Part 2 – 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.

Part 3 – 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.

1. Determine the date range of the records in the Temperature table

First Date	Last Date
1986-01-01	2017-05-09

2. Find the minimum, maximum and average temperature for each state

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

3. 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. Write the queries to find all suspect temperatures (below -39º and above 105º). Sort your output by State Name and Average Temperature.

State_Name	state_code	County_Code	Site_Number	average_Temp	date_local
Wisconsin	55	059	0002	-58.000000	2002-03-28
Washington	53	009	0013	-50.000000	2012-10-17
Texas	48	141	0050	106.041667	1991-07-28
Texas	48	141	0050	106.291667	1991-07-25

4. You noticed that the average temperatures become questionable below -39 º and above 125 º and that it is unreasonable to have temperatures over 105 º for state codes 30, 29, 37, 26, 18, 38. Write the queries that remove the questionable entries for these 3 set of circumstances.

5. Using the SQL RANK statement, rank the states by Average Temperature

State_Name	Minimum Temp	Maximum Temp	Average Temp	State_rank
Puerto Rico	73.462500	92.300000	81.690800	1
Virgin Islands	71.500000	84.541667	79.811958	2
Florida	35.958333	88.000000	73.347328	3

6. You decide that you are only interested in living in the United States, not Canada or the US territories. You will need to include SQL statements in all the remaining queries to limit the data returned in the remaining queries.

7. At this point, you've started to become annoyed at the amount of time each query is taking to run. You've heard that creating indexes can speed up queries. Create 5 indexes for your database. 2 of the indexes should index the temperature fields in the Temperature table, 1 index for the date in the Temperature table and 2 would index the columns used for joining the 2 tables (state, County and Site codes in the Temperate and aqs_site tables).

   To see if the indexing help, add print statements that write the start and stop time for the query in question #2 and run the query before and after the indexes are created. Note the differences in the times. Also make sure that the create index steps include a check to see if the index exists before trying to create it.

   The following is a sample of the output that should appear in the messages tab that you will need to calculate the difference in execution times before and after the indexes are created

```
Begin Question 6 before Index Create At - 13:40:03
(777 row(s) affected)
Complete Question 6 before Index Create At - 13:45:18
```

8. 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

9. 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.

10. 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	72.878784
1	Florida	2	Valrico	71.729440
2	Louisiana	1	Baton Rouge	69.704466
2	Louisiana	2	Laplace (La Place)	68.115400

11. You decide you like the average temperature to be in the 80's so you decide to research Pinellas Park, Mission, and Tucson in more detail. For Ludlow, California, calculate the average temperature by month. You also decide to include a count of the number of records for each of the cities to make sure your comparisons are being made with comparable data for each city.

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

City_Name	Month	# of Records	Average Temp
Mission	1	620	60.794048
Mission	2	565	64.403861
Mission	3	588	69.727512

12. 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 same 3 cities that fall within the range.

City_Name	Avg_Temp	Temp_Cume_Dist
Mission	73.916667	0.400686891814539
Mission	73.956522	0.400829994275902
Mission	73.958333	0.402404121350887

13. 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 the 3 cities your focusing on.

City_Name	40 Percentile Temp	60 Percentile Temp
Mission	73.956522	80.083333
Pinellas Park	71.958333	78.125000
Tucson	63.750000	74.250000

14. You decide you want more detail regarding the temperature ranges and you think of using the NTILE function to group the temperatures into 10 groups. You write a query that shows the minimum and maximum temperature in each of the ntiles by city for the 3 cities you are focusing on.

City_Name	Percentile	Min Temp	Max Temp
Mission	1	29.125000	57.916667
Mission	2	57.916667	65.333333
Mission	3	65.333333	70.125000

15. You now want to see the percent of the time that will be at a given average temperature. To make the percentages meaningful, you only want to use the whole number portion of the average temperature. You write a query that uses the percent_rank function to create a table of each temperature for each of the 3 cities sorted by percent_rank. The percent_rank needs to be formatted as a percentage with 2 decimal places.

City_Name	Avg_Temp	Percentage
Mission	29	0.00%
Mission	33	1.69%
Mission	34	3.39%

16. 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. You 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

Part 4 Front End and 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):

```
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 Spring2018_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 lataidute 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
    ii. Distance_In_Meters must be calculated using the following equation: 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 (geoLocation.STDistance(@h))/80000
    iv. Many of the rows in the AQS_Sites table do not contain valid Local_Site_Names. When that is the case, use a CASE statement to create a Local_Site_Name by combining the Site_Number and City_Name. The CASE statement must chech for both null values and for when the Local_Site_Name = ‘’
d. The DDL that creates the stored procedure must:
    i. check to see if the procedure exists and delete prior versions
    ii. have 2 exec Spring2018_Calc_GEO_Distance statements at the end that runs the stored procedure with different variable values

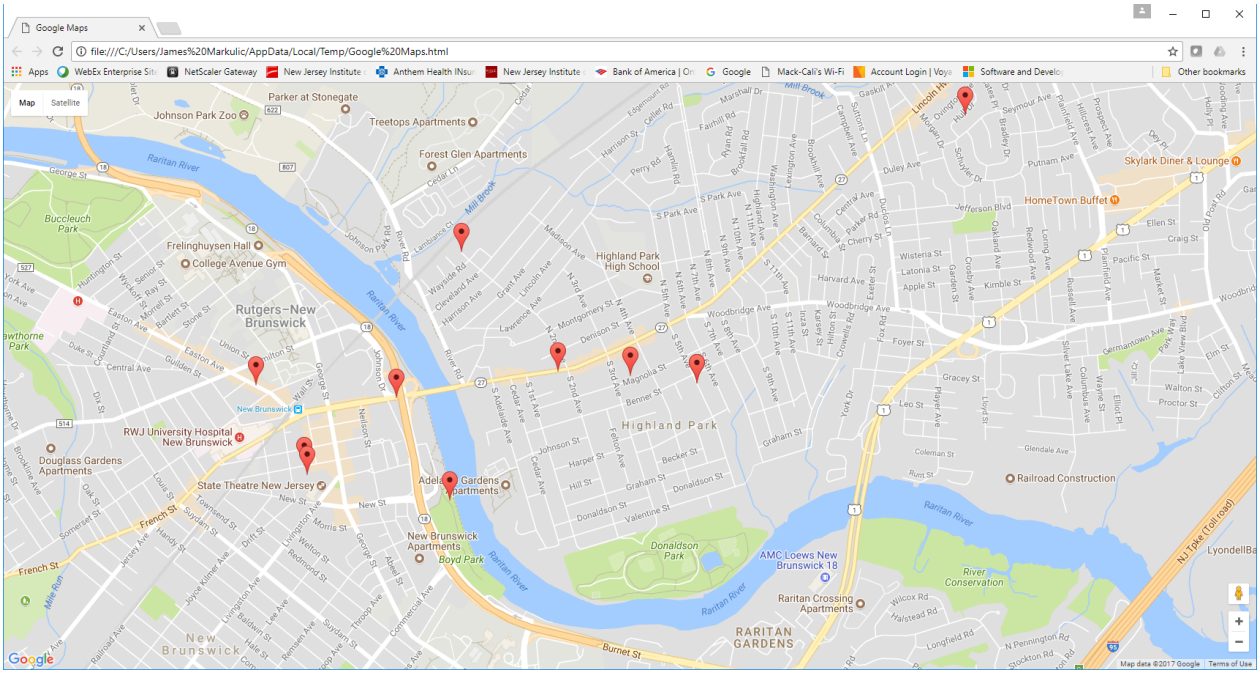
. You can get the latitude and longitude for any city by using the following link:

http://www.findlatitudeandlongitude.com/

2. Use the front end on the Moodle site. The spreadsheets are on Moodle as Class Project Front End for SQL Server XXXX File where XXX is the version of SQL Server you’re using. 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

1	Click Here To Run		To find latitude and longitudethe following link		http://www.findlatitudeandlongitude.com/?loc=njit&id=2881477#.VcbQzzZRGU					
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 mete	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 8	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 CANYOI	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	-111	
20	0001	CLAYPOOL-AZ HWY DEPT YARD	AZ HWY DEPT YARD-HWY 60-70, (	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.



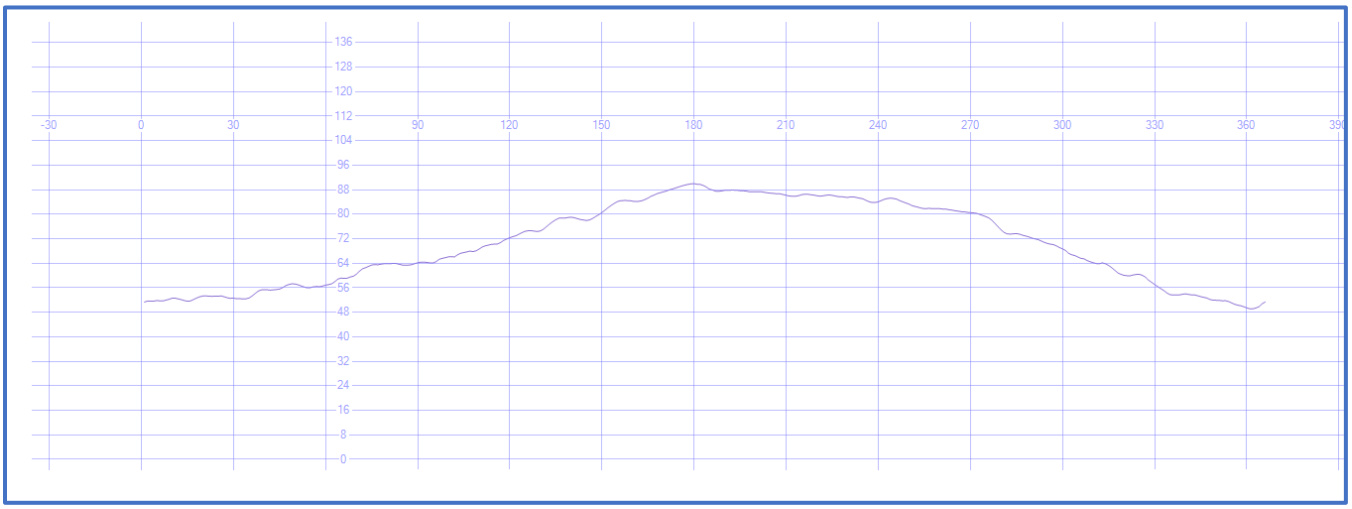
Part 5 – Using XML and Geospacial Data – Extra Credit

3. 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.

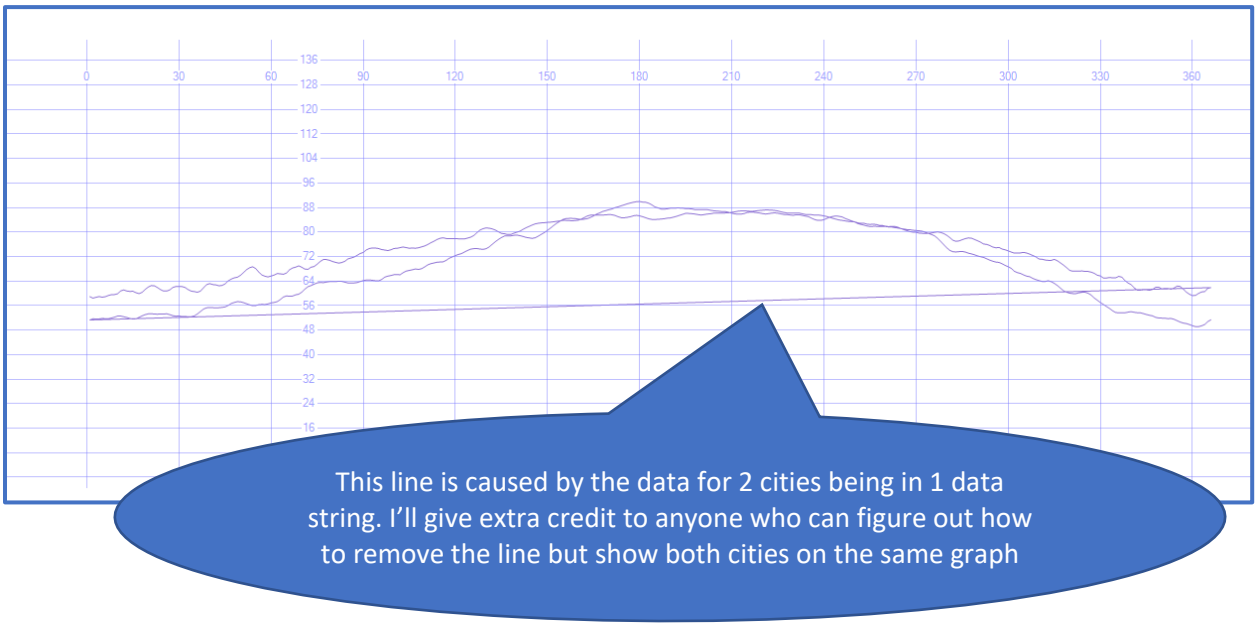
```
DECLARE @WKT AS VARCHAR(8000);
SET @WKT =
  STUFF(
     (SELECT ',' + CAST( FY AS CHAR(4) ) + ' ' + CAST( Sales AS VARCHAR(30) )
     FROM #Sales
     ORDER BY FY
     FOR XML PATH('')), 1, 1, '');
SELECT geometry::STGeomFromText( 'LINESTRING(' + @WKT + ')', 0 );
```

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



4. 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.



5. 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: