# Lesson 2: SQL Basics (Part C)

December 27, 2016 

# 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

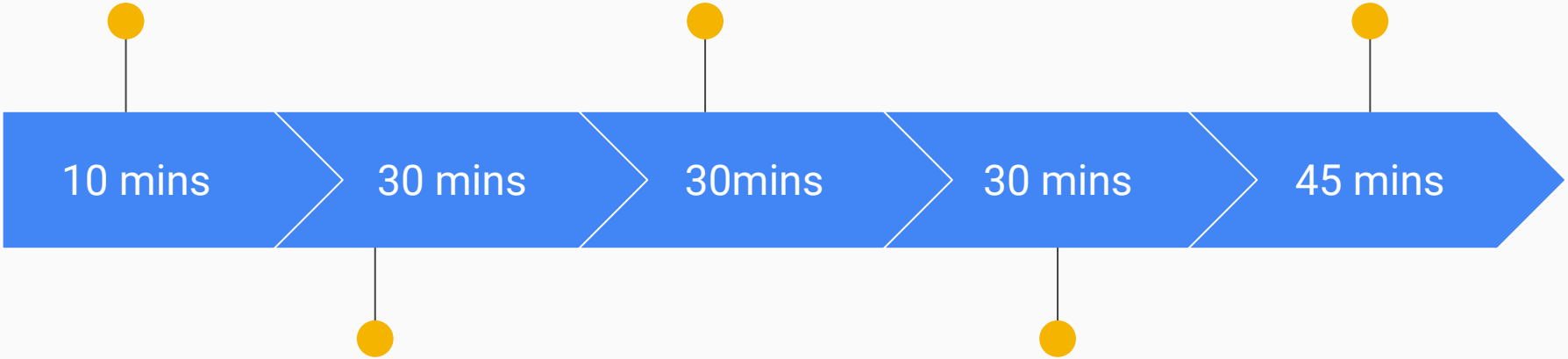
  - Online (sqlfiddle.com)
- Data
  - Test data on Github repo.
- Github repo for information

# Overview of Day 2

Basic SQL Queries (SELECT)

Expressions, Casting, NULL

Practice




SELECT, DISTINCT ORDER BY

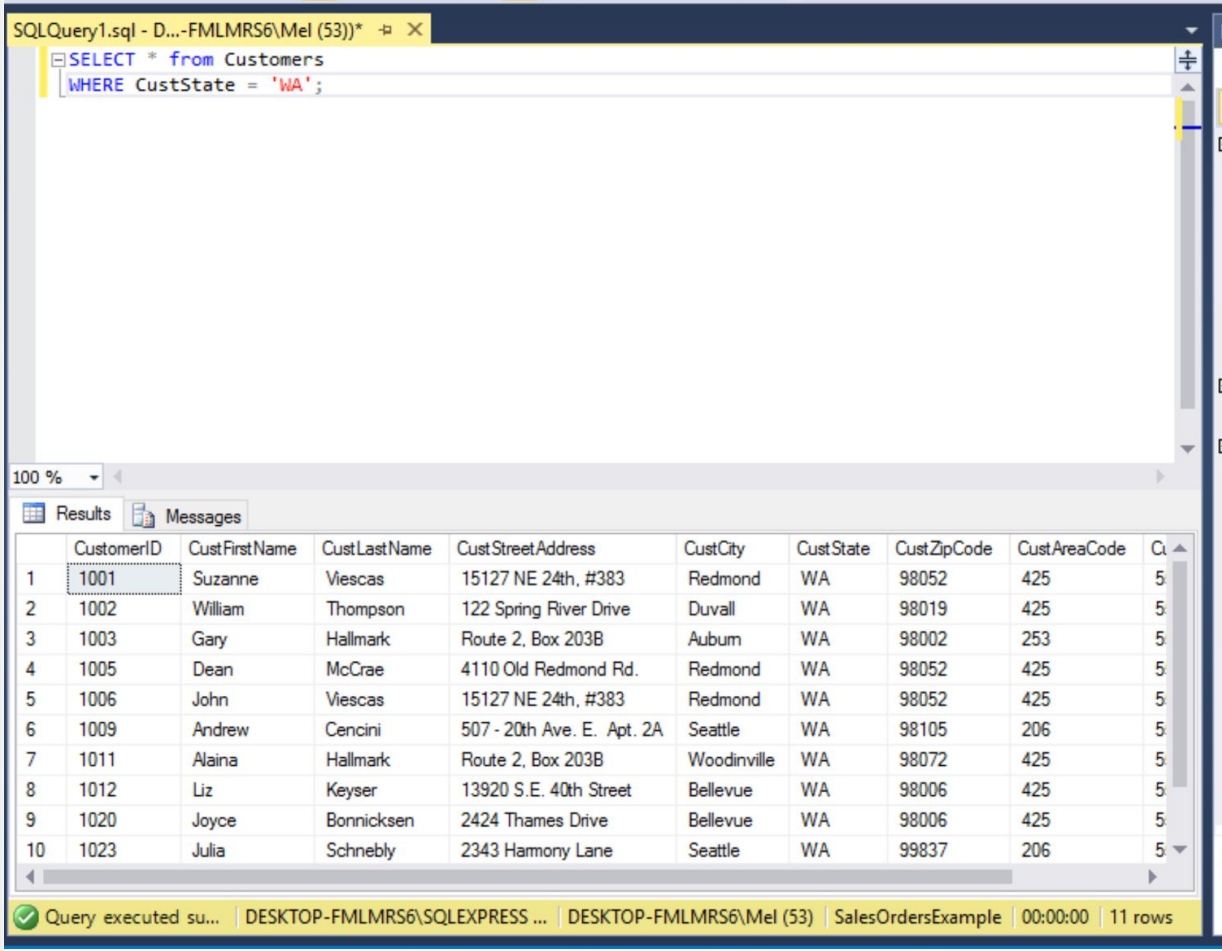
**Filtering using WHERE Clauses**

# 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.




| | CustomerID | CustFirstName | CustLastName | CustStreetAddress | CustCity | CustState | CustZipCode | CustAreaCode | Cu |
|---|---|---|---|---|---|---|---|---|---|
| 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 |

# 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

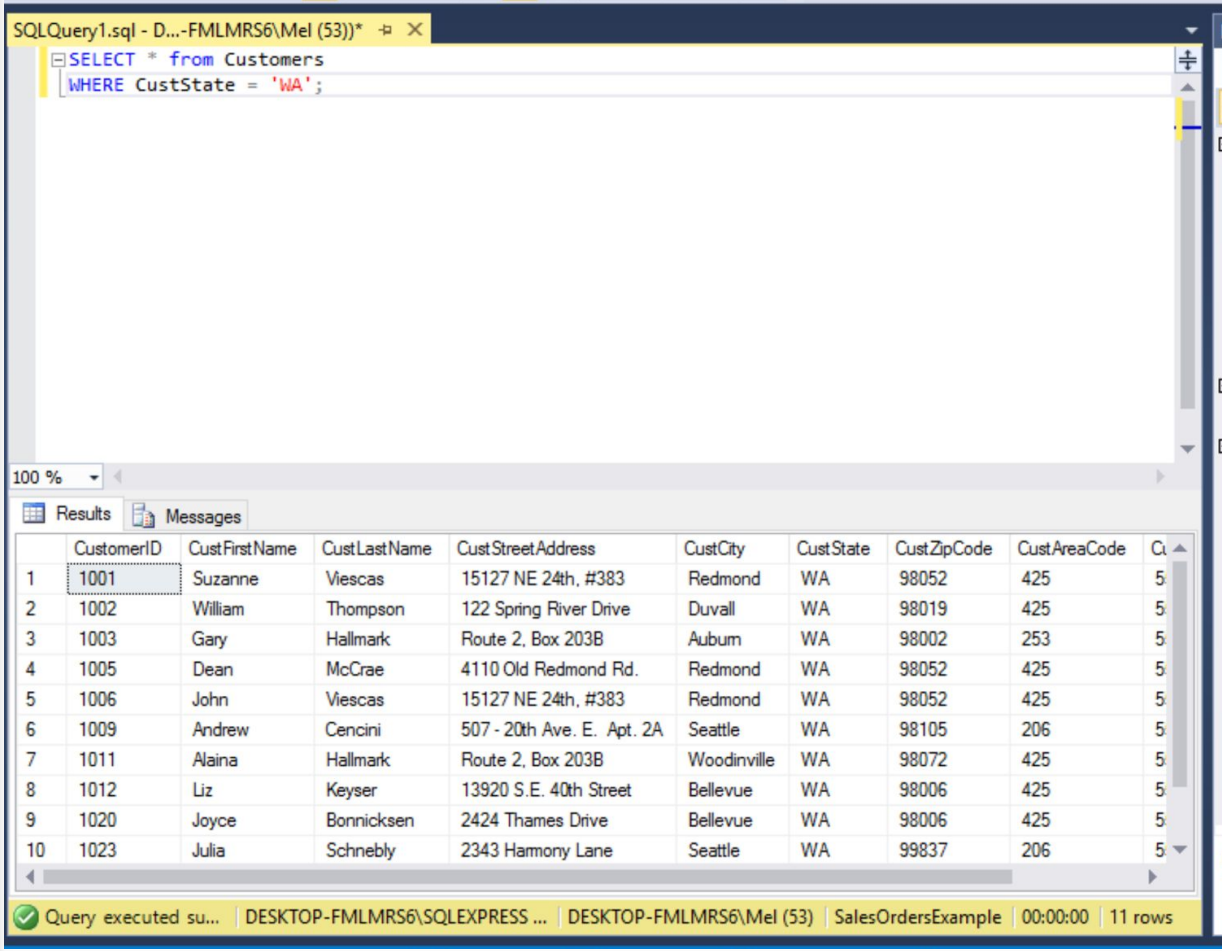
# Predicates

# 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: “=”

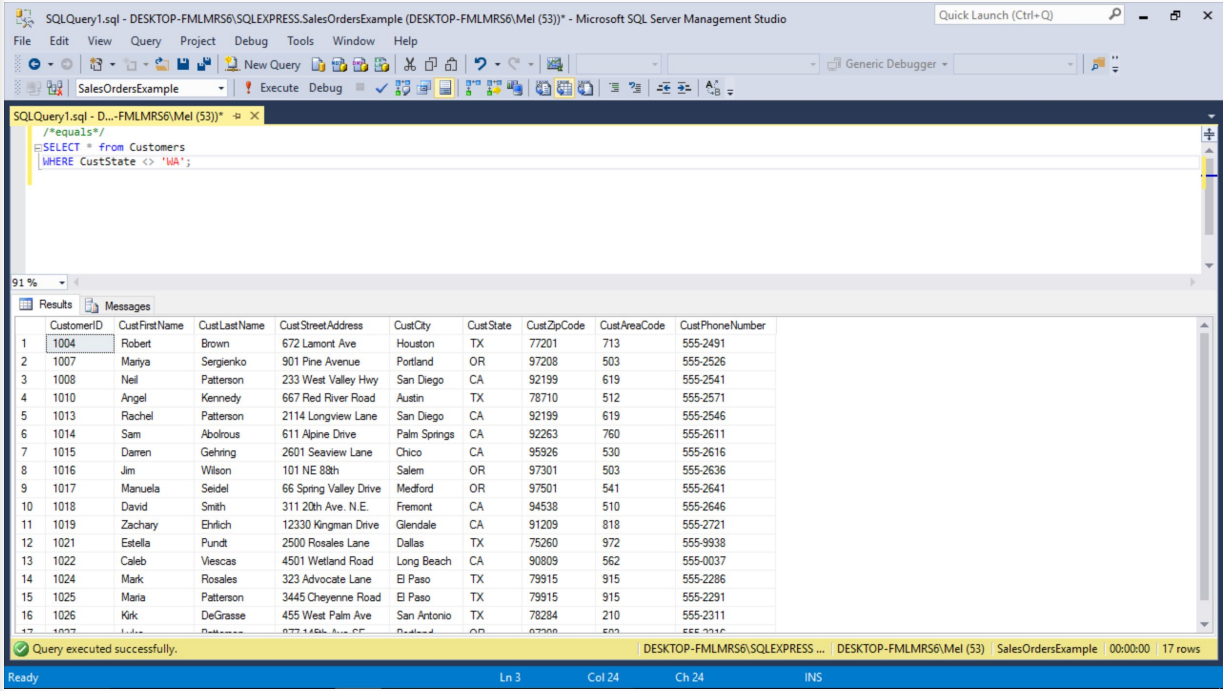
- Equal to.
- When you want a precise match.
- CustState = ‘WA’
- CustLastName = ‘Hallmark’
- CustomerID = 1001;
- CustAreaCode = 425;




| | CustomerID | CustFirstName | CustLastName | CustStreetAddress | CustCity | CustState | CustZipCode | CustAreaCode | Cu |
|---|---|---|---|---|---|---|---|---|---|
| 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 |

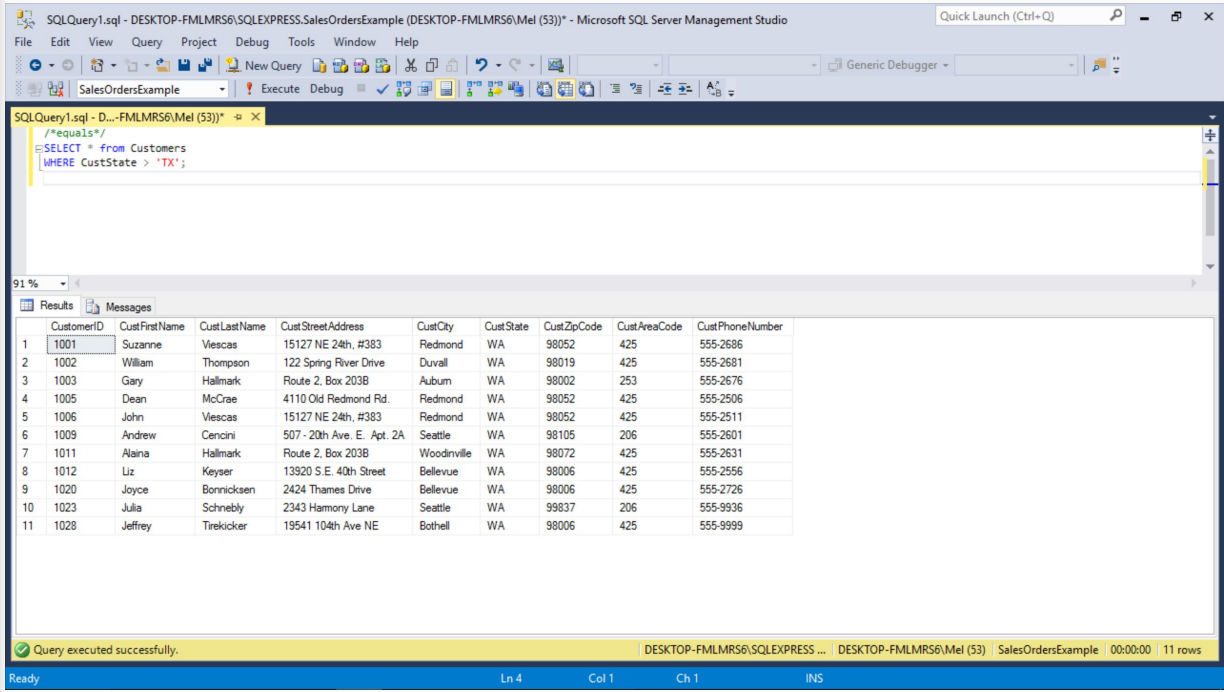
# Comparison Predicates: “<>”

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



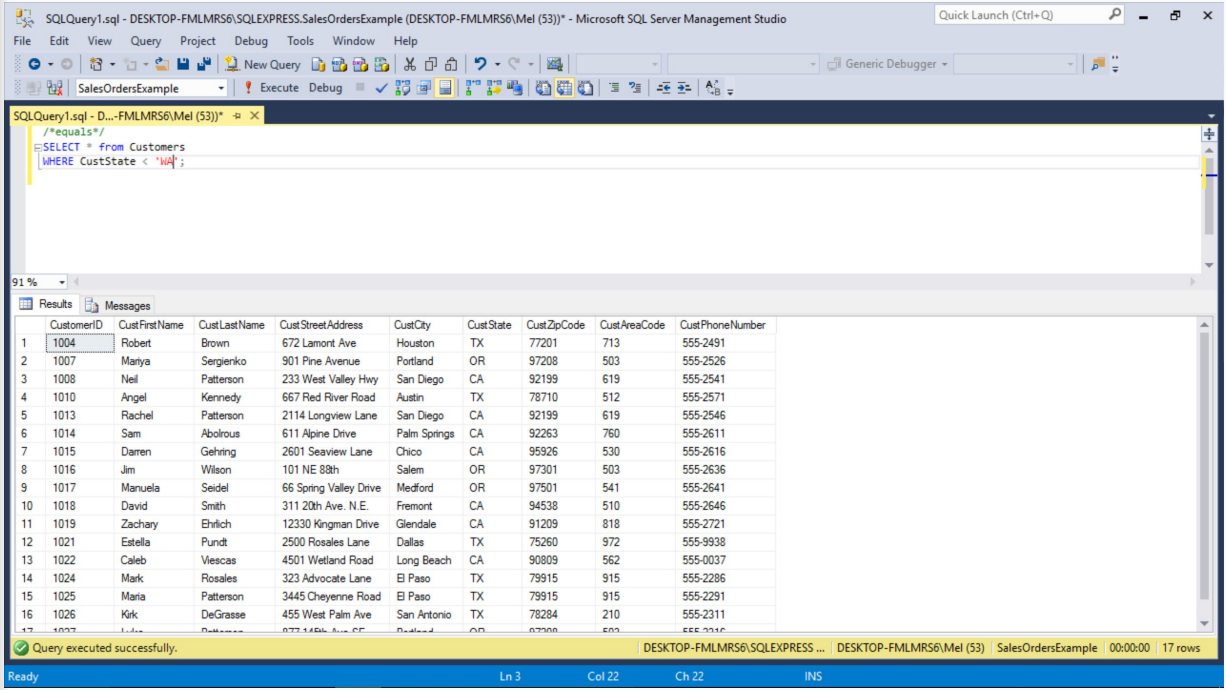
# Comparison Predicates: “>”

- 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;



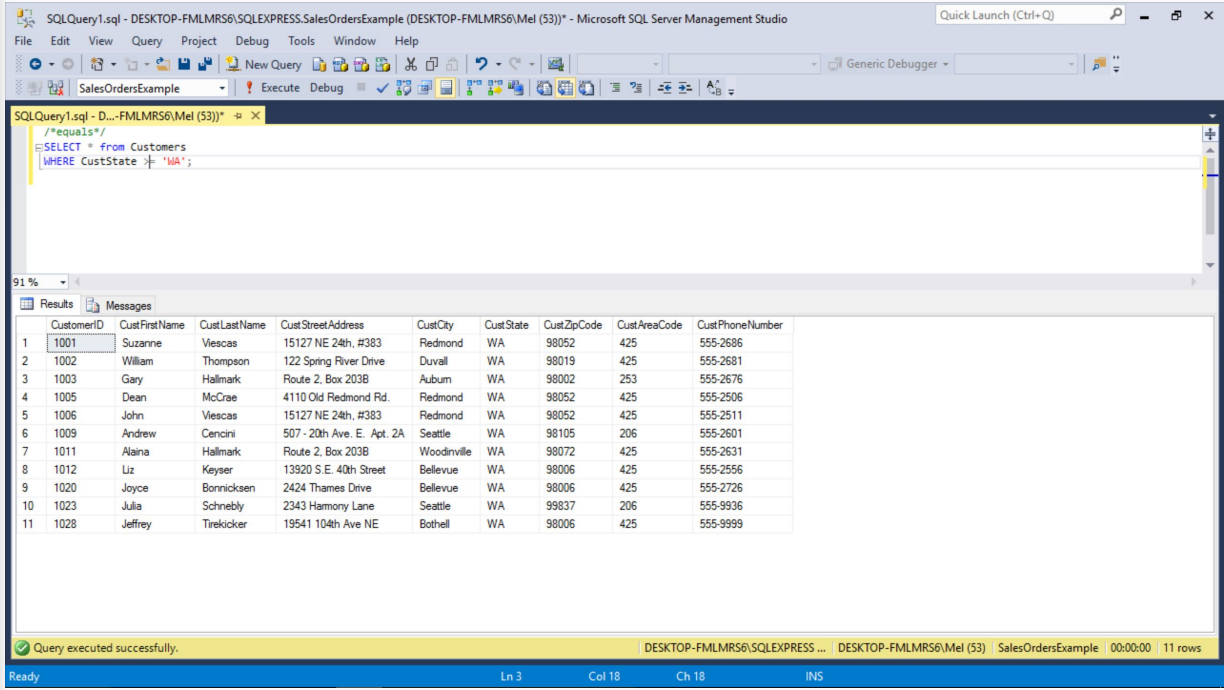
# Comparison Predicates: “<”

- 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;



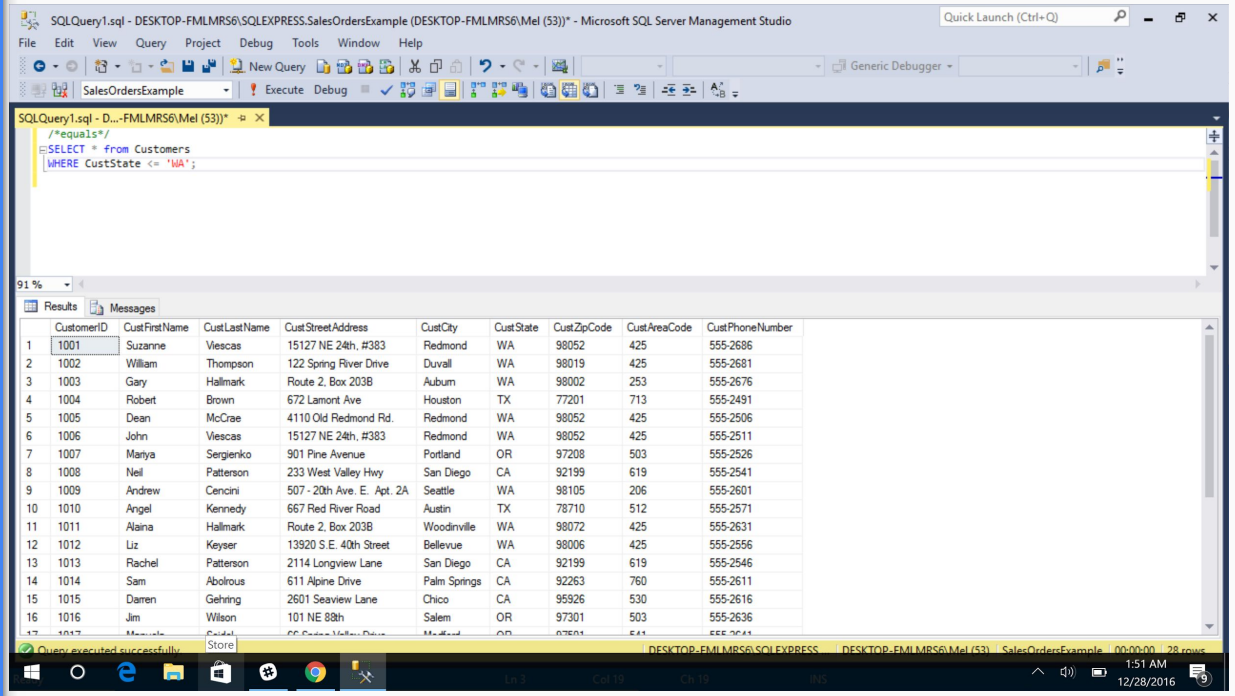
# Comparison Predicates: “>=”

- 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;



# Comparison Predicates: “<=”

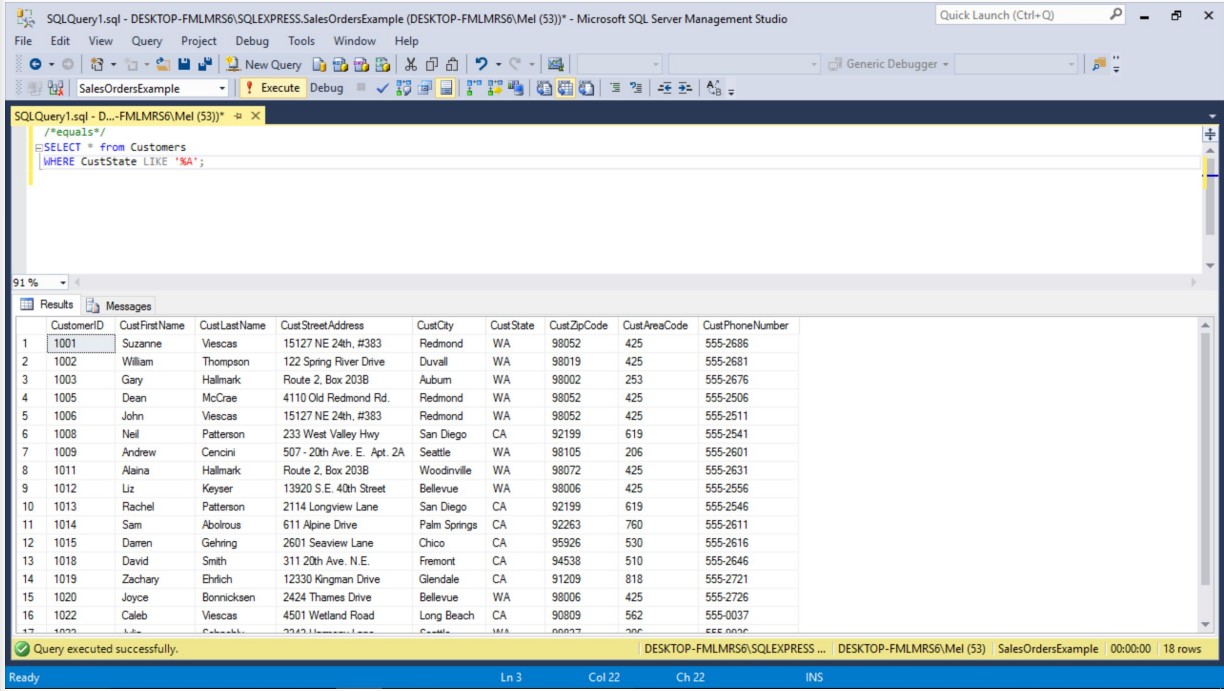
- Less than or equal to.
- When you want results where the attribute on the left is less than or equal to that of the right.
- CustState <= ‘WA’
- CustLastName <= ‘Hallmark’
- CustomerID <= 1001;
- CustAreaCode <= 425;




| | CustomerID | CustFirstName | CustLastName | CustStreetAddress | CustCity | CustState | CustZipCode | CustAreaCode | CustPhoneNumber |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1001 | Suzanne | Viescas | 15127 NE 24th, #383 | Redmond | WA | 98052 | 425 | 555-2686 |
| 2 | 1002 | William | Thompson | 122 Spring River Drive | Duvall | WA | 98019 | 425 | 555-2681 |
| 3 | 1003 | Gary | Hallmark | Route 2, Box 203B | Auburn | WA | 98002 | 253 | 555-2676 |
| 4 | 1004 | Robert | Brown | 672 Lamont Ave | Houston | TX | 77201 | 713 | 555-2491 |
| 5 | 1005 | Dean | McCrae | 4110 Old Redmond Rd. | Redmond | WA | 98052 | 425 | 555-2506 |
| 6 | 1006 | John | Viescas | 15127 NE 24th, #383 | Redmond | WA | 98052 | 425 | 555-2511 |
| 7 | 1007 | Mariya | Sergienko | 901 Pine Avenue | Portland | OR | 97208 | 503 | 555-2526 |
| 8 | 1008 | Neil | Patterson | 233 West Valley Hwy | San Diego | CA | 92199 | 619 | 555-2541 |
| 9 | 1009 | Andrew | Cencini | 507 - 20th Ave. E. Apt. 2A | Seattle | WA | 98105 | 206 | 555-2601 |
| 10 | 1010 | Angel | Kennedy | 667 Red River Road | Austin | TX | 78710 | 512 | 555-2571 |
| 11 | 1011 | Alaina | Hallmark | Route 2, Box 203B | Woodinville | WA | 98072 | 425 | 555-2631 |
| 12 | 1012 | Liz | Keyser | 13920 S.E. 40th Street | Bellevue | WA | 98006 | 425 | 555-2556 |
| 13 | 1013 | Rachel | Patterson | 2114 Longview Lane | San Diego | CA | 92199 | 619 | 555-2546 |
| 14 | 1014 | Sam | Abolrous | 611 Alpine Drive | Palm Springs | CA | 92263 | 760 | 555-2611 |
| 15 | 1015 | Darren | Gehring | 2601 Seaview Lane | Chico | CA | 95926 | 530 | 555-2616 |
| 16 | 1016 | Jim | Wilson | 101 NE 88th | Salem | OR | 97301 | 503 | 555-2636 |

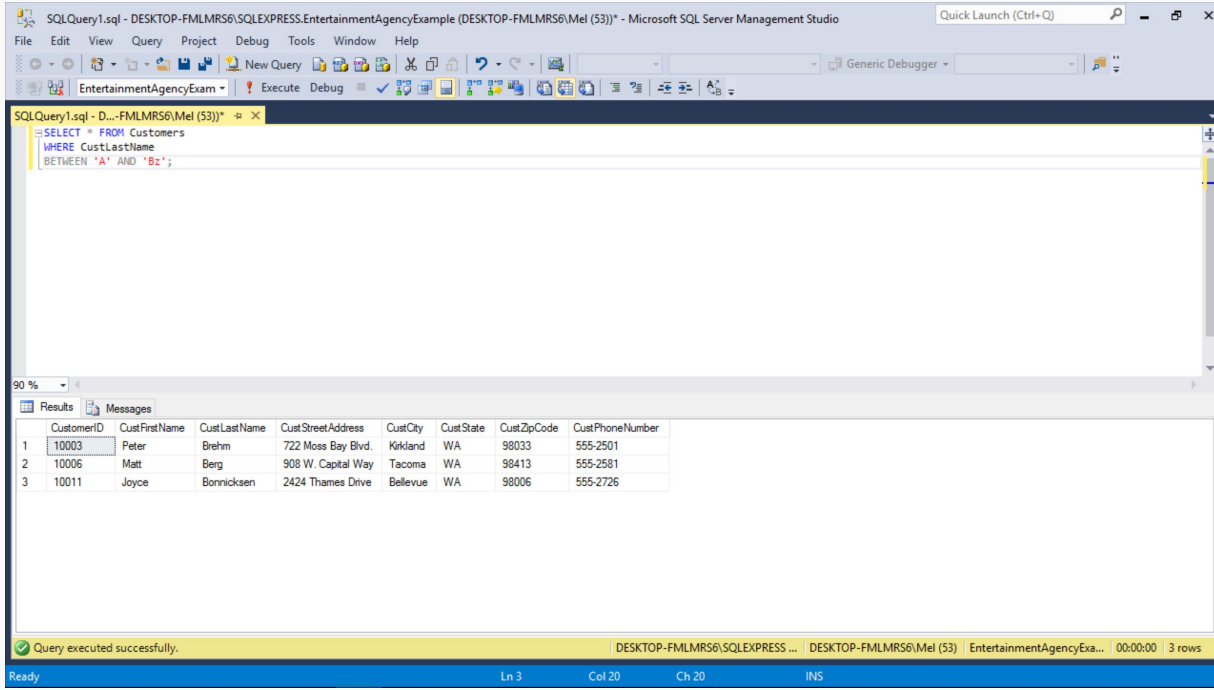
# LIKE Predicate

- 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%;



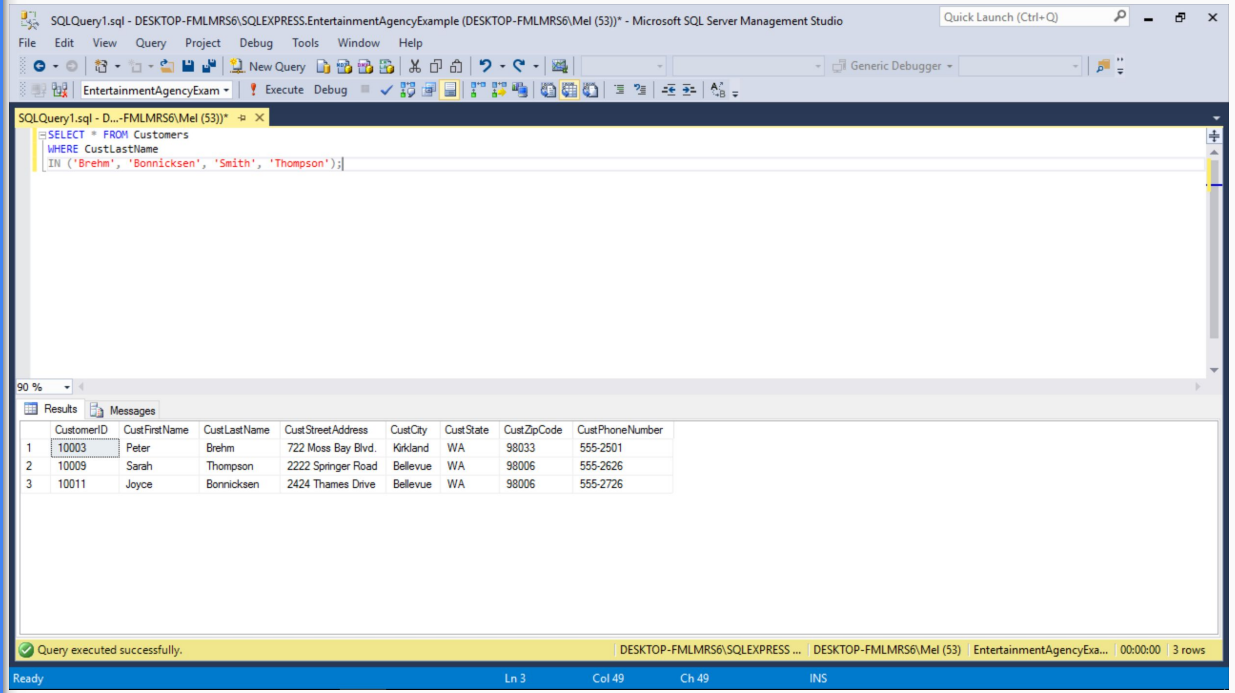
# BETWEEN Predicate

- 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 >= x AND something < = y.*
- CustState BETWEEN ‘A’ AND ‘Bz’;
- CustLastName  BETWEEN ‘s’ AND ‘Z’;
- CustomerID  BETWEEN 1000 AND 1099;
- CustAreaCode BETWEEN 400 AND 440;;



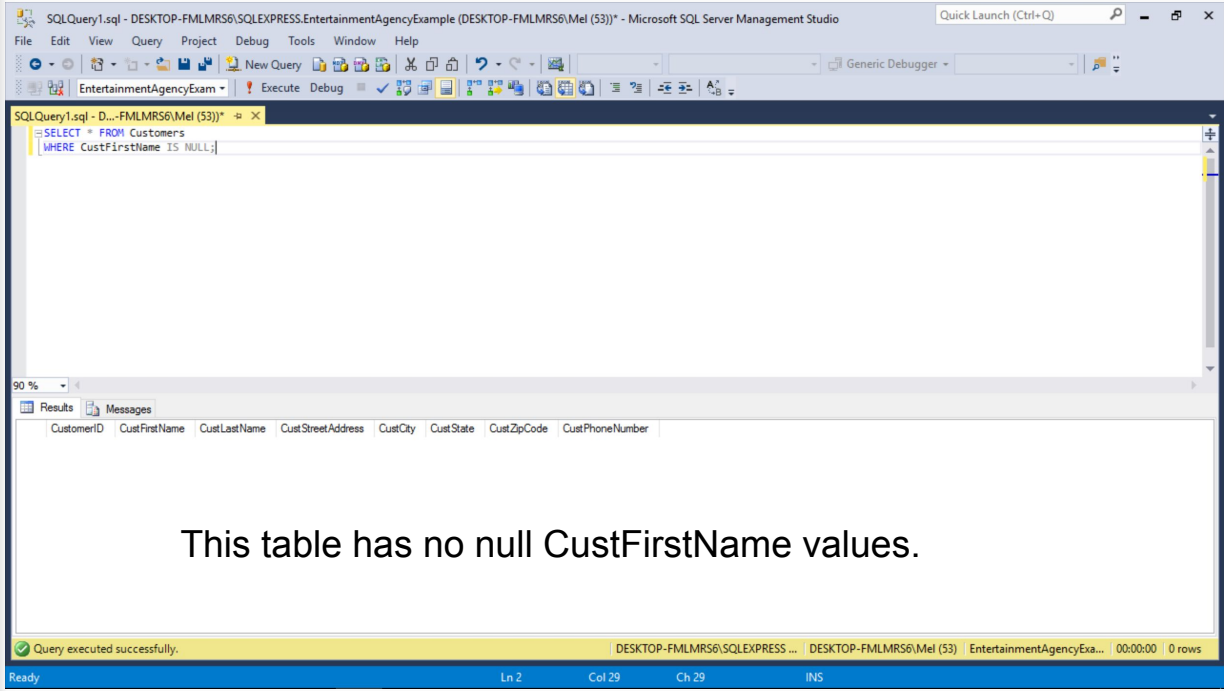
# IN Predicate

- 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);



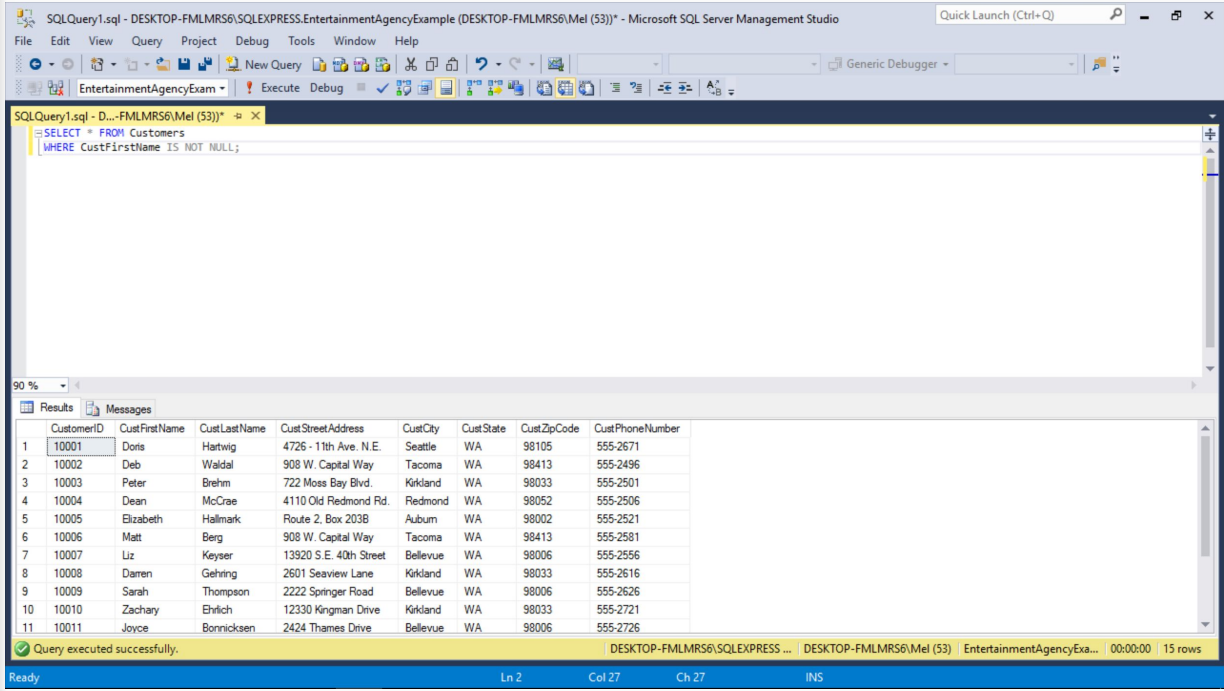
# IS NULL Predicate

- 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



# NOT Modifier

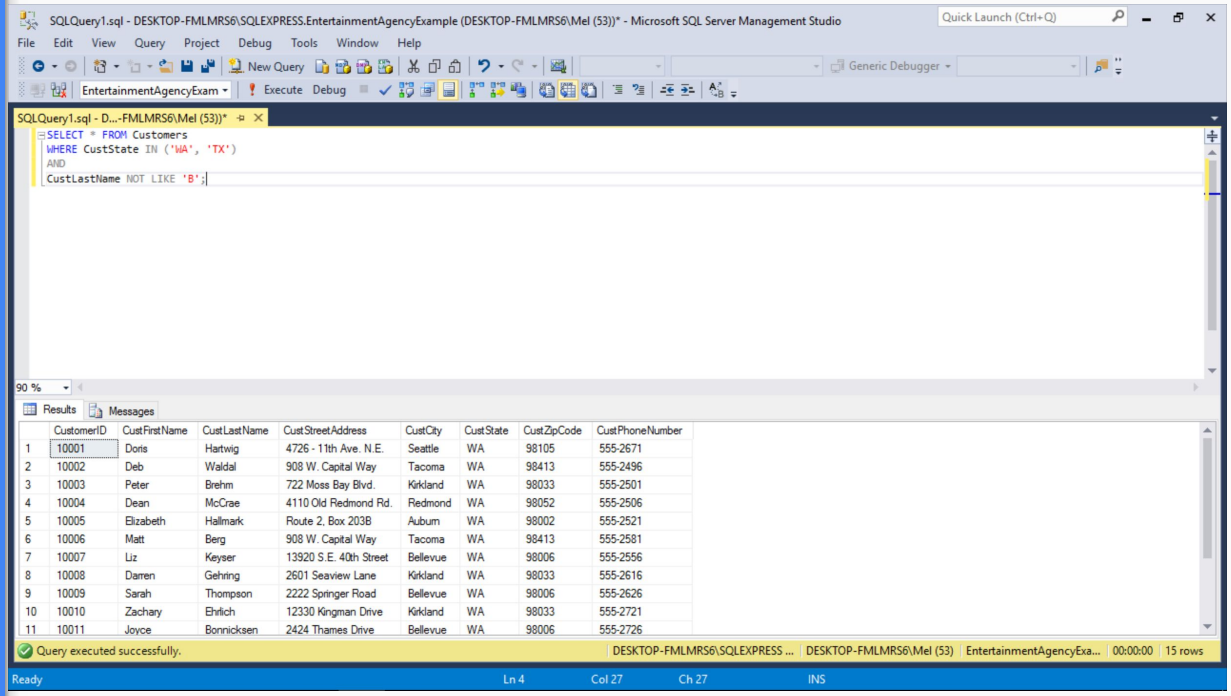
- With IN, LIKE, BETWEEN IS NULL, you can check for the inverse with the NOT modifier.
- CustState IS NOT NULL;
- CustLastName  IS NOT BETWEEN ‘A’ AND ‘Bz’;
- CustAreaCode IS NOT IN (440, 216, 614)



# Multiple Conditions

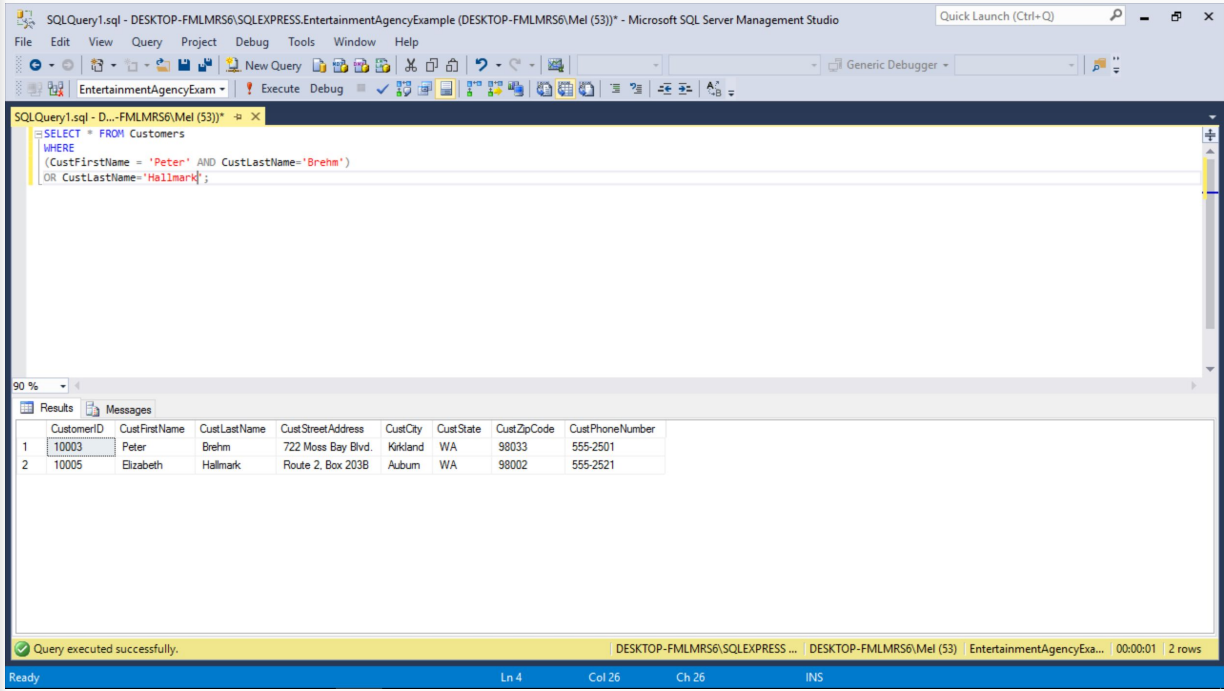
# Multiple Conditions Overview

- 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.



# 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';



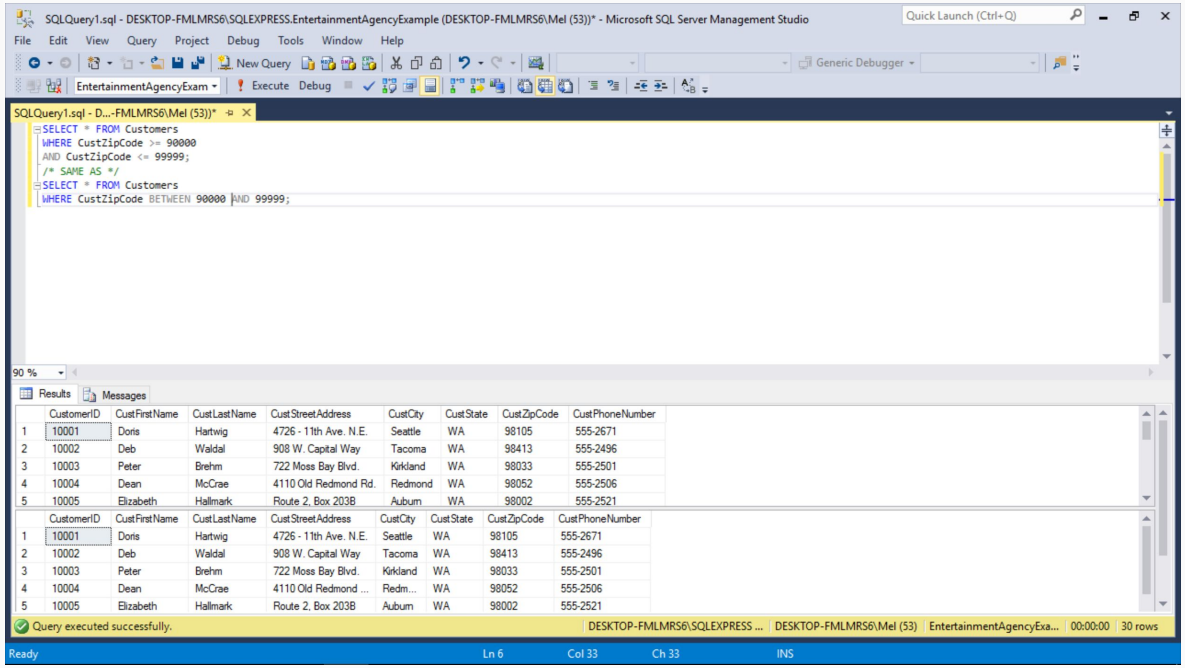
# Many Ways to Say the Same Thing

# 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.

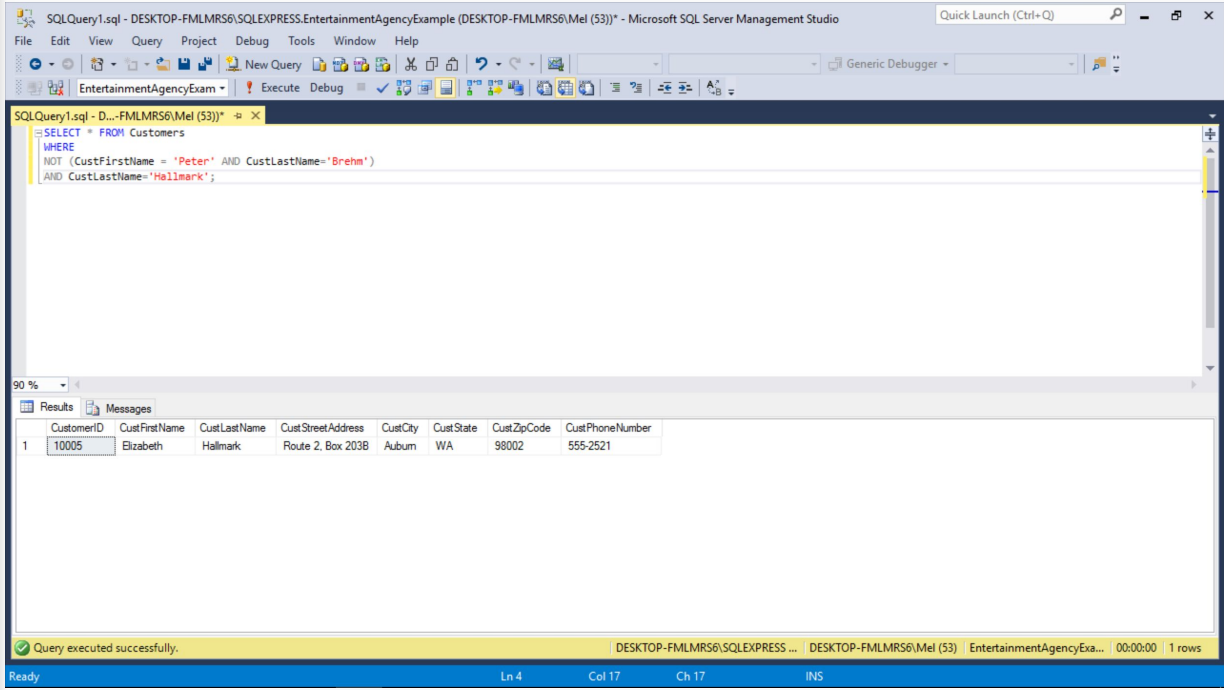
```
SELECT * FROM Customers
WHERE CustZipCode
BETWEEN 90000 AND 99999;
```

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



# 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');*

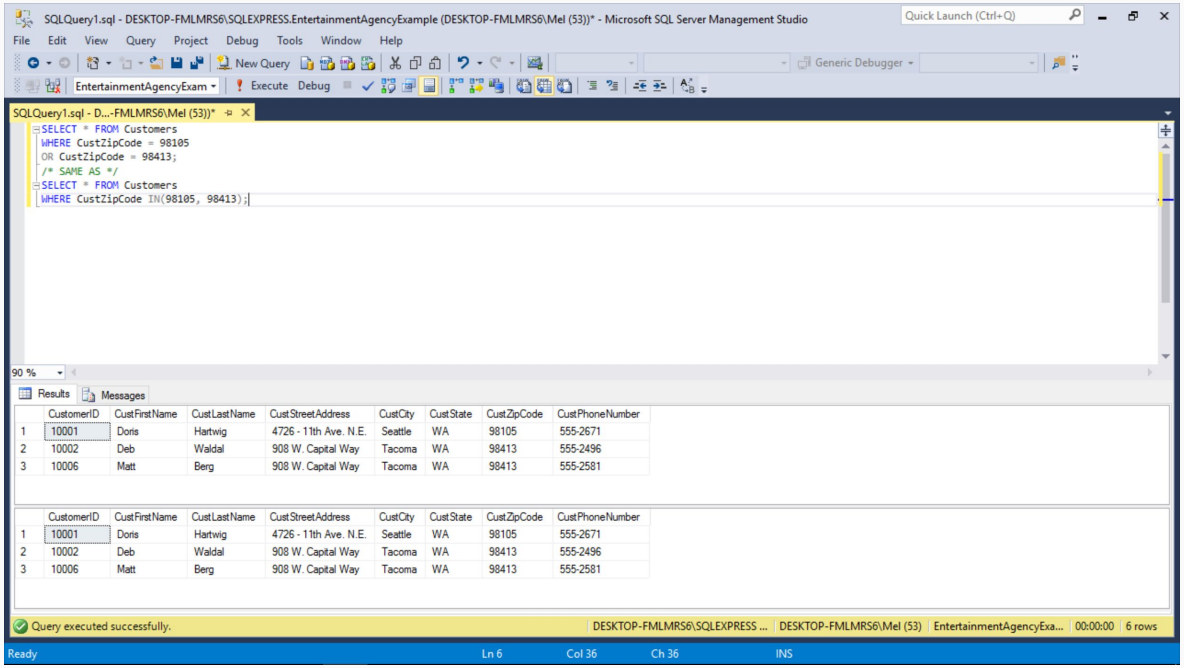


# IN vs multiple = / ORs

- 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;
```



# 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.