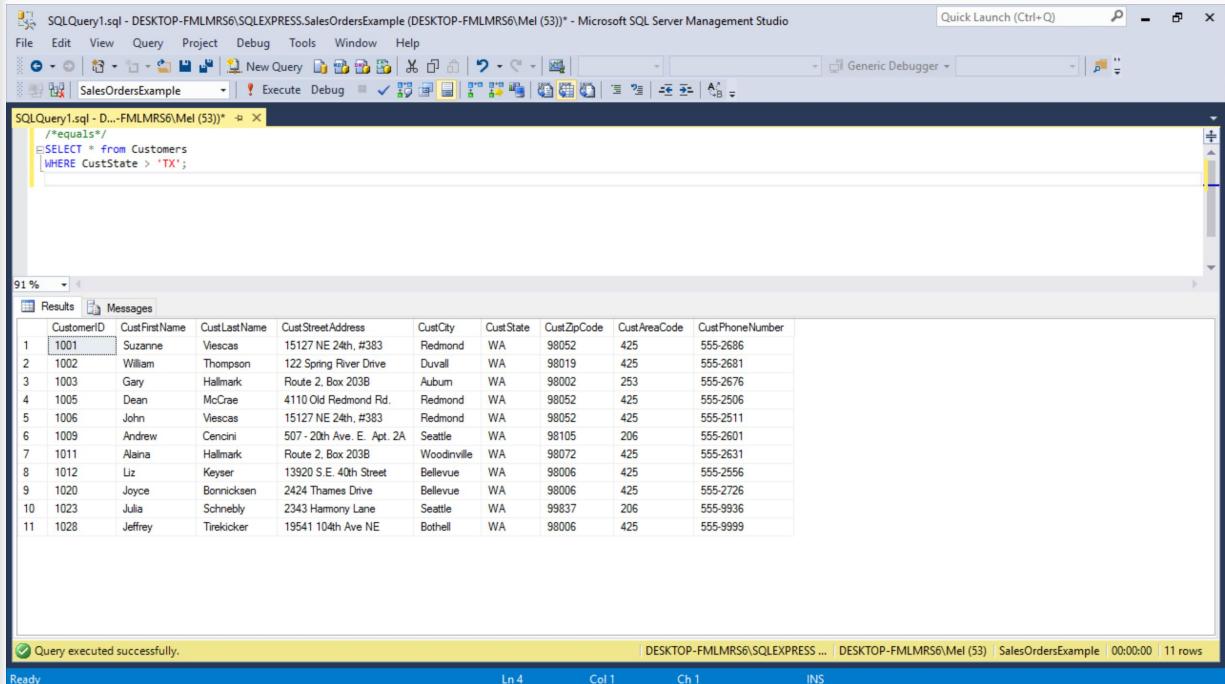
The results grid below the query window shows 17 rows of customer data from the 'Customers' table, filtered by CustState not equal to 'WA'. The columns are: CustomerID, CustFirstName, CustLastName, CustStreetAddress, CustCity, CustState, CustZipCode, CustAreaCode, and CustPhoneNumber. The data includes various customers like Robert Brown, Maria Sergienko, Neil Patterson, etc., from cities like Houston, Portland, San Diego, Austin, etc., across different states like TX, OR, CA, etc.

CustomerID	CustFirstName	CustLastName	CustStreetAddress	CustCity	CustState	CustZipCode	CustAreaCode	CustPhoneNumber
1004	Robert	Brown	672 Lamont Ave	Houston	TX	77201	713	555-2491
1007	Maria	Sergienko	901 Fine Avenue	Portland	OR	97208	503	555-2526
1008	Neil	Patterson	233 West Valley Hwy	San Diego	CA	92199	619	555-2541
1010	Angel	Kennedy	667 Red River Road	Austin	TX	78710	512	555-2571
1013	Rachel	Patterson	2114 Longview Lane	San Diego	CA	92199	619	555-2546
1014	Sam	Abolrous	611 Alpine Drive	Palm Springs	CA	92263	760	555-2611
1015	Daren	Gehring	2601 Seaview Lane	Chico	CA	95926	530	555-2616
1016	Jim	Wilson	101 NE 88th	Salem	OR	97301	503	555-2636
1017	Manuela	Seidel	66 Spring Valley Drive	Medford	OR	97501	541	555-2641
1018	David	Smith	311 20th Ave. N.E.	Fremont	CA	94538	510	555-2646
1019	Zachary	Ehrlich	12330 Kingman Drive	Glendale	CA	91209	818	555-2721
1021	Estella	Pundt	2500 Rosales Lane	Dallas	TX	75260	972	555-9938
1022	Caleb	Viecas	4501 Wetland Road	Long Beach	CA	90809	562	555-0037
1024	Mark	Rosales	323 Advocate Lane	El Paso	TX	79915	915	555-2286
1025	Maria	Patterson	3445 Cheyenne Road	El Paso	TX	79915	915	555-2291
1026	Kirk	DeGrasse	455 West Palm Ave	San Antonio	TX	78284	210	555-2311
1027	Laura	Dellinger	877 14th Ave SE	Seattle	OR	97206	500	555-7216

At the bottom of the results grid, a message indicates: "Query executed successfully." Below the results grid, the status bar shows: Ready, Ln 3, Col 24, Ch 24, INS.

Comparison Predicates: “>”

- Greater than.
- When you want results where the attribute on the left is greater than that of the right.
- CustState > 'WA'
- CustLastName > 'Hallmark'
- CustomerID > 1001;
- CustAreaCode > 425;



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled "SQLQuery1.sql - DESKTOP-FMLMRS6\Mel (53)*" is open, displaying the following SQL code:

```
/*equals*/
SELECT * from Customers
WHERE CustState > 'TX';
```

The results grid shows 11 rows of customer data from the "Customers" table. The columns are: CustomerID, CustFirstName, CustLastName, CustStreetAddress, CustCity, CustState, CustZipCode, CustAreaCode, and CustPhoneNumber. The data includes various customers from Washington (WA) and other states like Texas (TX), Oregon (OR), and others. The results grid has a header row and 11 data rows. The status bar at the bottom indicates "Query executed successfully." and "11 rows".

CustomerID	CustFirstName	CustLastName	CustStreetAddress	CustCity	CustState	CustZipCode	CustAreaCode	CustPhoneNumber
1	Suzanne	Viescas	15127 NE 24th, #383	Redmond	WA	98052	425	555-2686
2	William	Thompson	122 Spring River Drive	Duvall	WA	98019	425	555-2681
3	Gary	Hallmark	Route 2, Box 203B	Auburn	WA	98002	253	555-2676
4	Dean	McCrae	4110 Old Redmond Rd.	Redmond	WA	98052	425	555-2506
5	John	Viescas	15127 NE 24th, #383	Redmond	WA	98052	425	555-2511
6	Andrew	Cencini	507 - 20th Ave. E. Apt. 2A	Seattle	WA	98105	206	555-2601
7	Alaina	Hallmark	Route 2, Box 203B	Woodinville	WA	98072	425	555-2631
8	Liz	Keyser	13920 S.E. 40th Street	Bellevue	WA	98006	425	555-2556
9	Joyce	Bonnickson	2424 Thames Drive	Bellevue	WA	98006	425	555-2726
10	Julia	Schnebly	2343 Harmony Lane	Seattle	WA	99837	206	555-9936
11	Jeffrey	Tirekicker	19541 104th Ave NE	Bothell	WA	98006	425	555-9999

Comparison Predicates: “<”

- Less than.
- When you want results where the attribute on the left is less than that of the right.
- CustState < 'WA'
- CustLastName < 'Hallmark'
- CustomerID < 1001;
- CustAreaCode < 425;

The screenshot shows the Microsoft SQL Server Management Studio interface. The top window is titled "SQLQuery1.sql - DESKTOP-FMLMRS6\SQLEXPRESS.SalesOrdersExample (DESKTOP-FMLMRS6\Mel (53)) - Microsoft SQL Server Management Studio". The query window contains the following SQL code:

```
/*equal to*/
SELECT * From Customers
WHERE CustState < 'WA';
```

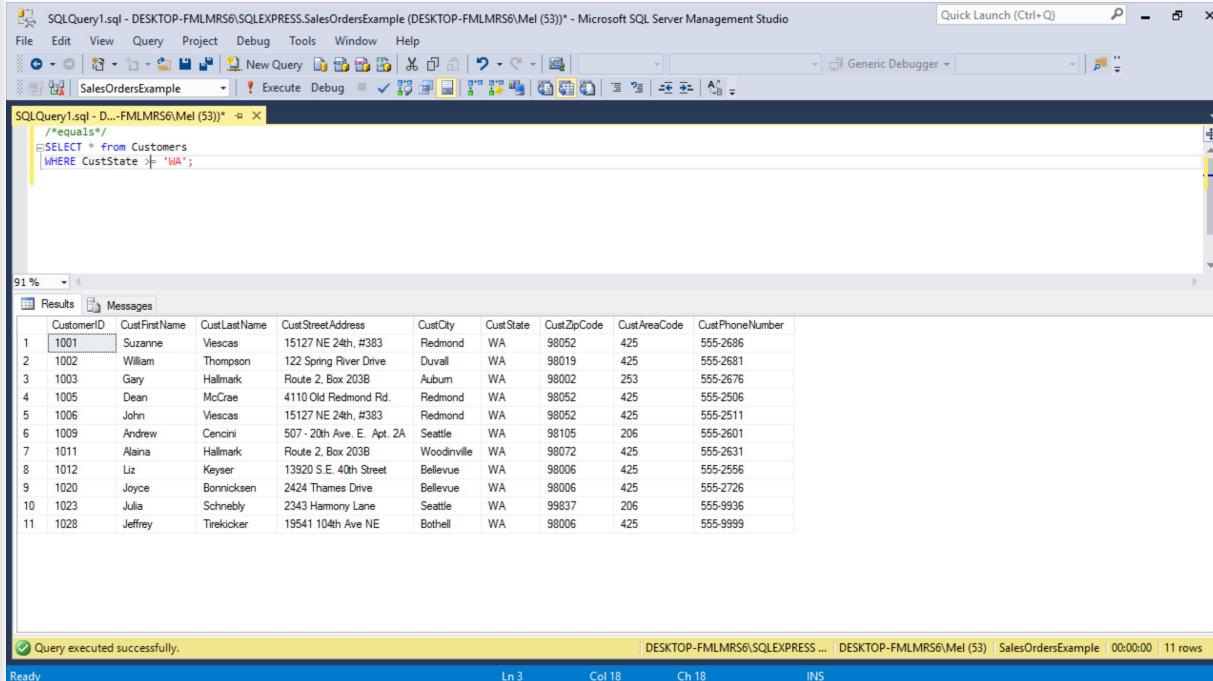
The results window displays a table of customer data. The columns are: CustomerID, CustFirstName, CustLastName, CustStreetAddress, CustCity, CustState, CustZipCode, CustAreaCode, and CustPhoneNumber. The data shows 17 rows of customers from various states, with the first few rows being:

CustomerID	CustFirstName	CustLastName	CustStreetAddress	CustCity	CustState	CustZipCode	CustAreaCode	CustPhoneNumber
1	Robert	Brown	672 Lamont Ave	Houston	TX	77201	713	555-2491
2	Maria	Sergienko	901 Pine Avenue	Portland	OR	97208	503	555-2526
3	Neil	Patterson	233 West Valley Hwy	San Diego	CA	92199	619	555-2541
4	Angel	Kennedy	667 Red River Road	Austin	TX	78710	512	555-2571
5	Rachel	Patterson	2114 Longview Lane	San Diego	CA	92199	619	555-2546
6	Sam	Abolrous	611 Alpine Drive	Palm Springs	CA	92263	760	555-2611
7	Daren	Gehring	2601 Seaview Lane	Chico	CA	95926	530	555-2616
8	Jim	Wilson	101 NE 8th	Salon	OR	97301	503	555-2636
9	Manuela	Seidel	66 Spring Valley Drive	Medford	OR	97501	541	555-2641
10	David	Smith	311 20th Ave. N.E.	Fremont	CA	94538	510	555-2646
11	Zachary	Ehrlich	12330 Kingman Drive	Glendale	CA	91209	818	555-2721
12	Estella	Pundt	2500 Rosales Lane	Dallas	TX	75260	972	555-9938
13	Caleb	Vescas	4501 Wetland Road	Long Beach	CA	90809	562	555-0037
14	Mark	Rosales	323 Advocate Lane	El Paso	TX	79915	915	555-2286
15	Maria	Patterson	3445 Cheyenne Road	El Paso	TX	79915	915	555-2291
16	Kirk	DeGrasse	455 West Palm Ave	San Antonio	TX	78284	210	555-2311
17	Debra	Ortiz	8731 15th Ave SE	Portland	OR	97206	502	555-2312

The status bar at the bottom indicates "Query executed successfully." and "DESKTOP-FMLMRS6\SQLEXPRESS ... DESKTOP-FMLMRS6\Mel (53) SalesOrdersExample 00:00:00 17 rows".

Comparison Predicates: “>=”

- Less than or equal to.
- When you want results where the attribute on the left is greater than or equal to that of the right.
- CustState >= 'WA'
- CustLastName >= 'Hallmark'
- CustomerID >= 1001;
- CustAreaCode >= 425;



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery1.sql' is open, displaying the following T-SQL code:

```
/*equals*/
SELECT * from Customers
WHERE CustState >= 'WA';
```

The results pane shows a grid of customer data from the 'Customers' table, filtered by the condition 'CustState >= 'WA''. The columns are: CustomerID, CustFirstName, CustLastName, CustStreetAddress, CustCity, CustState, CustZipCode, CustAreaCode, and CustPhoneNumber. The data includes rows for customers like Suzanne Vescas, William Thompson, and many others from Washington state.

CustomerID	CustFirstName	CustLastName	CustStreetAddress	CustCity	CustState	CustZipCode	CustAreaCode	CustPhoneNumber
1	1001	Suzanne	15127 NE 24th, #383	Redmond	WA	98052	425	555-2686
2	1002	William	122 Spring River Drive	Duvall	WA	98019	425	555-2681
3	1003	Gary	Hallmark	Route 2, Box 203B	Auburn	WA	98002	253
4	1005	Dean	McCrae	4110 Old Redmond Rd.	Redmond	WA	98052	425
5	1006	John	Vescas	15127 NE 24th, #383	Redmond	WA	98052	425
6	1009	Andrew	Cencini	507 - 20th Ave. E. Apt. 2A	Seattle	WA	98105	206
7	1011	Alaina	Hallmark	Route 2, Box 203B	Woodinville	WA	98072	425
8	1012	Liz	Keyser	13920 S.E. 40th Street	Bellevue	WA	98006	425
9	1020	Joyce	Bonnicksen	2424 Thames Drive	Bellevue	WA	98006	425
10	1023	Julia	Schnebly	2343 Harmony Lane	Seattle	WA	99837	206
11	1028	Jeffrey	Tirekicker	19541 104th Ave NE	Bothell	WA	98006	425

At the bottom of the results pane, a message states "Query executed successfully." and the status bar indicates "Ready", "Ln 3", "Col 18", "Ch 18", and "INS".

Comparison Predicates: “ \leq ”

- Less than or equal to.
- When you want results where the attribute on the left is less than or equal to that of the right.
- `CustState <= 'WA'`
- `CustLastName <= 'Hallmark'`
- `CustomerID <= 1001;`
- `CustAreaCode <= 425;`

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

```
/*@equals*/
SELECT * from Customers
WHERE CustState <= 'WA';
```

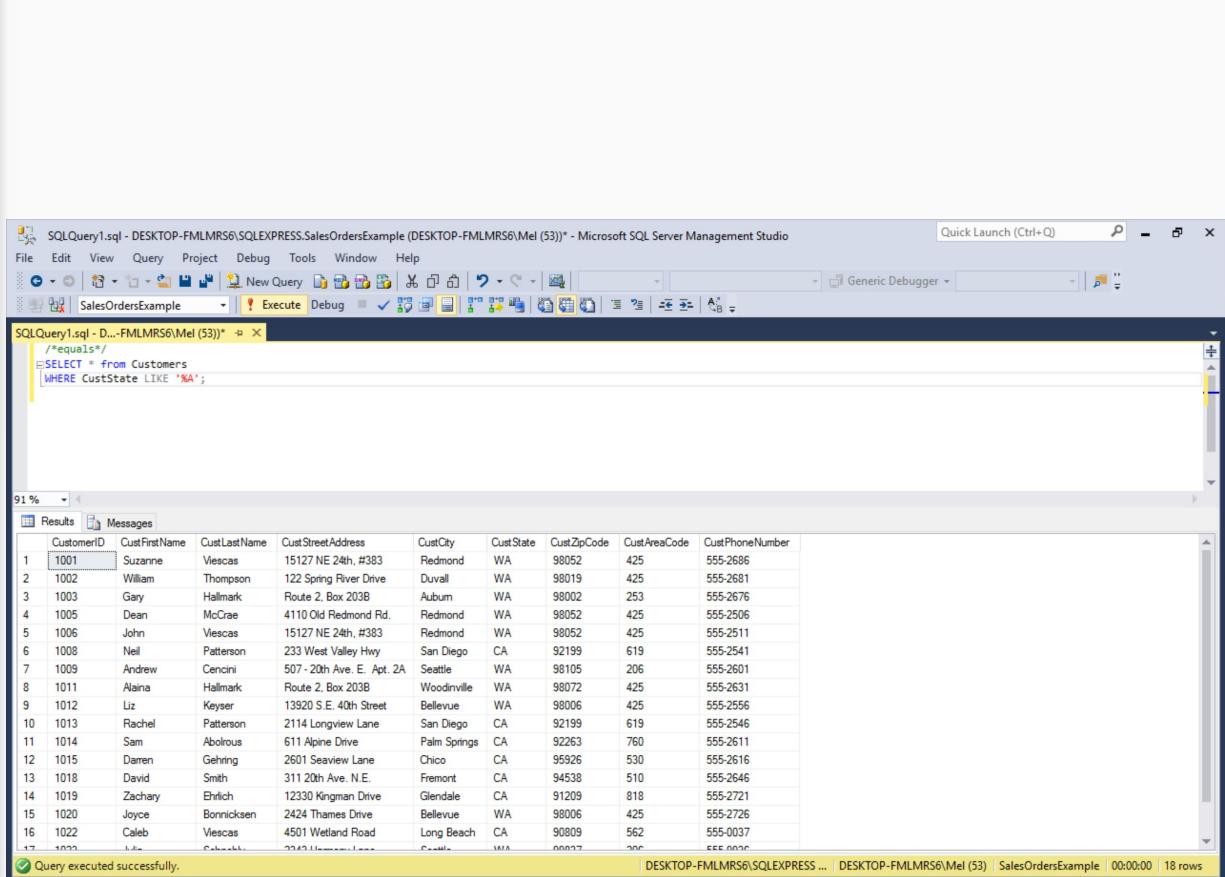
The results grid displays 18 rows of customer data from the 'Customers' table, filtered by state ('WA'). The columns are CustomerID, CustFirstName, CustLastName, CustStreetAddress, CustCity, CustState, CustZipCode, CustAreaCode, and CustPhoneNumber. The results are as follows:

CustomerID	CustFirstName	CustLastName	CustStreetAddress	CustCity	CustState	CustZipCode	CustAreaCode	CustPhoneNumber
1001	Suzanne	Vescas	15127 NE 24th, #383	Redmond	WA	98052	425	555-2686
1002	William	Thompson	122 Spring River Drive	Duvall	WA	98019	425	555-2681
1003	Gary	Hallmark	Route 2, Box 203B	Auburn	WA	98002	253	555-2676
1004	Robert	Brown	672 Lamont Ave	Houston	TX	77201	713	555-2491
1005	Dean	McCrae	4110 Old Redmond Rd.	Redmond	WA	98052	425	555-2506
1006	John	Vescas	15127 NE 24th, #383	Redmond	WA	98052	425	555-2511
1007	Maryya	Sergienko	901 Pine Avenue	Portland	OR	97208	503	555-2526
1008	Neil	Patterson	233 West Valley Hwy	San Diego	CA	92199	619	555-2541
1009	Andrew	Cencini	507 - 20th Ave. E. Apt. 2A	Seattle	WA	98105	206	555-2601
1010	Angel	Kennedy	667 Red River Road	Austin	TX	78710	512	555-2571
1011	Alaina	Hallmark	Route 2, Box 203B	Woodinville	WA	98072	425	555-2631
1012	Liz	Keyser	13920 S.E. 40th Street	Bellevue	WA	98006	425	555-2556
1013	Rachel	Patterson	2114 Longview Lane	San Diego	CA	92199	619	555-2546
1014	Sam	Abolrous	611 Alpine Drive	Palm Springs	CA	92263	760	555-2611
1015	Darren	Gehring	2601 Seaview Lane	Chico	CA	95926	530	555-2616
1016	Jim	Wilson	101 NE 88th	Salem	OR	97301	503	555-2636
1017	Marcia	Cardinal	6675 South Valley Drive	Marked	OR	97021	545	555-2641

At the bottom of the screen, the taskbar shows the status bar with "Query executed successfully." and the system clock at "1:51 AM 12/28/2016".

LIKE Predicate

- Pattern matching.
- Uses a wildcard to get a match that is similar to something.
- Great when you are looking up a value and you only have partial information.
- Wildcard is a percent sign (%) or an underscore (_).
- % is used for one or more characters.
- _ is used for one character.
- CustState LIKE '_A'
- CustLastName <= '_all_'
- CustomerID <= 10%;
- CustAreaCode <= _2%;



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery1.sql' is open, displaying the following T-SQL code:

```
/*equals*/
SELECT * FROM Customers
WHERE CustState LIKE '%A';
```

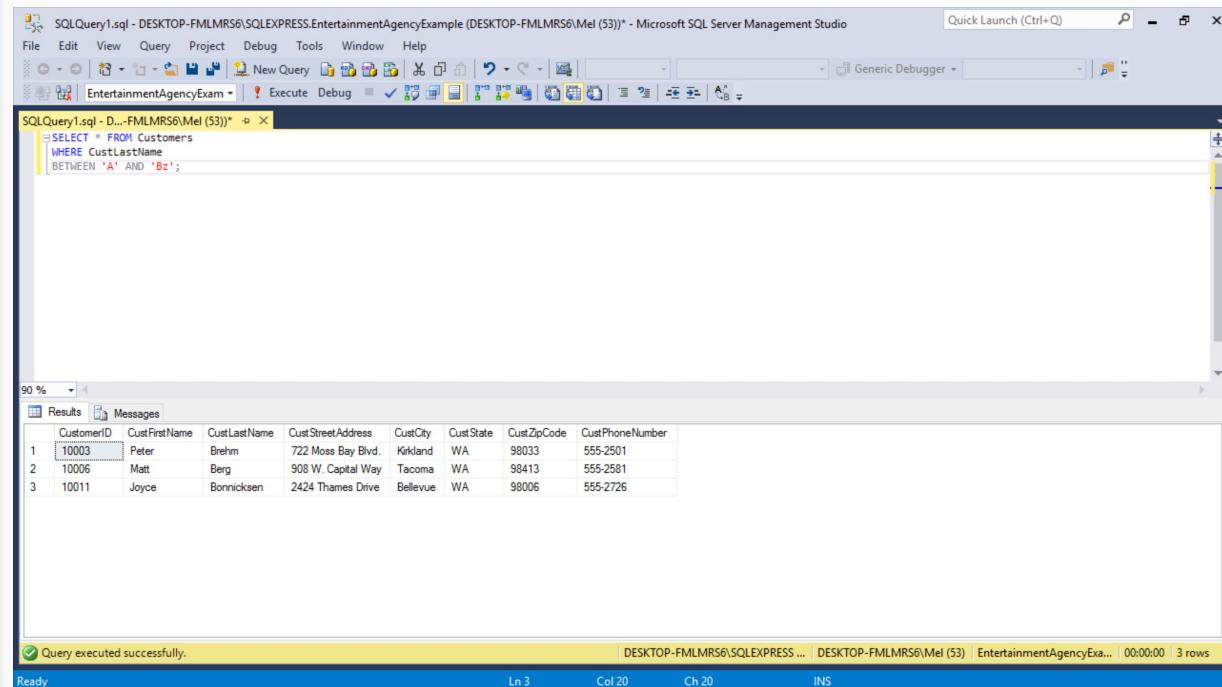
The results grid below shows 18 rows of customer data from the 'Customers' table, filtered by customers whose state name starts with 'A'. The columns are: CustomerID, CustFirstName, CustLastName, CustStreetAddress, CustCity, CustState, CustZipCode, CustAreaCode, and CustPhoneNumber. The data includes entries like 'Redmond WA' and 'Seattle WA'.

CustomerID	CustFirstName	CustLastName	CustStreetAddress	CustCity	CustState	CustZipCode	CustAreaCode	CustPhoneNumber
1001	Suzanne	Viescas	15127 NE 24th, #383	Redmond	WA	98052	425	555-2686
1002	William	Thompson	122 Spring River Drive	Duvall	WA	98019	425	555-2681
1003	Gary	Hallmark	Route 2, Box 203B	Auburn	WA	98002	253	555-2676
1005	Dean	McCrae	4110 Old Redmond Rd.	Redmond	WA	98052	425	555-2506
1006	John	Viescas	15127 NE 24th, #383	Redmond	WA	98052	425	555-2511
1008	Nell	Patterson	233 West Valley Hwy	San Diego	CA	92199	619	555-2541
1009	Andrew	Cencini	507 - 20th Ave. E. Apt. 2A	Seattle	WA	98105	206	555-2601
1011	Alaina	Hallmark	Route 2, Box 203B	Woodinville	WA	98072	425	555-2631
1012	Liz	Keyser	13920 S.E. 40th Street	Bellevue	WA	98006	425	555-2556
1013	Rachel	Patterson	2114 Longview Lane	San Diego	CA	92199	619	555-2546
1014	Sam	Abolrous	611 Alpine Drive	Palm Springs	CA	92263	760	555-2611
1015	Daren	Gehring	2601 Seaview Lane	Chico	CA	95926	530	555-2616
1018	David	Smith	311 20th Ave. N.E.	Fremont	CA	94538	510	555-2646
1019	Zachary	Ehrlich	12330 Kingman Drive	Glendale	CA	91209	818	555-2721
1020	Joyce	Bonnickson	2424 Thames Drive	Bellevue	WA	98006	425	555-2726
1022	Caleb	Viescas	4501 Wetland Road	Long Beach	CA	90809	562	555-0037
1023	Laura	Calderon	2242 University Lane	Seattle	WA	98101	206	555-0020

At the bottom of the results grid, a message states: 'Query executed successfully.'

BETWEEN Predicate

- Uses data to compare within a range.
- Great when you are looking up information between two dates, or numbers, etc.
- Use BETWEEN like you would for *something* $\geq x$ AND *something* $\leq y$.
- CustState BETWEEN 'A' AND 'Bz';
- CustLastName BETWEEN 's' AND 'Z';
- CustomerID BETWEEN 1000 AND 1099;
- CustAreaCode BETWEEN 400 AND 440;;



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled "SQLQuery1.sql" is open, displaying the following T-SQL code:

```
SELECT * FROM Customers
WHERE CustLastName
BETWEEN 'A' AND 'Bz';
```

The results pane below shows a table with three rows of customer data:

CustomerID	CustFirstName	CustLastName	CustStreetAddress	CustCity	CustState	CustZipCode	CustPhoneNumber
1	Peter	Brehm	722 Moss Bay Blvd.	Kirkland	WA	98033	555-2501
2	Matt	Berg	908 W. Capital Way	Tacoma	WA	98413	555-2581
3	Joyce	Bonnicksen	2424 Thames Drive	Bellevue	WA	98006	555-2726

A status bar at the bottom indicates "Query executed successfully." and "3 rows".

IN Predicate

- Use IN to see if something exists in a certain group or set.
- Great when you are looking up a value within a set of data.
- I've used subqueries (a sql query inside another query) to create a set of data to use within the IN statement.
- CustState IN ('OH', 'IL', 'TX');
- CustLastName IN ('Brehm', 'Bonnicksen', 'Thompson');
- CustomerID IN (1000, 1001, 10011, 1200);
- CustAreaCode IN (216, 440, 614);

The screenshot shows the Microsoft SQL Server Management Studio interface. In the top window, titled 'SQLQuery1.sql - DESKTOP-FMLMRS6\SQLEXPRESS.EntertainmentAgencyExample (DESKTOP-FMLMRS6\Mel (53))* - Microsoft SQL Server Management Studio', a T-SQL query is written:

```
SELECT * FROM Customers
WHERE CustLastName
IN ('Brehm', 'Bonnicksen', 'Smith', 'Thompson');
```

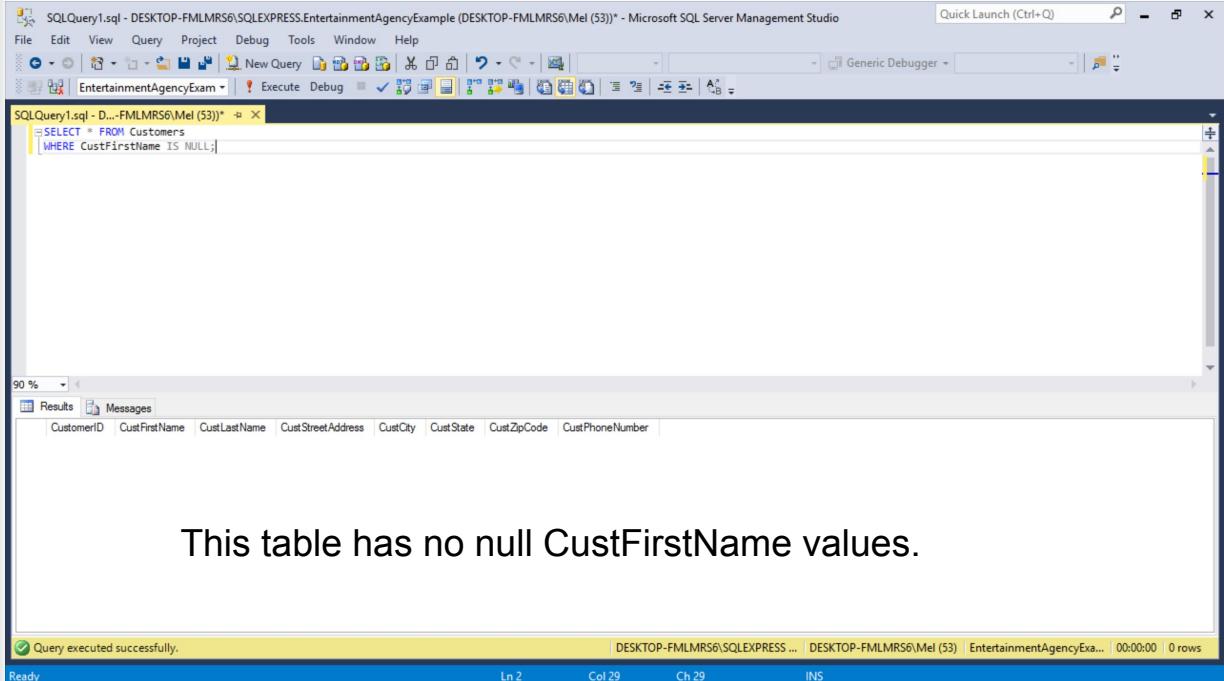
In the bottom window, titled 'Results', the query's output is displayed in a grid:

	CustomerID	CustFirstName	CustLastName	CustStreetAddress	CustCity	CustState	CustZipCode	CustPhoneNumber
1	10003	Peter	Brehm	722 Moss Bay Blvd.	Kirkland	WA	98033	555-2501
2	10009	Sarah	Thompson	2222 Springer Road	Bellevue	WA	98006	555-2626
3	10011	Joyce	Bonnicksen	2424 Thames Drive	Bellevue	WA	98006	555-2726

At the bottom of the interface, a message bar indicates: 'Query executed successfully.' and shows the session details: 'DESKTOP-FMLMRS6\SQLEXPRESS ... | DESKTOP-FMLMRS6\Mel (53) | EntertainmentAgencyExa... | 00:00:00 | 3 rows'.

IS NULL Predicate

- Use IS NULL to search for unknown values (NULL values in fields).
- Great when you are looking up information to fix or which is unknown.
- Great when you are looking up information to exclude when creating a mathematical expression that could choke on a NULL value.
- DO NOT use = NULL! Use IS NULL instead.
- CustState IS NULL;
- CustLastName IS NULL;
- CustAreaCode IS NULL



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery1.sql' is open, displaying the following SQL code:

```
SELECT * FROM Customers
WHERE CustFirstName IS NULL;
```

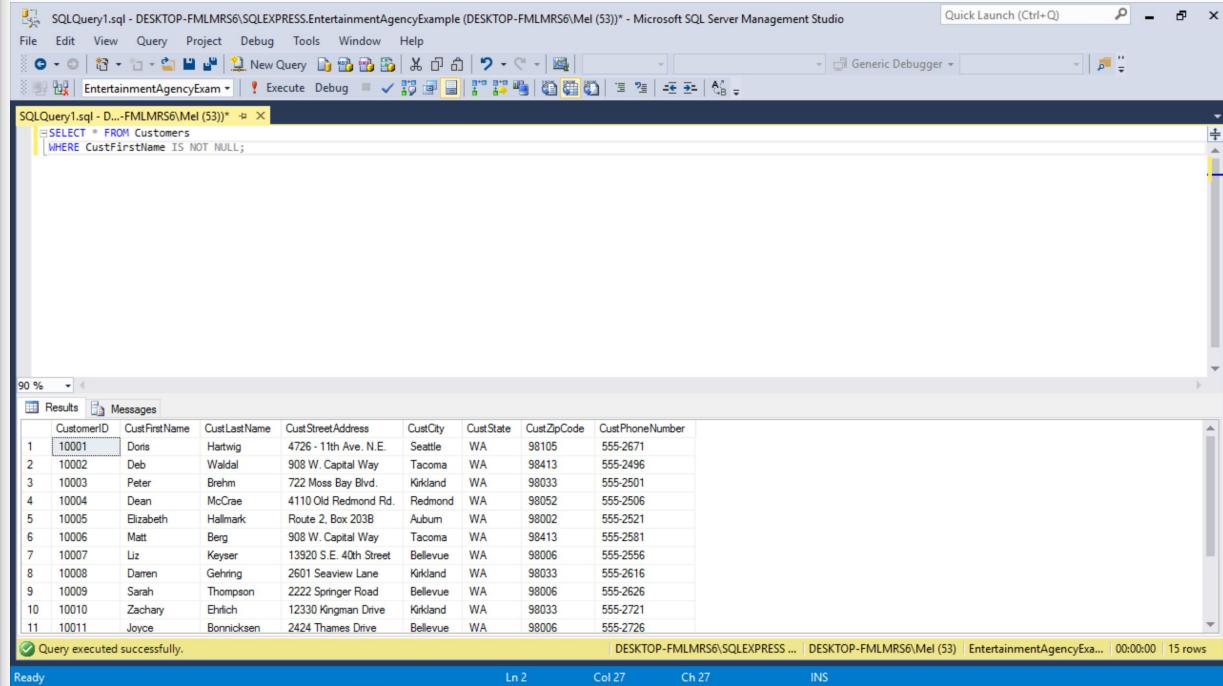
The results pane below shows a table with columns: CustomerID, CustFirstName, CustLastName, CustStreetAddress, CustCity, CustState, CustZipCode, and CustPhoneNumber. The results table is currently empty, indicating no rows were found where CustFirstName is NULL.

Text overlay: This table has no null CustFirstName values.

Status bar at the bottom: Query executed successfully. DESKTOP-FMLMRS6\SQLEXPRESS ... DESKTOP-FMLMRS6\Mel (53) EntertainmentAgencyExa... 00:00:00 | 0 rows

NOT Modifier

- With IN, LIKE, BETWEEN IS NULL, you can check for the inverse with the NOT modifier.
- CustState IS NOT NULL;
- CustLastName IS NOT BETWEEN 'A' AND 'Bz';
- CustAreaCode IS NOT IN (440, 216, 614)



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled "SQLQuery1.sql - DESKTOP-FMLMRS6\SQLEXPRESS.EducationAgencyExample (DESKTOP-FMLMRS6\Mel (53))" contains the following SQL code:

```
SELECT * FROM Customers
WHERE CustFirstName IS NOT NULL;
```

The results grid displays 15 rows of customer data from the "Customers" table, filtered to show only those with non-null first names. The columns are: CustomerID, CustFirstName, CustLastName, CustStreetAddress, CustCity, CustState, CustZipCode, and CustPhoneNumber. The data includes entries like Doris Hartwig, Deb Waldal, Peter Brehm, etc.

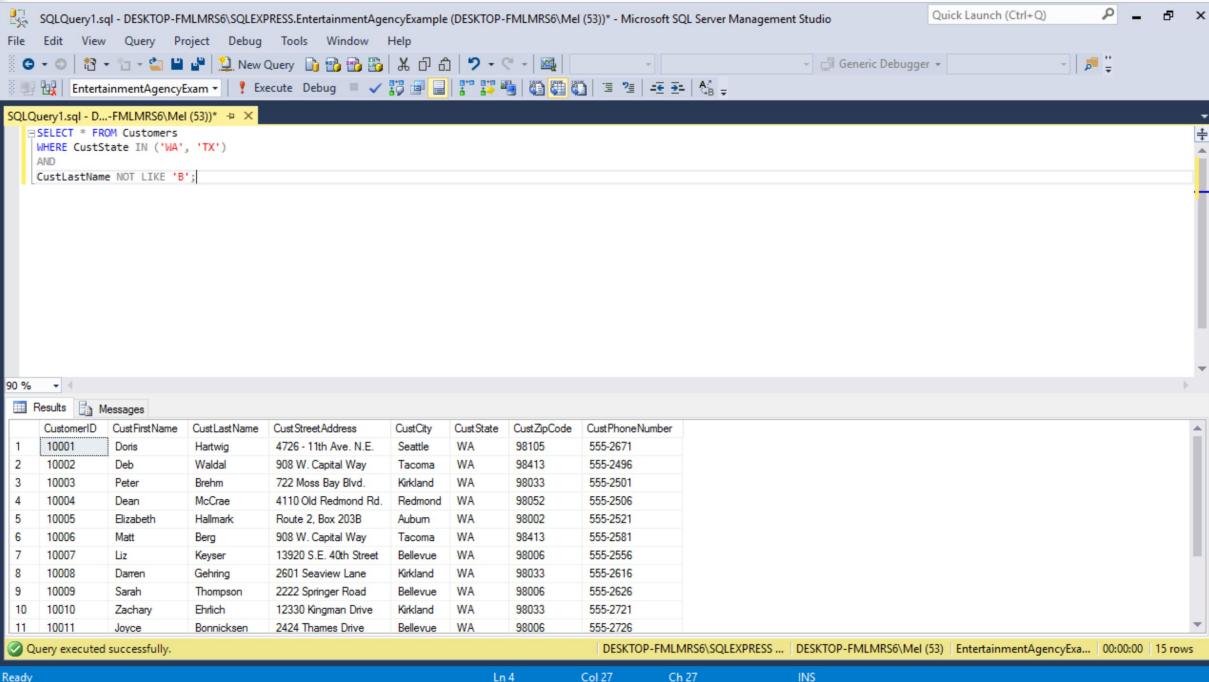
CustomerID	CustFirstName	CustLastName	CustStreetAddress	CustCity	CustState	CustZipCode	CustPhoneNumber
10001	Doris	Hartwig	4726 - 11th Ave. N.E.	Seattle	WA	98105	555-2671
10002	Deb	Waldal	908 W. Capital Way	Tacoma	WA	98413	555-2496
10003	Peter	Brehm	722 Moss Bay Blvd.	Kirkland	WA	98033	555-2501
10004	Dean	McCrae	4110 Old Redmond Rd.	Redmond	WA	98052	555-2506
10005	Elizabeth	Hallmark	Route 2, Box 203B	Auburn	WA	98002	555-2521
10006	Matt	Berg	908 W. Capital Way	Tacoma	WA	98413	555-2581
10007	Liz	Keyser	13920 S.E. 40th Street	Bellevue	WA	98006	555-2556
10008	Darren	Gehring	2601 Seaview Lane	Kirkland	WA	98033	555-2616
10009	Sarah	Thompson	2222 Springer Road	Bellevue	WA	98006	555-2626
10010	Zachary	Ehrlich	12330 Kingman Drive	Kirkland	WA	98033	555-2721
10011	Joyce	Bonnicksen	2424 Thames Drive	Bellevue	WA	98006	555-2726

At the bottom of the results grid, a message states "Query executed successfully." The status bar at the bottom of the screen shows "DESKTOP-FMLMRS6\SQLEXPRESS ... | DESKTOP-FMLMRS6\Mel (53) | EntertainmentAgencyExam... | 00:00:00 | 15 rows".

Multiple Conditions

Multiple Conditions Overview

- You can use keywords like AND and OR to include or exclude other conditions in your WHERE clause.
- WHERE CustState IN ('WA', 'TX') AND CustLastName NOT LIKE 'B';
- You can sew these together as much as you'd like, adding a series of conditions to your filter.



The screenshot shows the Microsoft SQL Server Management Studio interface. The query window displays the following T-SQL code:

```
SELECT * FROM Customers
WHERE CustState IN ('WA', 'TX')
AND
CustLastName NOT LIKE 'B';
```

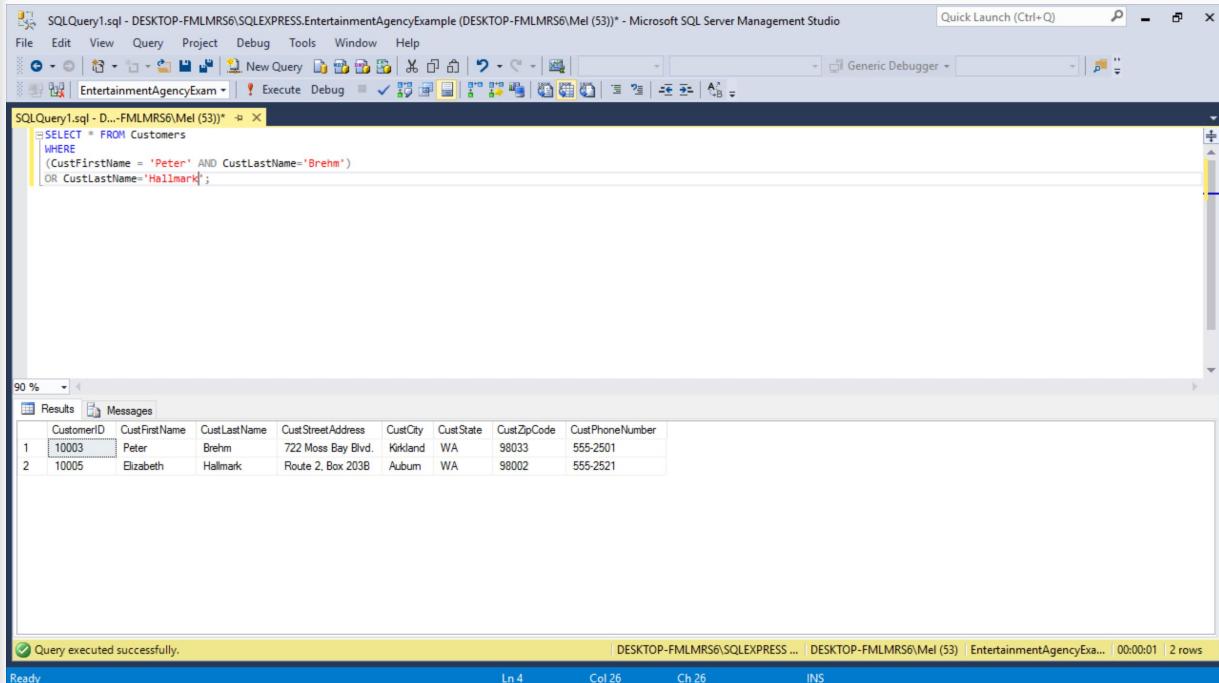
The results grid shows 15 rows of customer data from the 'Customers' table, filtered by state ('WA' or 'TX') and last name not starting with 'B'. The columns are CustomerID, CustFirstName, CustLastName, CustStreetAddress, CustCity, CustState, CustZipCode, and CustPhoneNumber. The data includes entries for Hartwig, Waldal, Brehm, McCrae, Elizabeth Hallmark, Matt Berg, Keyser, Darren Gehring, Sarah Thompson, Zachary Ehrlich, and Joyce Bonnicksen.

CustomerID	CustFirstName	CustLastName	CustStreetAddress	CustCity	CustState	CustZipCode	CustPhoneNumber
10001	Doris	Hartwig	4726 - 11th Ave. N.E.	Seattle	WA	98105	555-2671
10002	Deb	Waldal	908 W. Capital Way	Tacoma	WA	98413	555-2496
10003	Peter	Brehm	722 Moss Bay Blvd.	Kirkland	WA	98033	555-2601
10004	Dean	McCrae	4110 Old Redmond Rd.	Redmond	WA	98052	555-2506
10005	Elizabeth	Hallmark	Route 2, Box 203B	Auburn	WA	98002	555-2521
10006	Matt	Berg	908 W. Capital Way	Tacoma	WA	98413	555-2581
10007	Liz	Keyser	13920 S.E. 40th Street	Bellevue	WA	98006	555-2556
10008	Daren	Gehring	2601 Seaview Lane	Kirkland	WA	98033	555-2616
10009	Sarah	Thompson	2222 Springer Road	Bellevue	WA	98006	555-2626
10010	Zachary	Ehrlich	12330 Kingman Drive	Kirkland	WA	98033	555-2721
10011	Joyce	Bonnicksen	2424 Thaines Drive	Bellevue	WA	98006	555-2726

At the bottom of the results grid, a message indicates: "Query executed successfully." The status bar at the bottom of the screen shows "Ready", "Ln 4", "Col 27", "Ch 27", and "INS".

Combining AND and OR

- You can use parentheses to clarify your WHERE clause expressions.
- Like in other languages, whatever is inside parentheses is evaluated first.
- WHERE (CustFirstName = 'Peter' AND CustLastName='Brehm') OR CustLastName='Hallmark';



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery1.sql - DESKTOP-FMLMRS6\Mel (53)* - Microsoft SQL Server Management Studio' contains the following SQL code:

```
SELECT * FROM Customers
WHERE
(CustFirstName = 'Peter' AND CustLastName='Brehm')
OR CustLastName='Hallmark';
```

The results pane displays the following data:

CustomerID	CustFirstName	CustLastName	CustStreetAddress	CustCity	CustState	CustZipCode	CustPhoneNumber
1	Peter	Brehm	722 Moss Bay Blvd.	Kirkland	WA	98033	555-2501
2	Elizabeth	Hallmark	Route 2, Box 203B	Auburn	WA	98002	555-2521

The status bar at the bottom indicates 'Query executed successfully.' and shows the execution time as 00:00:01 for 2 rows.

Many Ways to Say the Same Thing

BETWEEN vs >= / <=

- These 2 statements evaluate to the same result. Which you choose to use is a matter of preference, just don't let seeing these on a test or in someone else's code throw you off.

```
SELECT * FROM Customers  
WHERE CustZipCode  
BETWEEN 90000 AND 99999;
```

```
SELECT * FROM Customers  
WHERE CustZipCode >= 90000 AND  
CustZipCode <= 99999;
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The top window is titled "SQLQuery1.sql - DESKTOP-FMLMRS6\SQLEXPRESS.EntertainmentAgencyExample (DESKTOP-FMLMRS6\Mel (53)) - Microsoft SQL Server Management Studio". It contains two SQL queries:

```
SELECT * FROM Customers  
WHERE CustZipCode >= 90000  
AND CustZipCode <= 99999;  
/* SAME AS */  
SELECT * FROM Customers  
WHERE CustZipCode BETWEEN 90000 AND 99999;
```

The bottom window is titled "Results" and displays the results of the query. It shows five rows of customer data:

CustomerID	CustFirstName	CustLastName	CustStreetAddress	CustCity	CustState	CustZipCode	CustPhoneNumber
1	Doris	Hartwig	4726 - 11th Ave. N.E.	Seattle	WA	98105	555-2671
2	Deb	Waldal	908 W. Capital Way	Tacoma	WA	98413	555-2496
3	Peter	Brehm	722 Moss Bay Blvd.	Kirkland	WA	98033	555-2501
4	Dean	McCrae	4110 Old Redmond Rd.	Redmond	WA	98052	555-2506
5	Elizabeth	Hallmark	Route 2, Box 203B	Auburn	WA	98002	555-2521

Below the results, a message bar indicates: "Query executed successfully." and "DESKTOP-FMLMRS6\SQLEXPRESS ... DESKTOP-FMLMRS6\Mel (53) EntertainmentAgencyExam... 00:00:00 30 rows".

NOT with AND / OR

- You can change the order of NOT a bit so that it comes before the predicate like shown on the right.
- WHERE NOT (CustFirstName = 'Peter' AND CustLastName='Brehm') AND CustLastName='Hallmark';
- Essentially, you can write NOT like this:
 - WHERE *CustFirstName NOT IN ('Sara', 'Bill');*
OR like this
 - WHERE **NOT** *CustFirstName IN ('Sara', 'Bill');*

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery1.sql - DESKTOP-FMLMRS6\SQLEXPRESS.EntertainmentAgencyExample (DESKTOP-FMLMRS6\Mel (53))' contains the following SQL code:

```
SELECT * FROM Customers
WHERE
NOT (CustFirstName = 'Peter' AND CustLastName='Brehm')
AND CustLastName='Hallmark';
```

The results pane displays a single row of data from the 'Customers' table:

CustomerID	CustFirstName	CustLastName	CustStreetAddress	CustCity	CustState	CustZipCode	CustPhoneNumber
1	Elizabeth	Hallmark	Route 2, Box 203B	Auburn	WA	98002	555-2521

At the bottom of the results pane, a message indicates: 'Query executed successfully.'

IN vs multiple = / ORs

- These 2 statements evaluate to the same result. Which you choose to use is a matter of preference, just don't let seeing these on a test or in someone else's code throw you off.

```
SELECT * FROM Customers  
WHERE CustZipCode  
IN (98105, 98413);
```

```
SELECT * FROM Customers  
WHERE CustZipCode = 98105 OR  
CustZipCode = 98413;
```

The screenshot shows the Microsoft SQL Server Management Studio interface. At the top, the title bar reads "SQLQuery1.sql - DESKTOP-FMLMRS6\SQLEXPRESS EntertainmentAgencyExample (DESKTOP-FMLMRS6\Mel (53)) - Microsoft SQL Server Management Studio". Below the title bar is the toolbar with various icons. The main area contains two queries in the "SQLQuery1.sql" window:

```
SELECT * FROM Customers  
WHERE CustZipCode = 98105  
OR CustZipCode = 98413;  
/* SAME AS */  
SELECT * FROM Customers  
WHERE CustZipCode IN(98105, 98413);
```

Below the queries is the "Results" tab showing the output of the second query. The results are displayed in a table with columns: CustomerID, CustFirstName, CustLastName, CustStreetAddress, CustCity, CustState, CustZipCode, and CustPhoneNumber. There are three rows of data:

CustomerID	CustFirstName	CustLastName	CustStreetAddress	CustCity	CustState	CustZipCode	CustPhoneNumber
10001	Doris	Hartwig	4726 - 11th Ave. N.E.	Seattle	WA	98105	555-2671
10002	Deb	Waldal	908 W. Capital Way	Tacoma	WA	98413	555-2496
10006	Matt	Berg	908 W. Capital Way	Tacoma	WA	98413	555-2581

Below the table, a message bar indicates "Query executed successfully." and shows the status bar at the bottom right: "DESKTOP-FMLMRS6\SQLEXPRESS ... DESKTOP-FMLMRS6\Mel (53) EntertainmentAgencyExa... 00:00:00 | 6 rows".

Try It

Using the EntertainmentAgencyExample DB,
show me an alphabetical list of entertainers based in Redmond, Bellevue, or
Woodinville.

Try It

Using the SalesOrdersExample DB,
show me all the orders for customer number 1001.

Try It

Using the database SchoolSchedulingExample,
show me an alphabetical list of all the staff members and their salaries if they
make between \$40,000 and \$50,000 annually.

Try It

Using the database `BowlingLeagueExample`,
list the ID numbers of the teams that won one or more of the first ten matches
in game number 3.

Try It

Using the database RecipesExample Database,
list the recipes that have no notes.