CS236
Database Management System
Final Project Report

Sakib Md Bin Malek
862004540

The goal of the project was to leverage spark's parallel ability to divide up the SQL queries to consolidate the precipitation data of the 50 states over the real data collected for about 4 years from various weather stations.

Script to run the program

I used python as the programming language to complete the project. I also used pyspark package and used hadoop (version 2.4.5)

```
#Setting up environment
$ pip3 install pyspark
$ wget https://downloads.apache.org/spark/spark-2.4.5/
spark-2.4.5-bin-hadoop2.7.tgz
$ tar xvf spark-2.4.5-bin-hadoop2.7.tgz

#After properly setting up the spark and hadoop
#Both input and output folder path are optional.
$ spark-submit project.py -o output_folder_path -i input_folder_path
$ less <output_folder_path>/output.txt
```

Project Approach

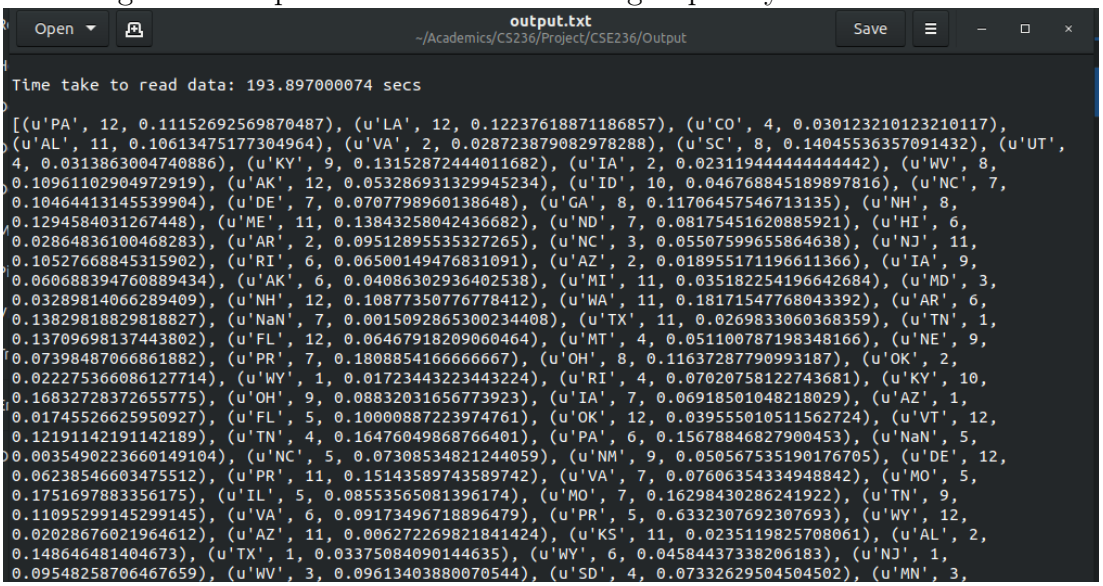
I used the following steps to complete the project.

- Read weather station metadata
- Read weather station data
- Remove invalid row and remove redundant columns

- Run queries to get aggregated data grouped by state and month
- Create a new table using the aggregated data in the earlier step
- Run queries on new table to get Max, Min and Difference data

Results

Following is the snapshot of the consolidated grouped by State and Month.



Following is the states sorted by the difference in maximum and minimum precipitation based on month in ascending order.

```
Row(STATE=u'DC', DIFF_PRCP=0.0, MAX_PRCP=0.0, MIN_PRCP=0.0)
Row(STATE=u'NaN', DIFF_PRCP=0.002393434166724912, MAX_PRCP=0.0035490223660149104,
MIN_PRCP=0.001155588199289985)
Row(STATE=u'NV', DIFF_PRCP=0.016430704822198952, MAX_PRCP=0.025384193194291978, MIN_PRCP=0.008953488372093026)
Row(STATE=u'RI', DIFF_PRCP=0.02807150010215058, MAX_PRCP=0.07020758122743681, MIN_PRCP=0.04213608112528623)
Row(STATE=u'UT', DIFF_PRCP=0.028588202826524446, MAX_PRCP=0.05096310193871169, MIN_PRCP=0.022374899112187242)
Row(STATE=u'MI', DIFF_PRCP=0.03955909884907908, MAX_PRCP=0.07095705521472392, MIN_PRCP=0.031397956365644845)
Row(STATE=u'ID', DIFF_PRCP=0.045042798976555776, MAX_PRCP=0.05972575115950796, MIN_PRCP=0.014682952182952184)
Row(STATE=u'IL', DIFF_PRCP=0.05178858515945441, MAX_PRCP=0.08596879788271349, MIN_PRCP=0.03418021272325907)
Row(STATE=u'WY', DIFF_PRCP=0.05182458498220462, MAX_PRCP=0.06414096916299558, MIN_PRCP=0.012316384818079096)
Row(STATE=u'MN', DIFF_PRCP=0.0537049132599366, MAX_PRCP=0.05808902657753138, MIN_PRCP=0.004384113317594779)
Row(STATE=u'AK', DIFF_PRCP=0.05499419962607067, MAX_PRCP=0.08550112826753889, MIN_PRCP=0.03050692864146822)
Row(STATE=u'MD', DIFF_PRCP=0.0550405872492519, MAX_PRCP=0.08597362296353761, MIN_PRCP=0.03093303571428571)
Row(STATE=u'WI', DIFF_PRCP=0.056384101983851034, MAX_PRCP=0.07517805839513596, MIN_PRCP=0.018793956411284927)
Row(STATE=u'IN', DIFF_PRCP=0.05903946510110894, MAX_PRCP=0.12301587301587301, MIN_PRCP=0.06397640791476407)
Row(STATE=u'NC', DIFF_PRCP=0.06100515622699178, MAX_PRCP=0.10464413145539904, MIN_PRCP=0.04363897522840726)
Row(STATE=u'CO', DIFF_PRCP=0.06136640976688339, MAX_PRCP=0.07250497434286311, MIN_PRCP=0.011138564575979724)
Row(STATE=u'AZ', DIFF_PRCP=0.062294371064985074, MAX_PRCP=0.0685666408868265, MIN_PRCP=0.006272269821841424)
```

Runtime

Total runtime of my program is **212.44 sec**. However, it only took **18.54 sec** to run both of the queries and create intermediate table. Most of the time (**193.90 sec**) is spent reading the data to the memory and clean the data.

Comment

There are number of issues that I faced while completing the project.

One of the issues that I encountered was that my program was very slow. I was looping though all the state and making queries to the SQL with a state filter. This made my project extremely slow, taking more than 10 mins. This issue was easily resolved by adding a **Group By** in the SQL query. That brought down the runtime to less than 4 mins, about **3X faster**.

Another issue was due to the configuration of the hadoop. I was getting **OutOfMemoryError Exception in Java** from hadoop. This was due to the fact that default configuration does not allocate enough memory for spark units. I used Python's garbage collector to removed already used table from the memory. That did not resolve the issue, thus I had to make changes to the config file. It solved the problem.