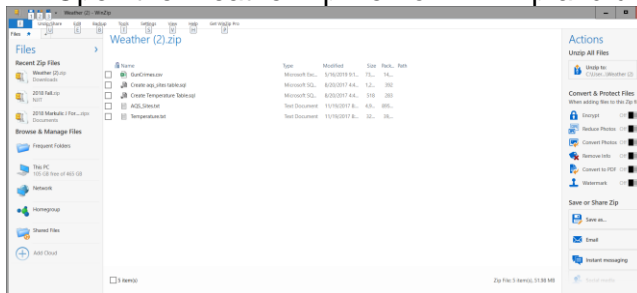


Step 1 – Import the Weather Database into you DBMS Using SSMS

See Step 1A for using Azure Data Studio (ASD)

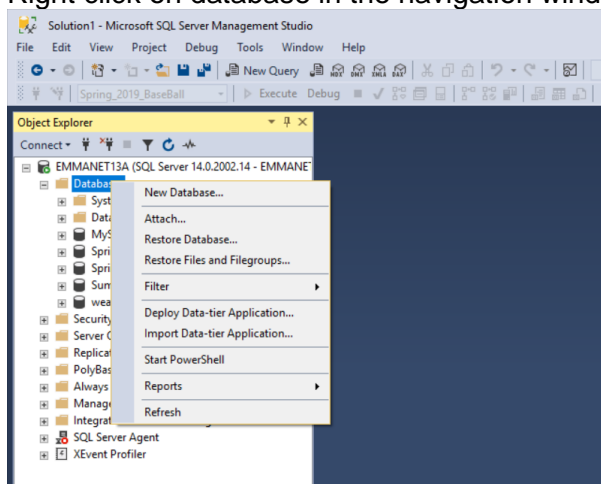
1. Download the Weather.zip file from the class [Virtual Machine Drive](#)
2. Open the Weather.zip file from Winzip and unzip the files to your P



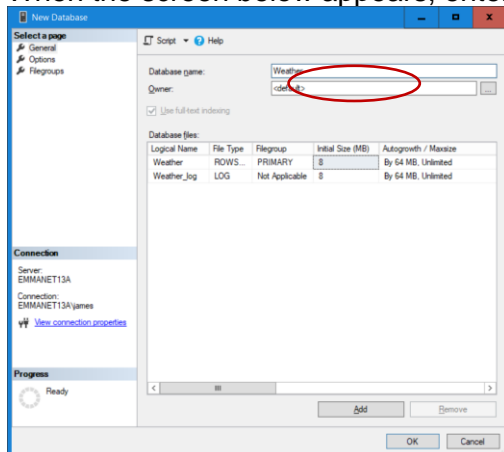
3. Open SQL Server Management Studio.

Note: a recorded copy of the following steps can be found at [Cisco Webex Meetings - Replay Recorded Meeting](#)

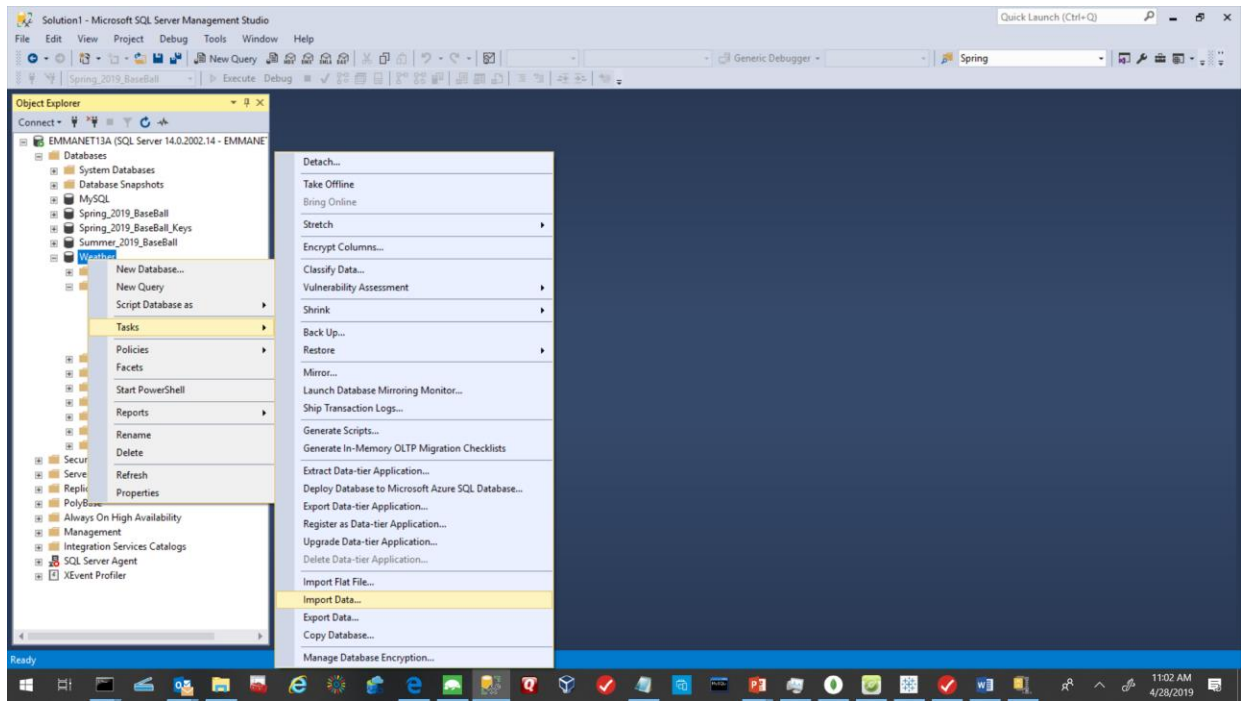
4. Right click on database in the navigation window and select **New Database**



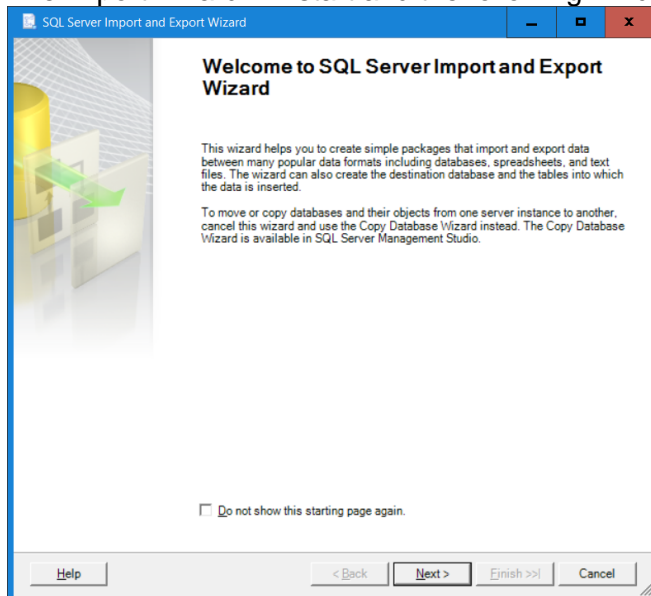
5. When the screen below appears, enter Weather as shown in the circle below and press the OK button



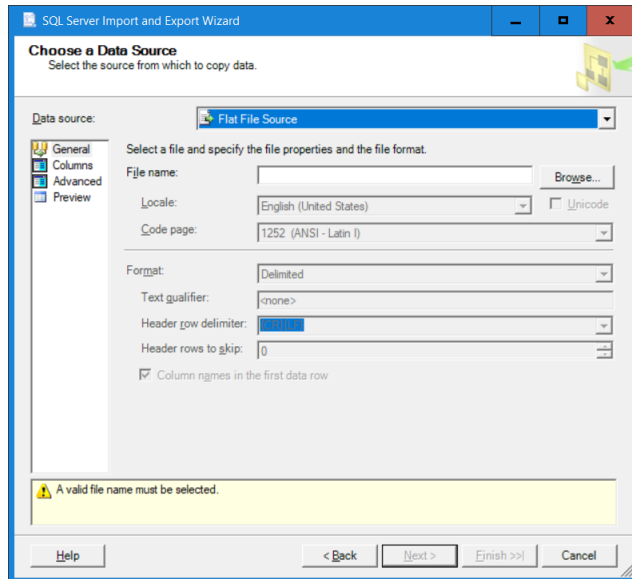
6. Next right click the Weather database and select Tasks and then Import Data as shown below



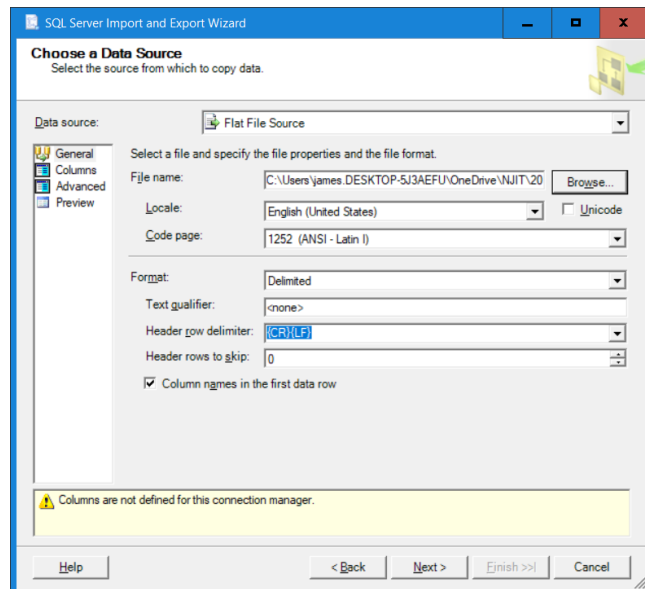
7. The import wizard will start and the following window will open. Click the NEXT button



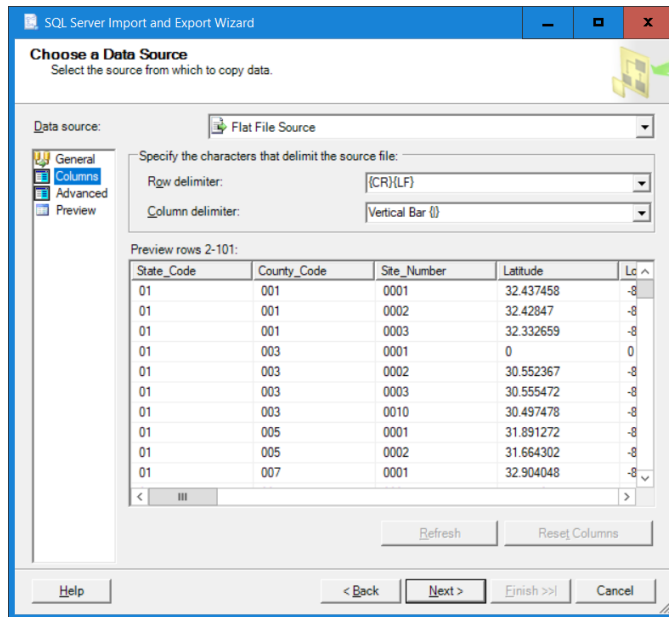
8. When the window below opens, select FLAT FILE as the data source and the window will expand to have the following additional selections



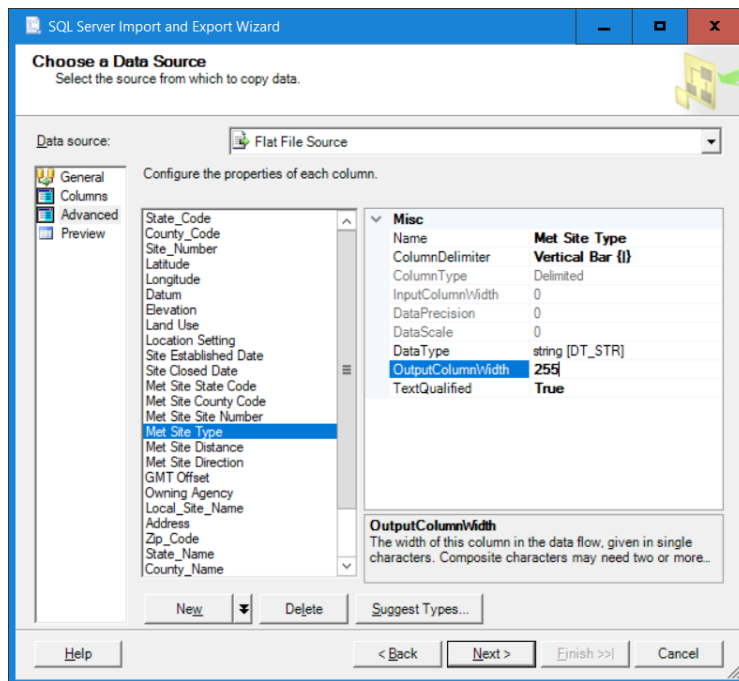
9. Click the Browse button and navigate to where you unzipped the zip file in step 2 and select the AQS_Sires.txt file as shown below



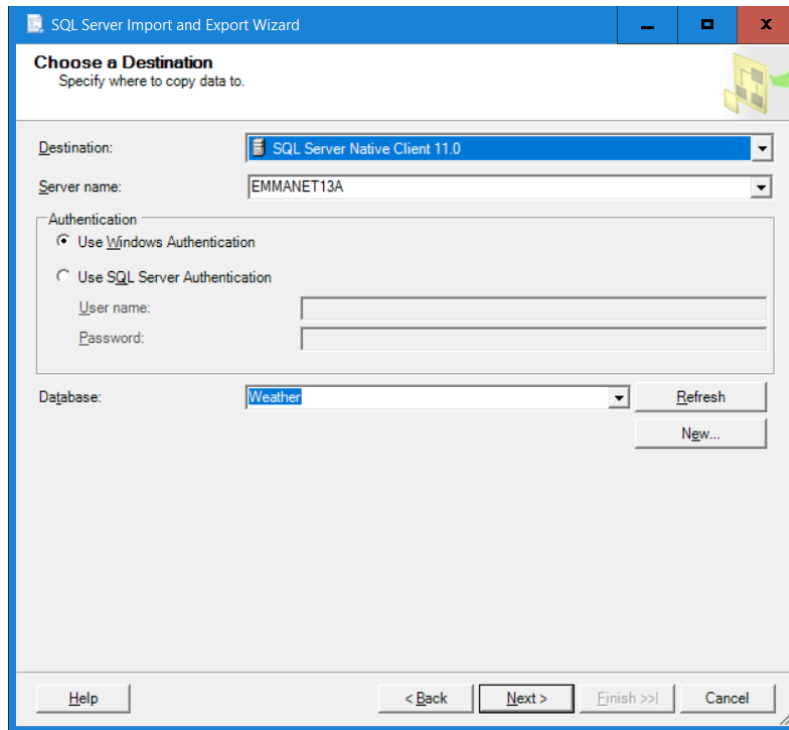
10. Select the COLUMNS option in the left hand navigation tab to make sure the columns have been properly identified. The screen should look like the following



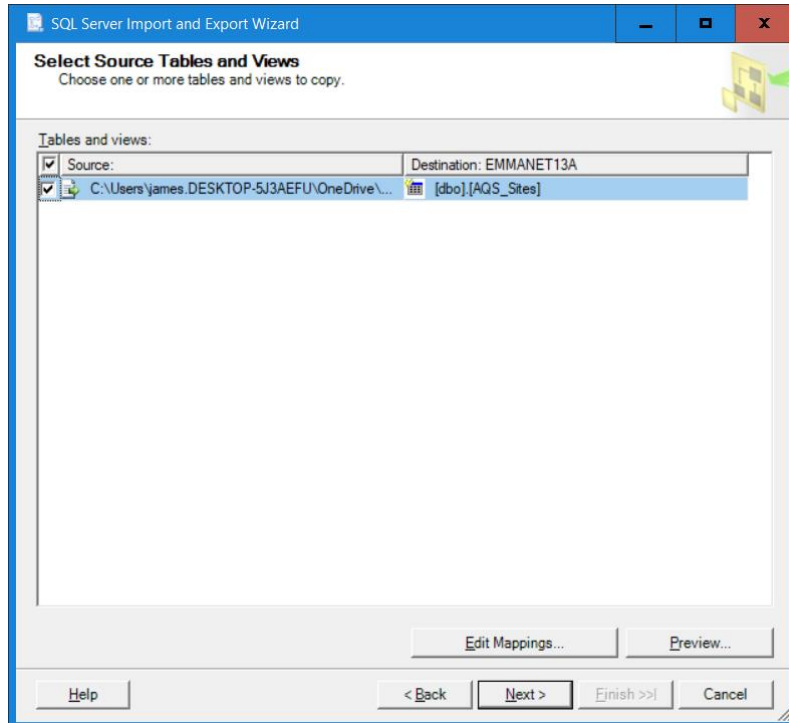
11. Next select the ADVANCED option and the following screen will appear. Once it does, you need to change the OUTPUT COLUMN WIDTH to 255 by click on the original width (shown as 50) and overwriting it with 255. This must be done for the Met_Site_Type, Met_Site_Direction, Owning_Agency, Local_Site_Name, Address, City Name, CBSA_Name and Tribe_Name and then click the NEXT button



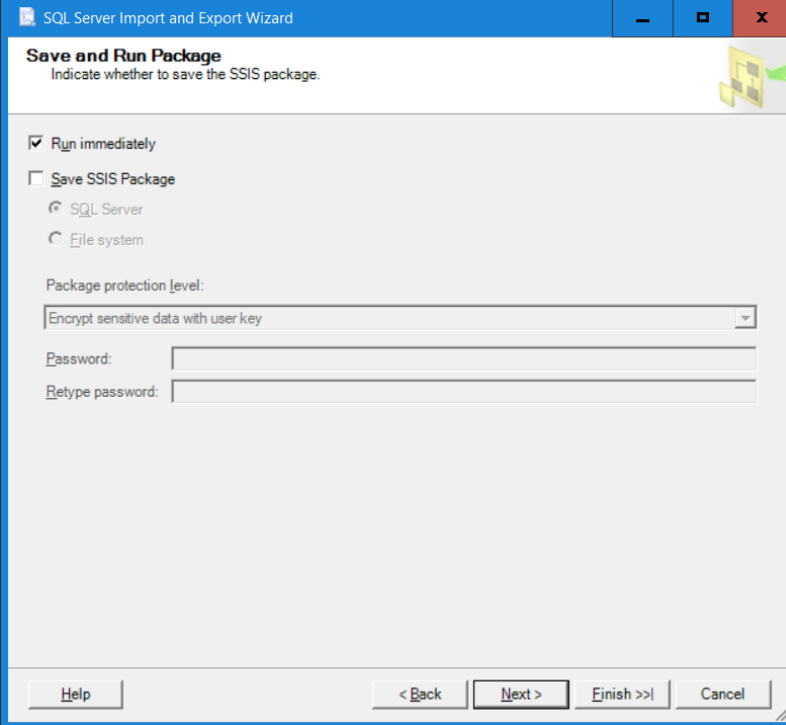
12. Next, change the Destination to SQL Server Native Client 11.0 and make sure the Server and Authentication information is correct and then hit NEXT



13. Hit NEXT when the following window appears

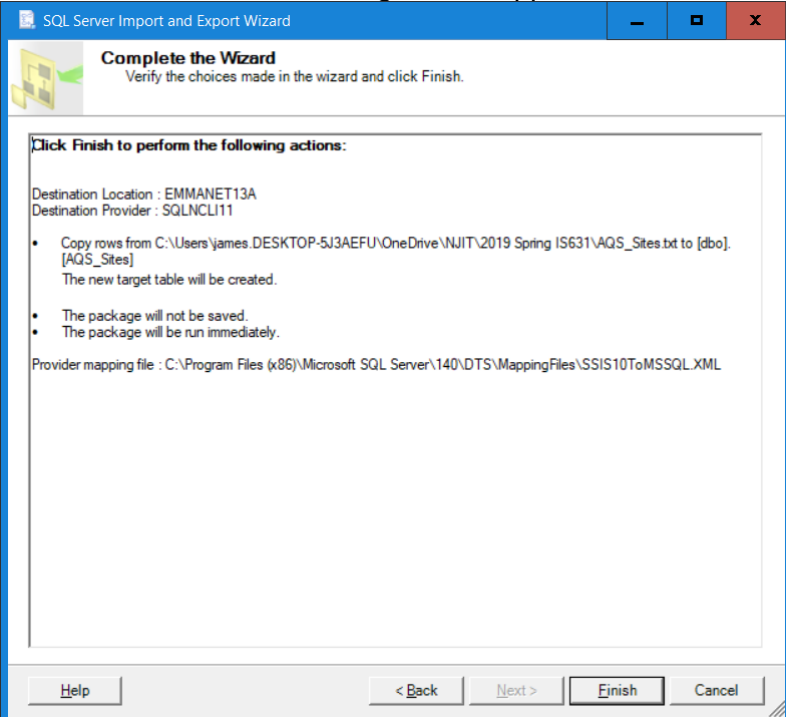


14. Hit the FINISH button when the following windows appears



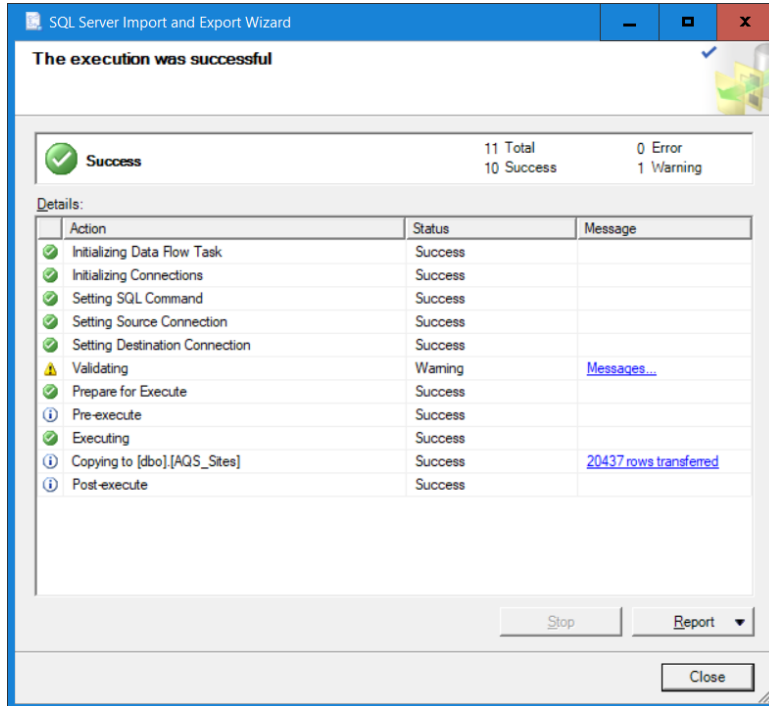
The screenshot shows the 'Save and Run Package' step of the SQL Server Import and Export Wizard. The window title is 'SQL Server Import and Export Wizard'. The main heading is 'Save and Run Package' with the instruction 'Indicate whether to save the SSIS package.' Below this, there are two radio buttons: 'Run immediately' (which is selected) and 'Save SSIS Package'. Under 'Save SSIS Package', there are two options: 'SQL Server' and 'File system'. Below these, there is a 'Package protection level:' section with a dropdown menu set to 'Encrypt sensitive data with user key'. There are also fields for 'Password:' and 'Retype password:'. At the bottom, there are buttons for 'Help', '< Back', 'Next >', 'Finish >>|', and 'Cancel'.

15. Click FINISH when the following screen appears

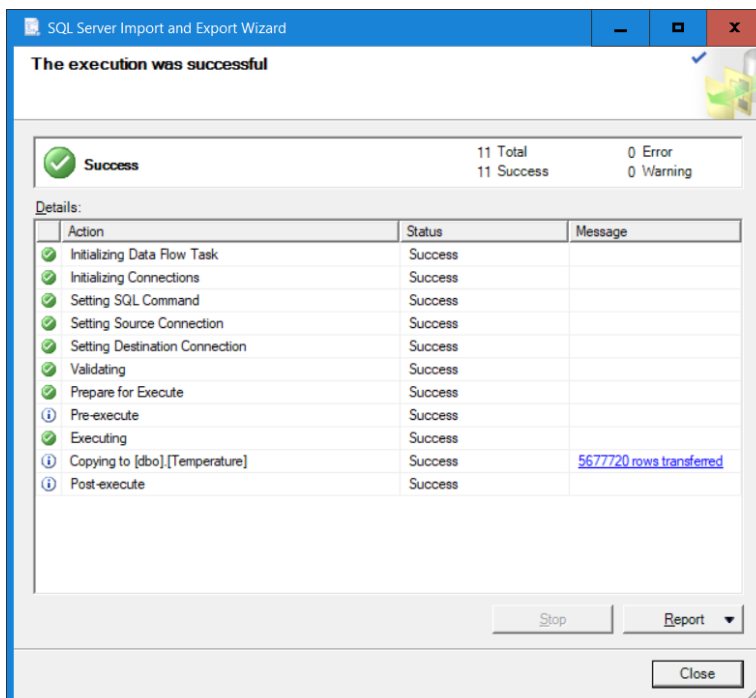


The screenshot shows the 'Complete the Wizard' step of the SQL Server Import and Export Wizard. The window title is 'SQL Server Import and Export Wizard'. The main heading is 'Complete the Wizard' with the instruction 'Verify the choices made in the wizard and click Finish.' Below this, there is a section titled 'Click Finish to perform the following actions:'. It lists the following actions: 'Destination Location : EMMANET13A', 'Destination Provider : SQLNCLI11', 'Copy rows from C:\Users\james.DESKTOP-5J3AEFU\OneDrive\NJIT\2019 Spring IS631\AQS_Sites.txt to [dbo].[AQS_Sites]. The new target table will be created.', 'The package will not be saved.', and 'The package will be run immediately.' Below this, there is a line for 'Provider mapping file : C:\Program Files (x86)\Microsoft SQL Server\140\DTS\MappingFiles\SSIS10ToMSSQL.XML'. At the bottom, there are buttons for 'Help', '< Back', 'Next >', 'Finish', and 'Cancel'.

16. Hit FINISH and the following screen will show the progress and have the below information when it finishes



17. Next, you must repeat steps 6 through 16 selecting the Temperature.txt file in Step 9 and there are no modifications that need to be made to column widths in Step 11.

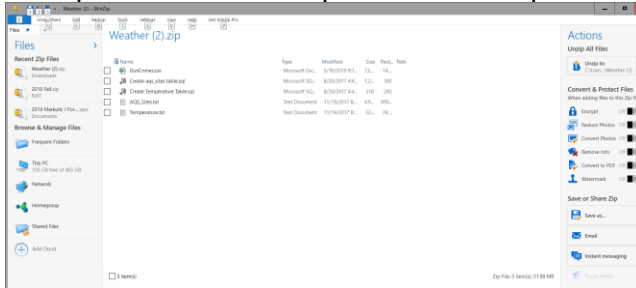


Note: There is no grade for this step. It is factored into the remaining Parts since you will not be able to complete them without being able to complete Part 1.

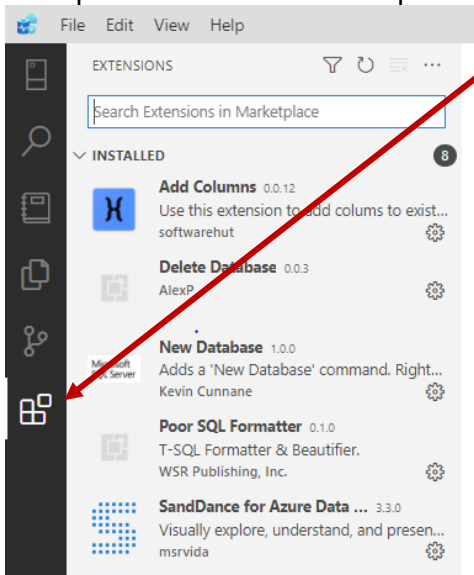
Step 1a – Import the Weather Database into you DBMS Using ADS

Go to Step 2 if using SSMS

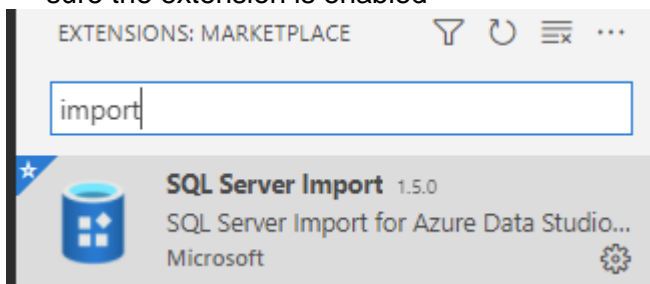
1. Download the Weather.zip file from the class [Virtual Machine Drive](#)
2. Open the Weather.zip file from Winzip and unzip the files to your PC



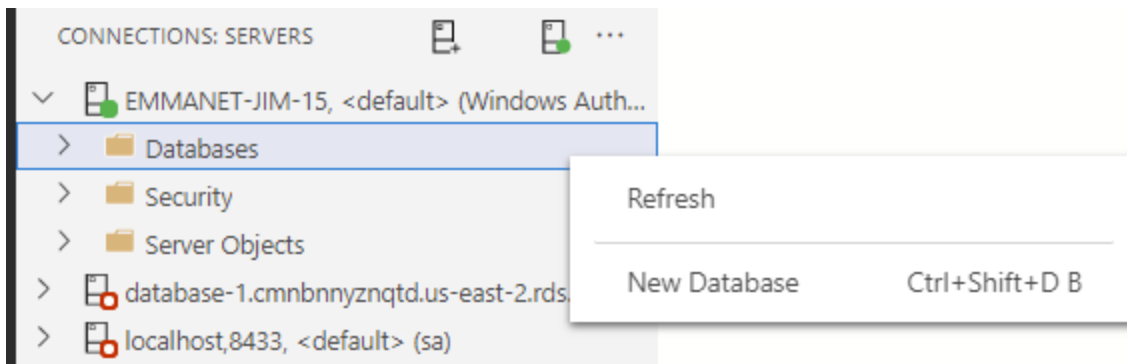
3. Open ASD and click the squares for Extensions



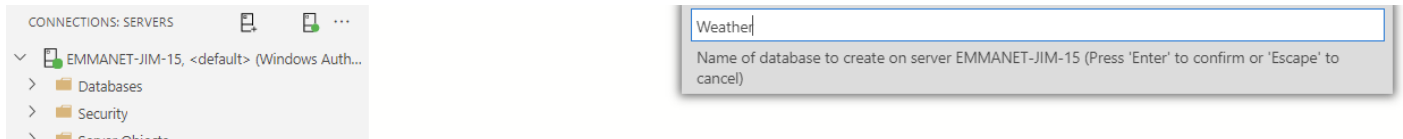
4. Make sure you installed the New Database extension as part of the ASD installation
5. Enter Import in the search bar and SQL Server Import extension will appear as shown below. Make sure the extension is enabled



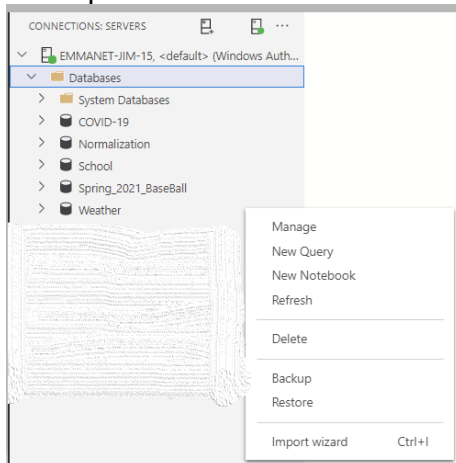
6. Navigate back to Connections (the top icon on the left and right click on Databases as shown below



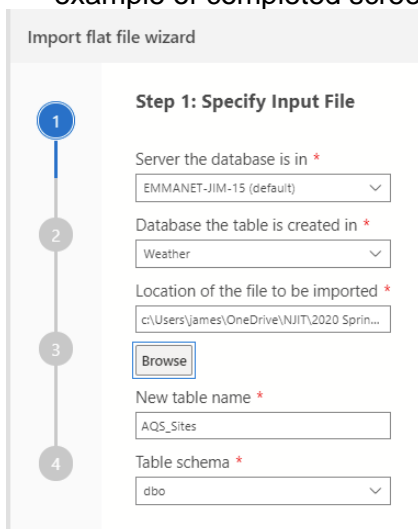
7. Select New Database and enter Weather for the Database Name



8. Expand the Database folder and right click the Weather Database and select Import Wizard



9. When the starting screen of the Wizard appears, fill in the server and database information, click browse and navigate to where you extracted the zip files, and select the AQS sites file (see below for example of completed screen). Click next at the bottom right hand portion of the window.



10. The Preview Data window will appear next. You do not have to change anything on this step, so click the next button.

1

Step 2: Preview Data

2

This operation analyzed the input file structure to generate the preview below for up to the first 50 rows.

3

State_Code	County_Code	Site_Number	Latitude	Longitude	Datum	Elevation	Land_Use	Location_Se
01	001	0001	32.437458	-86.472891	WGS84	64	RESIDENTIAL	SUBURBA
01	001	0002	32.42847	-86.443585	WGS84	0	AGRICULTURAL	RURAL
01	001	0003	32.332659	-86.791521	WGS84	41	FOREST	RURAL
01	003	0001	0	0	NAD27	0	UNKNOWN	RURAL
01	003	0002	30.552367	-87.706911	WGS84	0	COMMERCIAL	RURAL
01	003	0003	30.555472	-87.713599	WGS84	49	COMMERCIAL	URBAN A

11. When the Modify Columns window appears, select “Allow Nulls” at the top of the columns and change the Date Types shown below

Column Name	Data Type	Primary Key <input type="checkbox"/>	Allow Nulls <input checked="" type="checkbox"/>
State_Code	varchar(MAX) <input type="text"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
County_Code	varchar(MAX) <input type="text"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Site_Number	varchar(MAX) <input type="text"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Latitude	float <input type="text"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Longitude	float <input type="text"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Datum	nvarchar(50) <input type="text"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Elevation	float <input type="text"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Land_Use	nvarchar(50) <input type="text"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Location_Setting	nvarchar(50) <input type="text"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Site_Established_Date	date <input type="text"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Site_Closed_Date	date <input type="text"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Met_Site_State_Code	int <input type="text"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Met_Site_County_Code	int <input type="text"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Met_Site_Site_Number	int <input type="text"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Met_Site_Type	varchar(MAX) <input type="text"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Column Name	Data Type	Primary Key <input type="checkbox"/>	Allow Nulls <input checked="" type="checkbox"/>
Met_Site_Distance	float	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Met_Site_Direction	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
GMT_Offset	int	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Owning_Agency	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Local_Site_Name	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Address	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Zip_Code	int	<input type="checkbox"/>	<input checked="" type="checkbox"/>
State_Name	nvarchar(50)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
County_Name	nvarchar(50)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
City_Name	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
CBSA_Name	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Tribe_Name	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Extraction_Date	date	<input type="checkbox"/>	<input checked="" type="checkbox"/>

and then click the Import Data button

12. While the data is importing the following screen will appear and the circle will spin while the import is in progress. There is no progress bar, just the spinning circle


Import flat file wizard

Step 4: Summary

Import information

Object type	Name
Server name	localhost
Database name	Weather4
Table name	Temperature
Table schema	dbo
File to be imported	c:\Users\james\OneDrive\WJIT\2020 Spring IS631\Temperat...

Import status



13. The following screen will appear when the Import is complete

Step 4: Summary

Import information

Object type	Name
Server name	localhost
Database name	Weather4
Table name	AQS_Sites
Table schema	dbo
File to be imported	c:\Users\james\OneDrive\NJIT\2020 Spring\5631\AQS_Site...

Import status

✓ You have successfully inserted the data into a table.

14. Next Select the Import New File Button and Import the Temperature Text File repeating the steps above. For the Temperature table, modify the columns to have the data types shown below. Note the Temperature table is large and will take several minutes to import the 5 million rows.

Column Name	Data Type	Primary Key <input type="checkbox"/>	Allow Nulls <input checked="" type="checkbox"/>
State_Code	varchar(MAX) ▾	<input type="checkbox"/>	<input checked="" type="checkbox"/>
County_Code	varchar(MAX) ▾	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Site_Num	varchar(MAX) ▾	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Date_Local	date ▾	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Average_Temp	decimal(18, 10) ▾	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Daily_High_Temp	decimal(18, 10) ▾	<input type="checkbox"/>	<input checked="" type="checkbox"/>
_1st_Max_Hour	varchar(MAX) ▾	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Date_Last_Change	varchar(MAX) ▾	<input type="checkbox"/>	<input checked="" type="checkbox"/>

15. You can also import the Guncrimes.csv file now and skip that step in the Submission 3 Section. Modify the column data types shown below

Column Name	Data Type	Primary Key <input type="checkbox"/>	Allow Nulls <input checked="" type="checkbox"/>
incident_id	int	<input type="checkbox"/>	<input checked="" type="checkbox"/>
date	date	<input type="checkbox"/>	<input checked="" type="checkbox"/>
state	nvarchar(50)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
city_or_county	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
address	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
n_killed	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
n_injured	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
congressional_district	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
gun_stolen	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
gun_type	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
incident_characteristics	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
latitude	float	<input type="checkbox"/>	<input checked="" type="checkbox"/>
location_description	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
longitude	float	<input type="checkbox"/>	<input checked="" type="checkbox"/>
n_guns_involved	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
notes	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
participant_age	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
participant_age_group	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
participant_gender	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
participant_name	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
participant_relationship	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
participant_status	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
participant_type	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Last_Row	varchar(MAX)	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Database Description

1. The database contains 2 tables
 - a. Aqs_sites – contains information regarding the sites where the temperature information in the Temperature table was collected. All the column should be self explanatory. The linkage between the 2 tables use the State_Code, County_Code and Site_Number columns
 - b. Temperature – contains the daily temperature information collected at the site for 2 decades. The important columns for the assignment are:
 - i. State_Code, County_Code and Site_Number are used to join to the aqs_sites table
 - ii. Date_Local – The date that the sample was collected
 - iii. Average_Temp – The average temperature for that particular date
 - iv. Daily_High_Temp – The highest temperature for the day
 - v. All temperatures are in degrees Fahrenheit
2. **Suggestion:** Due to the large number of records in the database, your queries may take several minutes to execute. If your queries are taking a long time to run, I suggest you make a table containing a small subset of the data in the Temperature table to use for writing and debugging your queries. After they all execute successfully, you can change the queries to use the full Temperature table
3. **Grading** – You will receive separate grades for Part 3, Part 4 and Part 5 (if you complete Part 5 for the extra credit). Both parts 3 and 4 will be worth 100 points each and each question in the step will be worth a proportionate value (1/# of questions in the step) Parts 3 and 4 will make up 50% of the Project grade. The extra credit (Part 5) will be worth 15% extra.

Creating Geospatial Data

1. Your last concern is how long will it take to travel back home to visit friends and family after you move. Since the Weather database has latitude and longitude information, you have decided to convert this information into a new column with a GEOGRAPHY data type and populate the new column with a set command and one of the following formula (Depending on the data type for latitude and longitude) The example below is for when latitude and longitude are numeric:

```
Use Weather
go
IF NOT EXISTS(
    SELECT *
    FROM sys.columns
    WHERE Name = N'GeoLocation'
    AND Object_ID = Object_ID(N'AQS_Sites'))
BEGIN
    ALTER TABLE AQS_Sites ADD GeoLocation Geography NULL
END
go
UPDATE [dbo].[aqs_sites]
SET [GeoLocation] = geography::STPointFromText('POINT(' + CAST([LONGITUDE] AS VARCHAR(max))
    + ' ' + CAST([Latitude] AS VARCHAR(max)) + ')', 4326)
WHERE [LATITUDE] > 0 and latitude <> ''
```

Submission 1 – Problems

You are trying to decide where in the US to reside. The most important factor to you is temperature, you hate cold weather. Answer the following questions to help you make your decision. For all problems show all columns included in the examples. Note that the term temperature applies to the average daily temperature unless otherwise stated.

- Find the minimum, maximum and average of the average temperature column for each state sorted by state name.

State_Name	Minimum Temp	Maximum Temp	Average Temp
Alabama	-4.662500	88.383333	59.328094
Alaska	-43.875000	80.791667	29.146757
Arizona	-99.000000	135.500000	67.039050

- The results from question #2 show issues with the database. Obviously, a temperature of -99 degrees Fahrenheit in Arizona is not an accurate reading as most likely is 135.5 degrees for Delaware. Write a query to count all the suspect temperatures (below -39° and above 105°). Sort your output by State_Name, state_code, County_Code, and Site_Number

State_Name	state_code	County_Code	Site_Number	Num_Bad_Entries
Alaska	02	090	0034	2
Alaska	02	188	0004	1
Arizona	04	001	0007	129
Arizona	04	001	0008	150

- You noticed that the average temperatures become questionable below -39 ° and above 125 ° and that it is unreasonable to have temperatures over 105 ° for state codes 30, 29, 37, 26, 18, 38. You also decide that you are only interested in living in the United States, not Canada or the US territories. Create a view that combines the data in the AQS_Sites and Temperature tables. The view should have the appropriate SQL to exclude the data above. You should use this view for all subsequent queries. My view returned 5,616,127 rows. The view includes the State_code, State_Name, County_Code, Site_Number. **Make sure you include schema binding in your view for later problems.**

If you want to verify your view is correct, display the count of rows grouped by state. Please do not use a select * since it runs for a long time. The results would look like the following:


State_code	row_count
01	5,221
02	14,077
04	11,3325

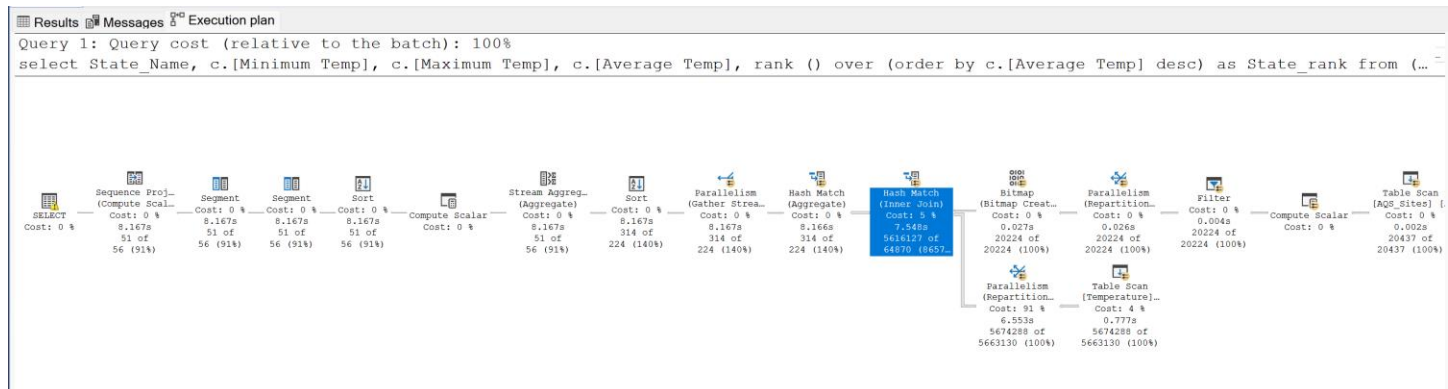
Unless specified, the remainder of the queries should use your view instead of the AQS_Sites and Temperature tables.

- Using the SQL RANK statement, rank the states by Average Temperature

State_Name	Minimum Temp	Maximum Temp	Average Temp	State_rank
Florida	35.9583330000	88.0000000000	73.3473286338	1
Texas	-1.1250000000	122.6000000000	68.7936401301	2
Mississippi	22.2250000000	91.1583330000	68.4937457796	3


5. At this point, you've started to become annoyed at the amount of time each query is taking to run. If you

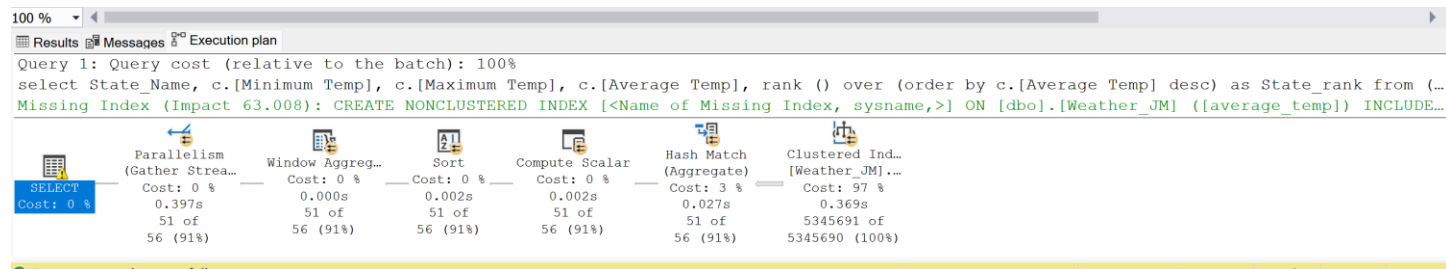
click the  button and execute the query you wrote for #4, you will see something like the execution plan below.



You've heard that creating indexes can speed up queries. Create an index for your **view** (not the underlying tables). You are required to create **a single index** with the unique and clustered parameters and the index will be on the State_Code, County_Code, Site_Number, **average_temp**, and Date_Local columns. **DO NOT** create the index on the tables, the **index** must be created on the VIEW.

6. There are 270,511 duplicate rows that you must delete before you can create a unique index. Use the Rownumber parameter in a partition statement and deleted any row where the row number was greater than 1. (Remember the problem in the Chapter 5 SQL assignment?)

To see if the indexing help, click the  button and execute the query you wrote for #4, you will see something like the execution plan below.



You only need to submit the SQL that Creates the Index and Deletes the duplicate entries. The execution plan diagrams are only included to show the before and after difference the index makes.

7. You've decided that you want to see the ranking of each high temperatures for each city in each state to see if that helps you decide where to live. Write a query that ranks (using the rank function) the states by averages temperature and then ranks the cities in each state. The ranking of the cities should restart at 1 when the query returns a new state. You also want to only show results for the 15 states with the highest average temperatures.

Note: you will need to use multiple nested queries to get the State and City rankings, join them together and then apply a where clause to limit the state ranks shown.

State_Rank	State_Name	State_City_Rank	City_Name	Average Temp
1	Florida	1	Not in a City	73.975759
1	Florida	2	Pinellas Park	72.878784

1	Florida	3	Valrico	71.729440
1	Florida	4	Saint Marks	69.594272
2	Texas	1	McKinney	76.662423
2	Texas	2	Mission	74.701098

8. You notice in the results that sites with Not in a City as the City Name are include but do not provide you useful information. Exclude these sites from all future answers. You can do this by either adding it to the where clause in the remaining queries or updating the view you created in #4. Include the SQL for #7 with the revised answer

State_Rank	State_Name	State_City_Rank	City_Name	Average Temp
1	Florida	1	Pinellas Park	72.8787846986
1	Florida	2	Valrico	71.7294404954
1	Florida	3	Saint Marks	69.5942725766
2	Texas	1	McKinney	76.6624237831
2	Texas	2	Mission	74.7010981330

9. You've decided that the results in #8 provided too much information and you only want to 2 cities with the highest temperatures and group the results by state rank then city rank.

State_Rank	State_Name	State_City_Rank	City_Name	Average Temp
1	Florida	1	Pinellas Park	72.8787846986
1	Florida	2	Valrico	71.7294404954
2	Texas	1	McKinney	76.6624237831
2	Texas	2	Mission	74.7010981330
3	Mississippi	1	Jackson	67.5992336723
4	Louisiana	1	Baton Rouge	69.7044660969
4	Louisiana	2	Laplace (La Place)	68.1154000983

10. You decide you like the monthly average temperature to be at least 70 degrees. Write a query that returns the states and cities that meets this condition, the number of months where the average is above 70, the number of days in the database where the days are about 70 and calculate the average monthly temperature by month.

Hint, use the datepart function to identify the month for your calculations.

State_Name	City_Name	# of months	# of Day in Average	Average Temp
Arizona	Marana	12	14453	70.6322659585
Arizona	Mesa	6	184	77.2358813967
Arizona	Phoenix	12	5602	74.6881230972
Arizona	Queen Valley	12	1814	73.1228846135
Arizona	Scottsdale	6	368	76.5169080788

11. You assume that the temperatures follow a normal distribution and that the majority of the temperatures will fall within the 40% to 60% range of the cumulative distribution. Using the CUME_DIST function, show the temperatures for the cities having an average temperature of at least 70 degrees that fall within the range.

State_Name	City_Name	Avg_Temp	Temp_Cume_Dist
Arizona	Marana	66.0000000000	0.418044696602781
Arizona	Marana	67.0000000000	0.436241610738255
Arizona	Marana	68.0000000000	0.453815816785442
Arizona	Marana	69.0000000000	0.472151110496091
Arizona	Marana	70.0000000000	0.488479900366706

12. You decide this is helpful, but too much information. You decide to write a query that shows the first temperature and the last temperature that fall within the 40% and 60% range for cities you are focusing on in question #11.

State_Name	City_Name	40 Percentile Temp	60 Percentile Temp
Arizona	Marana	66.0000000000	76.0000000000
Arizona	Mesa	72.0000000000	86.0000000000
Arizona	Phoenix	69.0000000000	80.0000000000
Arizona	Queen Valley	69.0000000000	77.0000000000

13. You remember from your statistics classes that to get a smoother distribution of the temperatures and eliminate the small daily changes that you should use a moving average instead of the actual temperatures. Using the windowing within a ranking function to create a 4 day moving average, calculate the moving average for each day of the year.

Hint: You will need to datepart to get the day of the year for your moving average. Your moving average should use the 3 days prior and 1 day after for the moving average.

City_Name	Day of the Year	Rolling_Avg_Temp
Mission	1	59.022719
Mission	2	58.524868
Mission	3	58.812967
Mission	364	60.657749
Mission	365	61.726333
Mission	366	61.972514

Submission 2 - Geospatial Data, Stored Procedure and Optional Front End

2. Your last concern is how long will it take to travel back home to visit friends and family after you move. Since the Weather database has latitude and longitude information, you have decided to convert this information into a new column with a GEOGRAPHY data type and populate the new column with a set command and one of the following formula (Depending on the data type for latitude and longitude):

```
UPDATE [dbo].[AQS_Sites]
SET [GeoLocation] = geography::Point([Latitude], [Longitude], 4326)
```

Alternate versions that can work if the data was imported:

```
UPDATE [dbo].[AQS_Sites]
SET [GeoLocation] = geography::STPointFromText('POINT(' + CAST([LONGITUDE] AS
VARCHAR(20)) + ' ' + CAST([LATITUDE] AS VARCHAR(20)) + ')', 4326)
where [GEOCODE_LATITUDE] is not null
```

```
UPDATE [dbo].[aqs_sites]
SET [GeoLocation] = geography::STPointFromText('POINT(' + [LONGITUDE] + ' ' +
[Latitude] + ')', 4326)
where [LATITUDE] is not null
```

Once you've created and populated the new column, you write a stored procedure that will use has the latitude and longitude of where you are moving from (you will enter this data) and the state you are moving to as input parameters and returns all the cities in that state that the in the database and the distance between that city and where you are moving from. Provide the SQL to create the geospatial column and populate it as the first part of your answer. The second requirement for Part 4 is to create

the stored procedure and execute the stored procedure for from a spreadsheet (see question 2 below for information about the spreadsheet). The stored procedure has the following requirements:

- a. The name of the stored procedure must be UCID_Fall2022_Calc_GEO_Distance
- b. The stored procedure must have the following variables:
 - i. @longitude – This will contain the longitude of the starting location
 - ii. @latitude – this contains the latitude of the starting location
 - iii. @State – this contains the state name to get the data for
 - iv. @rownum – this contains the number of rows the stored procedure will return
- c. The logic in the stored procedure must do the following:
 - i. Select the site number, Local_Site_Name, Address, City_Name, State_Name, Zip_Code, Distance_In_Meters, Latitude, Longitude and Hours_of_Travel. If the Local_Site_Name is null, generate a value for the column by concatenating the Site_Number and City_Name
 - ii. Distance_In_Meters must be calculated using the following equation:
 $\text{geoLocation.STDistance}(@h)$ where @h is a geography variable calculated from the latitude and longitude of the starting location.
 - iii. Hours_of_Travel must be calculated using the following formula
 $(\text{geoLocation.STDistance}(@h) * 0.000621371) / 55$ (Assume you'll be traveling at the legal speed limit)
- d. The DDL that creates the stored procedure must:
 - i. check to see if the procedure exists and delete prior versions
 - ii. **include 2 exec statements for UCID_Fall2022_Calc_GEO_Distance at the end that runs the stored procedure with different variable values**

. You can get the latitude and longitude for any city by using the following link:

<http://www.findlatitudeandlongitude.com/>

To test your stored procedure, you need to use the EXEC statement. The format would be as follows:

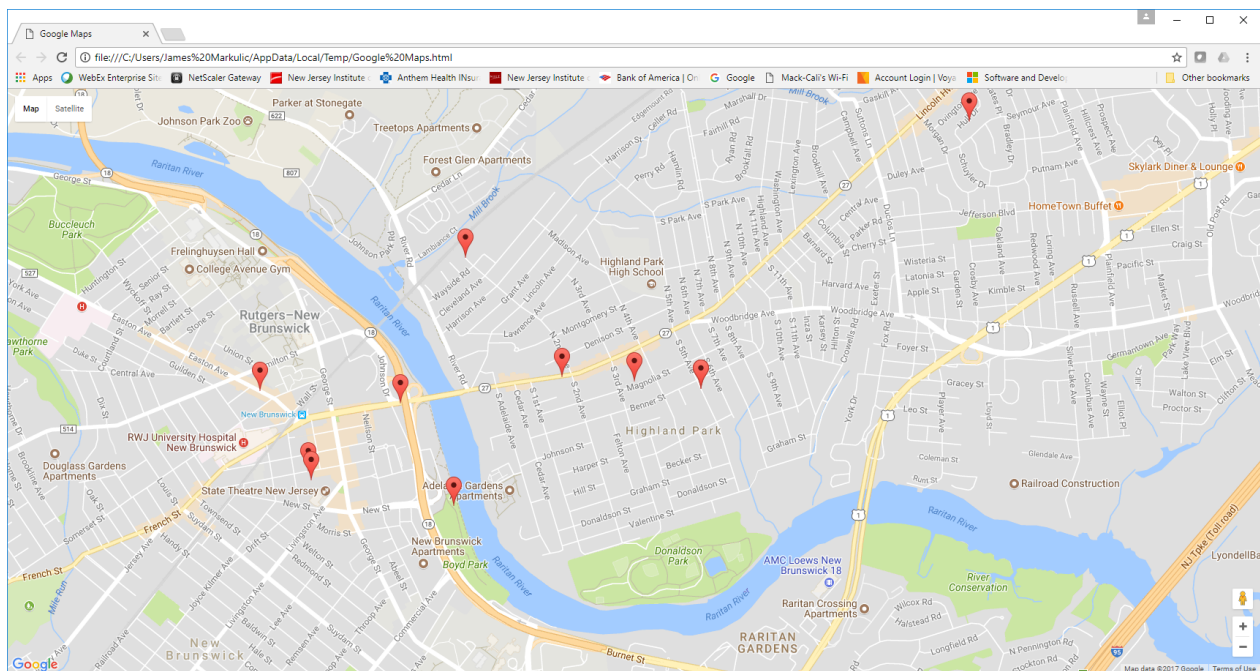
```
EXEC [Stored Procedure Name] - change the name to the one you used
@latitude = '36.778261',
@longitude = '-119.417932',
@State = 'California',
@rownum = 30
GO
```

3. **Optional:** Use the front end on the CANVAS site. The spreadsheets are on Moodle as [Class Project Front End for SQL Server 2019 File](#). The Front End/Spreadsheet allows you to enter the data needed for a stored procedure you will write and presents the output in a more readable format. The grey cells indicate where the VB program in the spreadsheet will get the variable values from. Rows 7 on (depending on the number entered for ROWS) contain the results from the stored procedure queries

NOTE: The spreadsheet requires specific installation parameters. If Office is installed differently on your PC, you will get messages like “Can’t find project or library”. This is very difficult to fix and you can skip this step.

1	Click Here To Run		To find latitude and longitude the following link	http://www.findlatitudeandlongitude.com/?loc=njit&id=2881477#.VcbQzz2RGI					
2									
3	Longitude	-74.426598	Rows	20					
4	Latitude	40.4991	Check for SQL Injection	No					
5	State	Arizona							
6									
7	Site Number	Local Site Name	ADDRESS	City Name	State Name	Zip Code	distance in meter	Latitude	Lon
8	0001	0001 Avondale	1201 S 4TH ST, AVONDALE	Avondale	Arizona	85323	3439075.822	33.420631	-112
9	0001	0001 Bisbee	CITY HALL 118 AZ. ST. BISBEE AZ	Bisbee	Arizona		3327799.342	31.410556	-109
10	0001	0001 Clifton	COURT HOUSE, CLIFTON	Clifton	Arizona	85540	3197819.344	33.046212	-109
11	0001	0001 Grand Canyon Village	VISITOR CENTER GRAND CANYON	Grand Canyon Village	Arizona	86046	3311398.947	36.051403	-112
12	0001	0001 Holbrook	ARIZ HWY DEPT YARD US 66-3 M	Holbrook	Arizona		3184779.769	34.939722	-110
13	0001	0001 Kingman	305 W BEALE ST, KINGMAN	Kingman	Arizona		3508281.3	35.187222	-114
14	0001	0001 Not in a City	US QUARANTINE STN, NOGALES	Not in a City	Arizona	85621	3415328.164	31.333473	-110
15	0001	0001 Prescott	500 S MARINA ST, PRESCOTT	Prescott	Arizona		3401354.146	34.534722	-112
16	0001	0001 Teec Nos Pos	TEEC NOS POS	Teec Nos Pos	Arizona		3022235.969	36.940833	-109
17	0001	0001 Yuma	COURT HOUSE-2ND AVE & 2ND S	Yuma	Arizona	85364	3664748.312	32.723968	-114
18	0001	AJO	AZ HWY DEPT YARD-WELL RD, AJ	Ajo	Arizona	85321	3531376.127	32.382036	-112
19	0001	CASA GRANDE DOWNTOWN	401 N MARSHALL ST, CASA GRAN	Casa Grande	Arizona	85222	3414105.361	32.877583	-112
20	0001	CLAYPOOL-AZ HWY DEPT YARD	AZ HWY DEPT YARD-HWY 60-70, I	Claypool	Arizona		3310463.239	33.446944	-110
21	0001	SAFFORD	523 10TH AVE, SAFFORD	Safford	Arizona	85546	3244038.36	32.83388	-109

The spreadsheet also has the functionality to generate the HTML to map the location of the responses similar to the example below. You can view the VB code in the spreadsheet to see how this was accomplished.



Submission 3 – GunCrime

1. In addition to the temperature, you are also very concerned with crime in the area. You've searched very hard and have been unable to find data that easily correlates to the temperature data. The best data you can find is the GunCrime.csv file in the Weather.zip file you downloaded in the beginning of the project.
2. Import the GunCrime.csv file into the database using the same method you used for the temperature and AQS_Sites data. Similar to step 11 in Part1, you need to make the following changes in the Advanced Tab:
 - a. The column date needs to be changed to have a datatype of date [DT_DATE]
 - b. For all columns except incident_id, date and state change the OutputColumnWidth to 5000
 - c. For Gun_type needs to change the OutputColumnWidth to 8000

This should import a total of 239,677 rows
3. Using the same methods as you used in Step 1 of Part 4, create and populate a geography column to the GunCrimes table you just created
 - a. Update the procedure you wrote for Part 2 of the project to also identify the number of crimes within 10 miles (16000 meters) of each site in the state you pass to the stored procedure by year that the crime was committed. Note: you will need consider that there are no columns to join the GunCrimes and AQS_Sites tables together, so you need to create a Cartesian Product and filter data out in the where statement using the following formula (which you'll need to modify) and the location data from your queries geoLocation.STDistance(@h)
4. Write a query that ranks all the cities in the state you selected from lowest to highest number of GunCrimes

Results	Messages	Site_Number	Local_Site_Name	ADDRESS	City_Name	State_Name	Zip_Code	distance in meters	distance in miles	Hours of Travel	Crime_Year	Shooting_Count
1		1001	Ancora State Hospital	Ancora State Hospital, 202 Spring Garden Road	Winslow (Township of)	New Jersey	8037	131375.193480659	81.6327353482707	1.48423155178674	2018	1
2		1006	Atlantic City	1535 Bacharach Boulevard	Atlantic City	New Jersey	8401	154838.838270125	96.212363774746	1.74931570499538	2014	2
3		0006	Bayonne	Veterans Park on Newark Bay, 25th Street near Pa...	Bayonne	New Jersey	7002	9152.71473380714	5.68723150686048	0.103404209215645	2014	4
4		0006	Bayonne	Veterans Park on Newark Bay, 25th Street near Pa...	Bayonne	New Jersey	7002	9152.71473380714	5.68723150686048	0.103404209215645	2015	3
5		0006	Bayonne	Veterans Park on Newark Bay, 25th Street near Pa...	Bayonne	New Jersey	7002	9152.71473380714	5.68723150686048	0.103404209215645	2016	7
6		0006	Bayonne	Veterans Park on Newark Bay, 25th Street near Pa...	Bayonne	New Jersey	7002	9152.71473380714	5.68723150686048	0.103404209215645	2017	8
7		0006	Bayonne	Veterans Park on Newark Bay, 25th Street near Pa...	Bayonne	New Jersey	7002	9152.71473380714	5.68723150686048	0.103404209215645	2018	2

5. Use your stored procedure to dense rank the cities by the number of shootings in descending order. Note you will need to use a row_num value that returns all the data for the state. See the [SQL Server select from stored procedure return table section of this site: SQL Server select from stored procedure \(9 Examples\) - SQL Server Guides](#) for instructions on how to use a stored procedure in a select query. The following used New Jersey as the state_name

state_name	city_name	shootings	gun_rank
New Jersey	Camden	70	1
New Jersey	Newark	66	2
New Jersey	Union City	64	3
New Jersey	Elizabeth	60	4
New Jersey	Jersey City	58	5
New Jersey	Fort Lee	50	6

Extra Credit Submission – Using XML and Geospatial Data Extra Credit

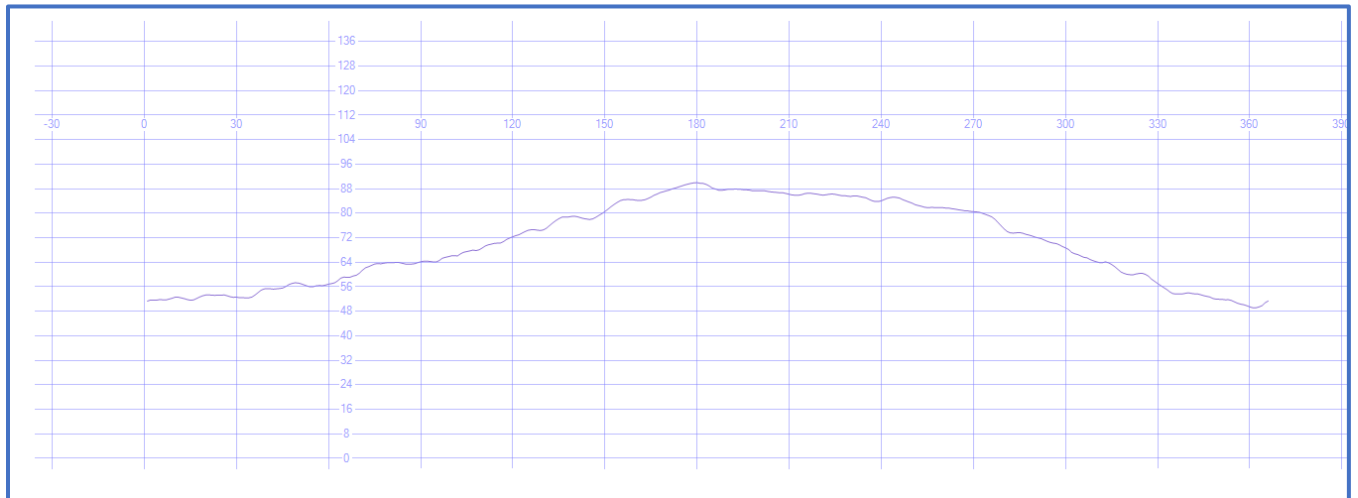
Note: Azure Data Studio does not currently support this functionality. ADS users cannot do this extra credit assignment.

1. You come across an article at <http://sqlmag.com/t-sql/generating-charts-and-drawings-sql-server-management-studio> that shows you how to graph data in Sql Server Management Studio. You decide to modify the following query provided in the article to graph your moving average for Tucson.

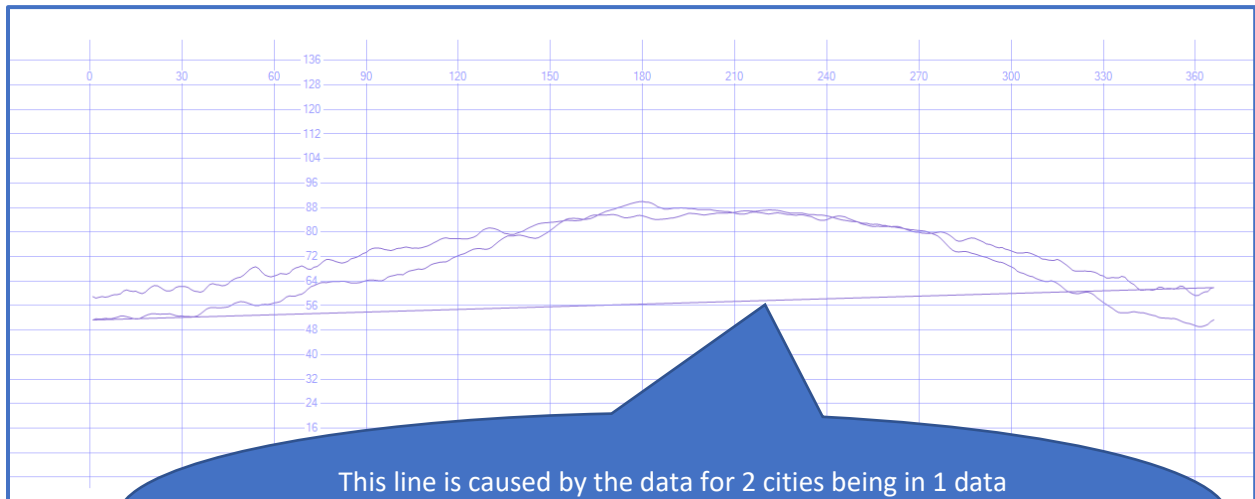
```
DECLARE @WKT AS VARCHAR(8000);
SET @WKT =
    STUFF (
        (SELECT ', ' + CAST( FY AS CHAR(4) ) + ' ' + CAST( Sales AS VARCHAR(30) )
        FROM #Sales
        ORDER BY FY
        FOR XML PATH('')), 1, 1, '');
SELECT geometry::STGeomFromText( 'LINESTRING(' + @WKT + ')', 0 );
```

You need to modify the part in red to house your query. You modify the select to cast the Day of the Year and Rolling_Average_Temp into varchar columns. You also have to modify the from to be a nested query that provides the 2 columns (Day of the Year and Rolling_Average_Temp) for Tucson.

Your query should provide a graph similar to the one below in the special results tab



2. You next want to add the data for Mission to the chart. To do this, you need to add a second part to the nested query that you replaced the FROM and ORDER BY statements with. You can use a UNION statement to join the data for the 2 cities.



This line is caused by the data for 2 cities being in 1 data string. I'll give extra credit to anyone who can figure out how to remove the line but show both cities on the same graph

3. You notice that if you change the sort order to the above data, you can get the graph to show the difference in the temperatures between the 2 cities. The resulting graphs should look like the following:

