# CSEE5590/490: Big Data Programming

# Project Proposal (Increment 2)

## Due Date: Friday, Mar 26, 2021

**Project Title:** Analysis for determination of a relationship between energy demand and weather.

**Team Members:** Joe Goldsich, Anna Johnson, Kyle Son, and Bill Yerkes

**Introduction:** Information overload can make it difficult to understand issues and solve problems. The appearance of excessive quantity of information prevents the understanding of the problem and the ability to construct a solution to the problem. Big Data technologies takes this paradigm and turns it inside out. The students working on this project are going to attempt to construct a model which will be able to take weather and energy related data to be able to forecast energy demand.

**Goals and Objectives:** Utilize the tools and technologies learned from CSEE 5590 to be able to analyze collected data so that it will be possible to determine if there is a relationship between weather and energy consumption and if a relationship exists determine the possibilities of using that relationship to predict future energy needs.

**Motivation:** The global population continues to increase, and the weather patterns seem to be getting more extreme, from extending periods of both above and below normal temperatures in various parts of the world and in the United States. The demand and consumption of energy increase with the population and with the extreme weather, the need for air conditioning in the summer and for heating in the winter. The recent crisis in Texas has demonstrated what can happen if the energy providers are not able to meet the demands of the consumers. Being able to forecast accurately future demand and plan accordingly can help prevent or mitigate such crises in the future.

**Significance:** Better planning of resources for Utility Companies can result in reduced cost to the consumers and more reliable service. This also dips into the area of public safety, as loss of power during extreme weather with no warning can be dangerous for vulnerable groups.

**Objectives:**

Visualize the load and marginal supply curves.

What weather measurements and cities influence most the electrical demand, prices, generation capacity?

Can we forecast 24 hours in advance better than the TSO?

Can we predict electrical prices by time of day better than TSO?

Forecast intraday price or electrical demand hour-by-hour.

Potentially identify the abilities of different energy sources/ systems of regulation to keep up with fluctuating demand.

What is the next generation source to be activated on the load curve?

**Features:**

Hive (Map Reduce):

Use Hive and Map Reduce to store and gather metrics on the data to be able to feed the Spark, GraphX and MLLib portions of the project.

Sqoop (MySQL):

Sqoop and MySQL can be used to query our assembled database to identify patterns between extreme weather and fluctuations in energy consumption. MySQL can also be used to identify notable power outages in the past, or times when utility companies were able to keep up with rising demands.

Solr & Lucene (Search Engines):

We have not covered these technologies at this point in time.  We need to determine how search engines can be applied to the project.

Cassandra (No SQL):

We will determine if we need to store data in a NoSQL repository, versus storing the data in Hive and MySQL, to be able to perform the required work to make a predictive model.  (We have not covered Cassandra as of yet, do not have enough information as to how it can benefit in solving the problem.)

Spark / GraphX / MLLib (Programming and Analytics):

Use Spark, GraphX, and MLLib to analyze the weather and electrical data to be able to create a model which can predict the demand/cost of electricity based upon supplied weather information

**Approaches/Methods:**

The team's initial form of communication was via Email. The discussion over email resulted in the team deciding to communicate via Discord, which has allowed for the team member to communicate on a regular basis. The team also meets periodically via Zoom to review items and discuss plans.

The team uses Google Docs and Google Slides to collaborate on documentation. The team also uses GitHub as the repository for source code and final version of documentation.

The team met to discuss several ideas on what topic to do the project on and came to a consensus to do the project on Energy prediction based on Weather. The team reviewed the information on the Kaggle site. The team decided to leverage the knowledge we had gained over the first half of the course to investigate both the technologies, Hadoop, MapReduce, Hive, Sqoop, Cassandra, Solr, that were covered and the data, Weather and Energy datasets.

Each team member worked on their own. They utilized the various tools to do analysis on the data for the project. This allowed the team members to get more practice with the various tools and to get familiar with the data. This approach led to discussions of common issues found and exchanges of ideas between team members. The team shared with each other their findings on various tools and data. The team also started discussion on what approach the team should take for the next iteration.

**Story Telling:**

**Life**

1. **Who** are the people or communities in need of help?

   Consumers of energy utility companies (gas and electricity) are one portion of people who need help, as the recent crisis in Texas has demonstrated. The other people who need help are the producer of the energy being consumed (Utility Companies) and their suppliers. Being able to accurately forecast demand can help them better plan on how much to produce, how much raw materials to keep in stock, and how to better plan to use their resources, including labor, to meet their client's demands. In the case where it is not possible for utility companies to keep up with demand, adequate warning could be given to energy consumers so they can plan for loss of power.

2. **What** problem happened to them?

Recently in Texas with the severe cold weather, the demand for electricity far exceeded the capacity the Utility Companies were able to provide. In addition, the cold weather caused some producers of gas to halt production. If the Utility Companies could have foreseen the spike in demand, it may have been possible for them to take actions to mitigate the negative impacts which were caused by the shortages in energy as compared to the demand that was required to keep people warm.

3. **When** did the problem take place?

This recent issue occurred in February 2021. There were also rolling blackouts in Kansas City Missouri in February 2021 because of the cold. There have been rolling blackouts in California because of the heat over numerous years.

4. **Where** means two things:

    a. The environment and settings that the people or the community is living in, and

Due to a rapidly changing climate, these problems can affect a wide range of environments. Communities that are not accustomed to extreme cold or hot weather are particularly vulnerable to energy shortfalls when hit with unexpected demand. Notably in the case of Texas, homes there were designed to shed heat, so when they were hit with uncharacteristically cold weather, the infrastructure was especially unequipped to deal with power outages.

    b. The place/location where the problem takes place.

The problem of energy shortages can occur anywhere, in any country. The recent cases have been in Kansas City Missouri and in Texas. They have occurred in California. They can occur anywhere.

5. **Why** means the possible causes and/origin of the problem.

The inability of the Utility Companies to accurately predict demand and a lack of excess capacity to handle a reduction in production capacity by outside forces results in a failure to meet the demands of the consumer. IT companies have HA and redundancy built into their server farms to help prevent outages of their services. Utility companies need to have the same HA and redundancy built into their systems, and they need to understand and be able to reasonably predict how much demand there will be from their consumers.

6. **How**: If you would like, you can add a dimension of how. How did it happen? Sometimes, the answer to how can be covered by what, when, and where.

In the case of Texas, lawmakers were warned years in advance of the possibility of this situation happening. They ignored recommendations that they winterize their energy infrastructure, and that contributed greatly to the energy shortages as some systems failed. Analysis of energy consumption vs. weather patterns could provide the necessary information to utility companies and lawmakers in advance of disastrous situations and could help hold them accountable for their lack of preparedness.

**Workflow:**

Gain knowledge and experience with Big Data Programming tools.

Select the problem the team wishes to solve.

Research aspects of the problem, data, tools, why, benefits, issues.

Select the tools the team wishes to use to solve the problem.

[Repeat following steps as necessary]

Refine the data associated with the problem.

Design Model for solving the problem.

Construct Model for solving the problem.

Test/validate Model for solving the problem.

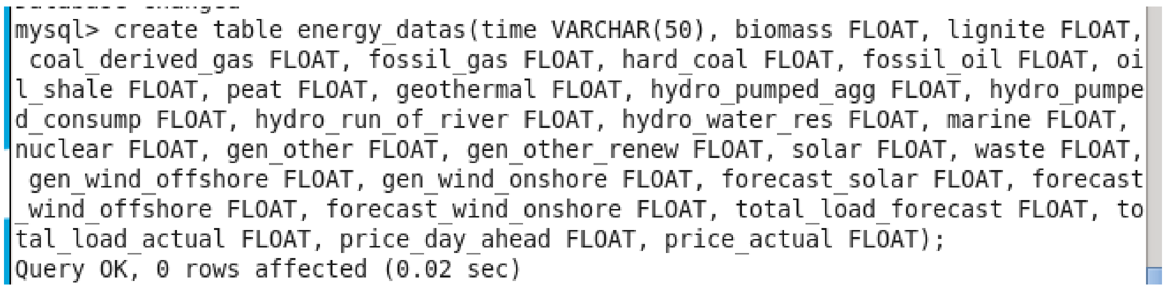
Create a presentation of the solution the team has created.

**Working screens from project:**

**Anna Johnson:**

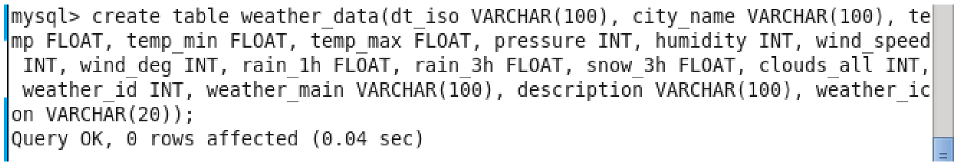
Create tables in MySQL:

Energy dataset:



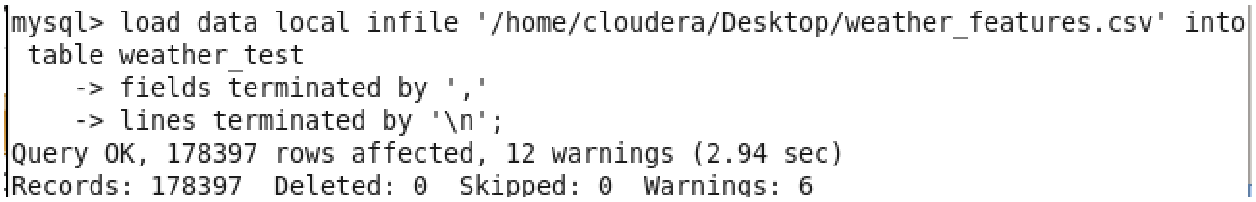
```
mysql> create table energy_datas(time VARCHAR(50), biomass FLOAT, lignite FLOAT,
 coal_derived_gas FLOAT, fossil_gas FLOAT, hard_coal FLOAT, fossil_oil FLOAT, oi
l_shale FLOAT, peat FLOAT, geothermal FLOAT, hydro_pumped_agg FLOAT, hydro_pumpe
d_consump FLOAT, hydro_run_of_river FLOAT, hydro_water_res FLOAT, marine FLOAT,
nuclear FLOAT, gen_other FLOAT, gen_other_renew FLOAT, solar FLOAT, waste FLOAT,
 gen_wind_offshore FLOAT, gen_wind_onshore FLOAT, forecast_solar FLOAT, forecast
_wind_offshore FLOAT, forecast_wind_onshore FLOAT, total_load_forecast FLOAT, to
tal_load_actual FLOAT, price_day_ahead FLOAT, price_actual FLOAT);
Query OK, 0 rows affected (0.02 sec)
```

Weather dataset:



```
mysql> create table weather_data(dt_iso VARCHAR(100), city_name VARCHAR(100), te
mp FLOAT, temp_min FLOAT, temp_max FLOAT, pressure INT, humidity INT, wind_speed
 INT, wind_deg INT, rain_1h FLOAT, rain_3h FLOAT, snow_3h FLOAT, clouds_all INT,
 weather_id INT, weather_main VARCHAR(100), description VARCHAR(100), weather_ic
on VARCHAR(20));
Query OK, 0 rows affected (0.04 sec)
```

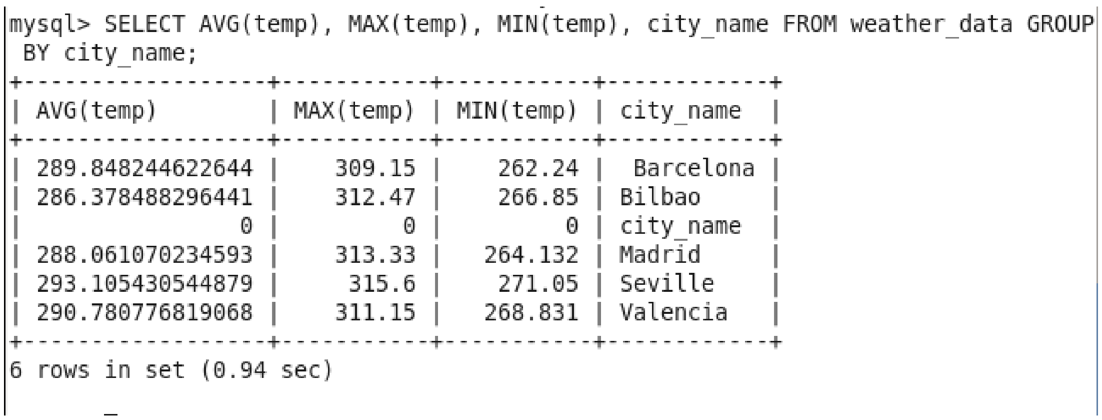
Load Data into MySQL tables from csv files:



```
mysql> load data local infile '/home/cloudera/Desktop/weather_features.csv' into
 table weather_test
    -> fields terminated by ','
    -> lines terminated by '\n';
Query OK, 178397 rows affected, 12 warnings (2.94 sec)
Records: 178397  Deleted: 0  Skipped: 0  Warnings: 6
```

Run Queries on data in MySQL:

Query on weather dataset to get statistics on weather for each city:

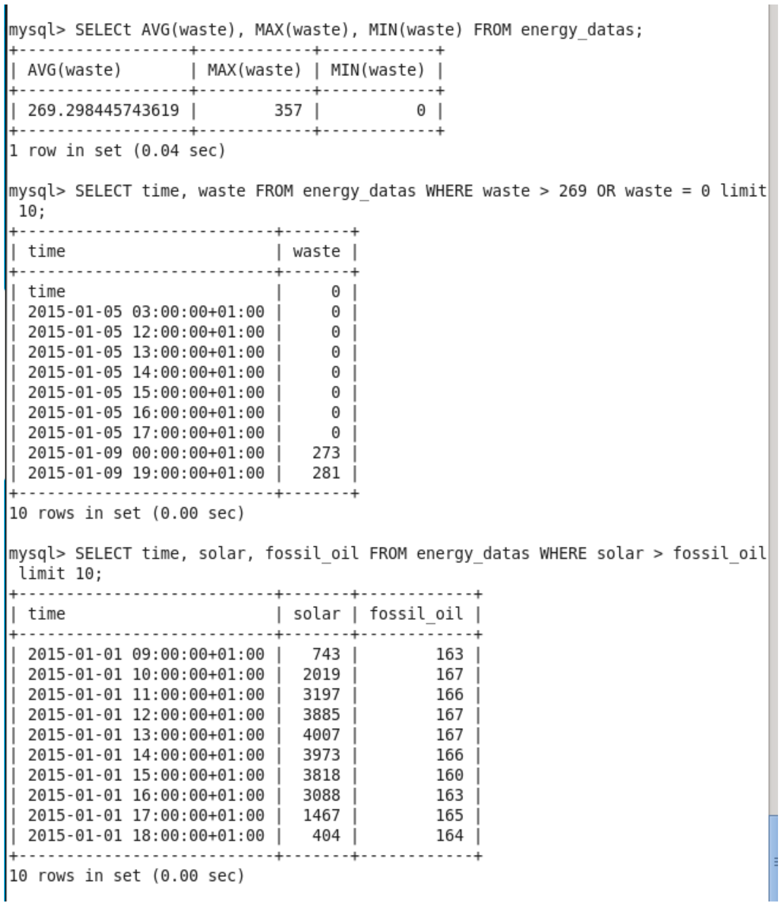


```
mysql> SELECT AVG(temp), MAX(temp), MIN(temp), city_name FROM weather_data GROUP
 BY city_name;
+------------------+-----------+-----------+------------+
| AVG(temp)        | MAX(temp) | MIN(temp) | city_name  |
+------------------+-----------+-----------+------------+
| 289.848244622644 |    309.15 |    262.24 |  Barcelona |
| 286.378488296441 |    312.47 |    266.85 | Bilbao     |
|                0 |         0 |         0 | city_name  |
| 288.061070234593 |    313.33 |   264.132 | Madrid     |
| 293.105430544879 |     315.6 |    271.05 | Seville    |
| 290.780776819068 |    311.15 |   268.831 | Valencia   |
+------------------+-----------+-----------+------------+
6 rows in set (0.94 sec)
```

Queries on energy dataset:

1. Find the average, min, and max for the ‘waste’ column
2. Find times where there was above average waste, or no waste at all

3. Find times where more energy was generated from solar sources than from fossil oil sources

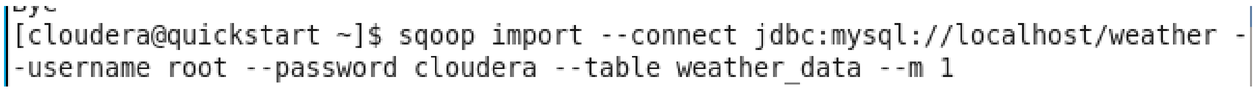


```
mysql> SELECt AVG(waste), MAX(waste), MIN(waste) FROM energy_datas;
+------------------+------------+------------+
| AVG(waste)       | MAX(waste) | MIN(waste) |
+------------------+------------+------------+
| 269.298445743619 |        357 |          0 |
+------------------+------------+------------+
1 row in set (0.04 sec)

mysql> SELECT time, waste FROM energy_datas WHERE waste > 269 OR waste = 0 limit
 10;
+---------------------------+-------+
| time                      | waste |
+---------------------------+-------+
| time                      |     0 |
| 2015-01-05 03:00:00+01:00 |     0 |
| 2015-01-05 12:00:00+01:00 |     0 |
| 2015-01-05 13:00:00+01:00 |     0 |
| 2015-01-05 14:00:00+01:00 |     0 |
| 2015-01-05 15:00:00+01:00 |     0 |
| 2015-01-05 16:00:00+01:00 |     0 |
| 2015-01-05 17:00:00+01:00 |     0 |
| 2015-01-09 00:00:00+01:00 |   273 |
| 2015-01-09 19:00:00+01:00 |   281 |
+---------------------------+-------+
10 rows in set (0.00 sec)

mysql> SELECT time, solar, fossil_oil FROM energy_datas WHERE solar > fossil_oil
 limit 10;
+---------------------------+-------+------------+
| time                      | solar | fossil_oil |
+---------------------------+-------+------------+
| 2015-01-01 09:00:00+01:00 |   743 |        163 |
| 2015-01-01 10:00:00+01:00 |  2019 |        167 |
| 2015-01-01 11:00:00+01:00 |  3197 |        166 |
| 2015-01-01 12:00:00+01:00 |  3885 |        167 |
| 2015-01-01 13:00:00+01:00 |  4007 |        167 |
| 2015-01-01 14:00:00+01:00 |  3973 |        166 |
| 2015-01-01 15:00:00+01:00 |  3818 |        160 |
| 2015-01-01 16:00:00+01:00 |  3088 |        163 |
| 2015-01-01 17:00:00+01:00 |  1467 |        165 |
| 2015-01-01 18:00:00+01:00 |   404 |        164 |
+---------------------------+-------+------------+
10 rows in set (0.00 sec)
```

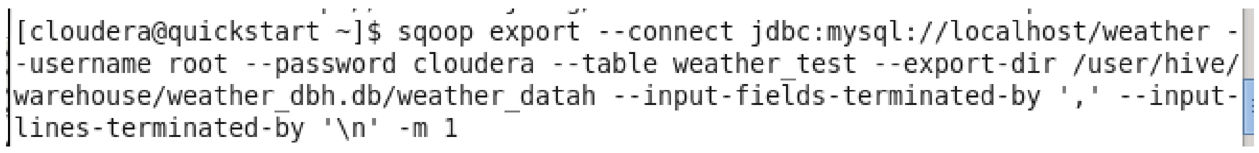
These queries helped us to gain a better insight into our datasets. We can apply this knowledge in the next steps of our project to identify important features to compare between the energy and weather datasets to determine the relationship between them.

Lastly, it's important that we are able to move these datasets between Hadoop and MySQL so that we can utilize the different functionality of both the Hive and MySQL databases. This can be done using Sqoop.

Exporting and importing tables between Hadoop, Hive, and MySQL:

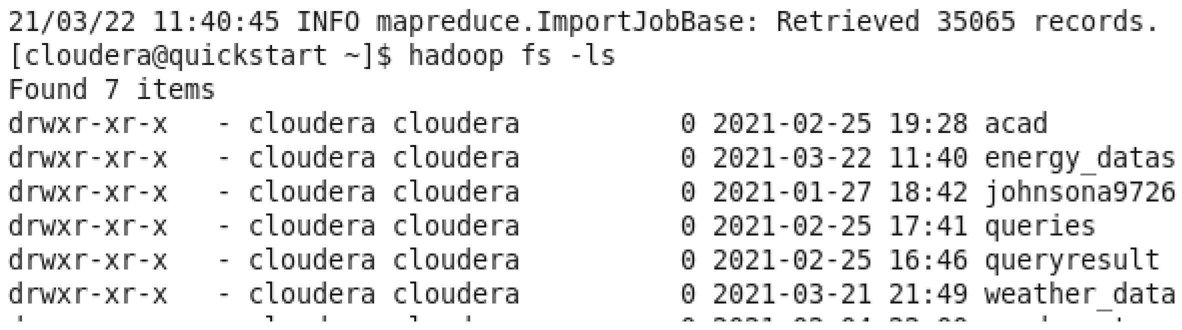


```
[cloudera@quickstart ~]$ sqoop import --connect jdbc:mysql://localhost/weather -
-username root --password cloudera --table weather_data --m 1
```



```
[cloudera@quickstart ~]$ sqoop export --connect jdbc:mysql://localhost/weather -
-username root --password cloudera --table weather_test --export-dir /user/hive/
warehouse/weather_dbh.db/weather_datah --input-fields-terminated-by ',' --input-
lines-terminated-by '\n' -m 1
```

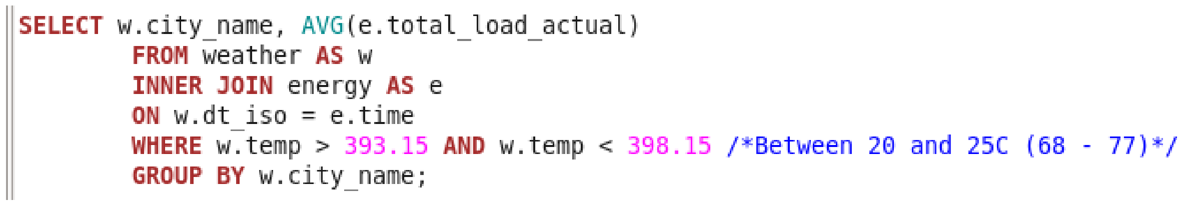
The MySQL tables in HDFS after importing using sqoop:



```
21/03/22 11:40:45 INFO mapreduce.ImportJobBase: Retrieved 35065 records.
[cloudera@quickstart ~]$ hadoop fs -ls
Found 7 items
drwxr-xr-x   - cloudera cloudera          0 2021-02-25 19:28 acad
drwxr-xr-x   - cloudera cloudera          0 2021-03-22 11:40 energy_datas
drwxr-xr-x   - cloudera cloudera          0 2021-01-27 18:42 johnsona9726
drwxr-xr-x   - cloudera cloudera          0 2021-02-25 17:41 queries
drwxr-xr-x   - cloudera cloudera          0 2021-02-25 16:46 queryresult
drwxr-xr-x   - cloudera cloudera          0 2021-03-21 21:49 weather_data
```

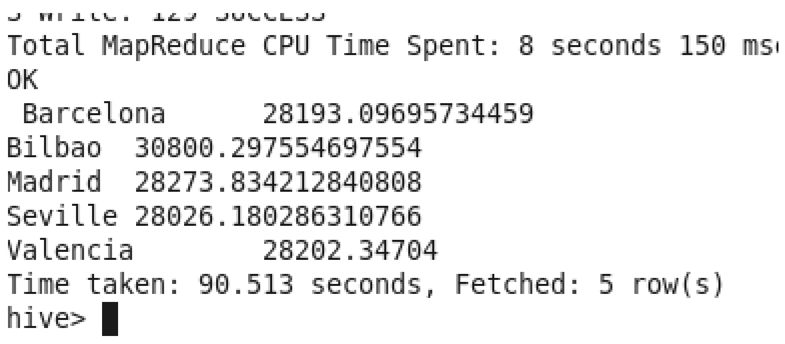
**Joe Goldsich:**

An example of a simple query on a join of the two data sets using HiveQL:



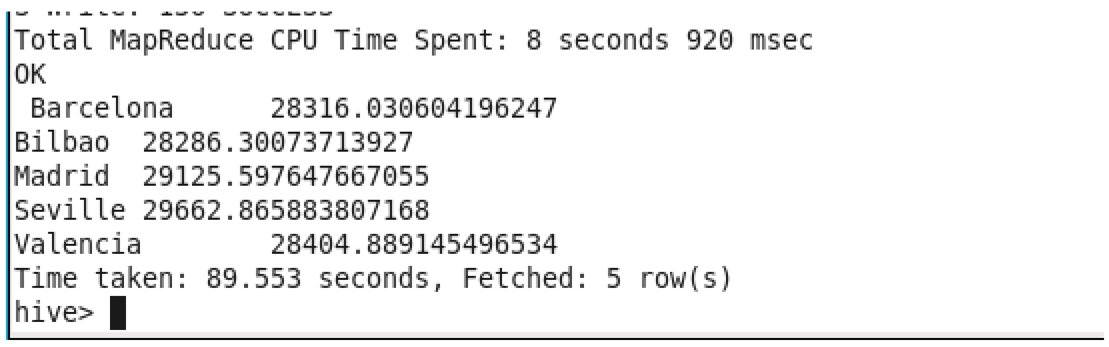
```sql
SELECT w.city_name, AVG(e.total_load_actual)
        FROM weather AS w
        INNER JOIN energy AS e
        ON w.dt_iso = e.time
        WHERE w.temp > 393.15 AND w.temp < 398.15 /*Between 20 and 25C (68 - 77)*/
        GROUP BY w.city_name;
```

And the results of this simple query:



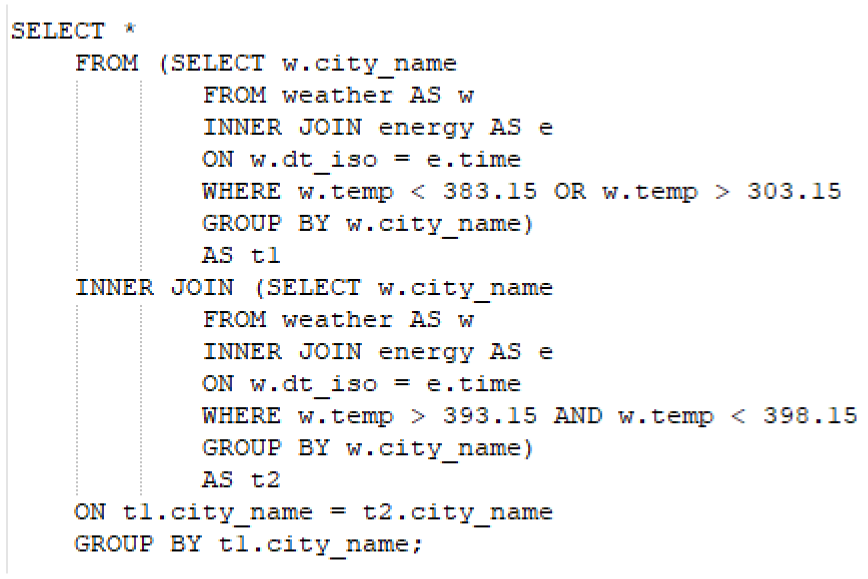
```
Total MapReduce CPU Time Spent: 8 seconds 150 ms
OK
 Barcelona      28193.09695734459
Bilbao  30800.297554697554
Madrid  28273.834212840808
Seville 28026.180286310766
Valencia        28202.34704
Time taken: 90.513 seconds, Fetched: 5 row(s)
hive>
```

With the exception of Bilbao (which has a very temperate climate) most of the cities experienced an increase in their energy usage at more extreme temperatures (below 10C and over 30C):



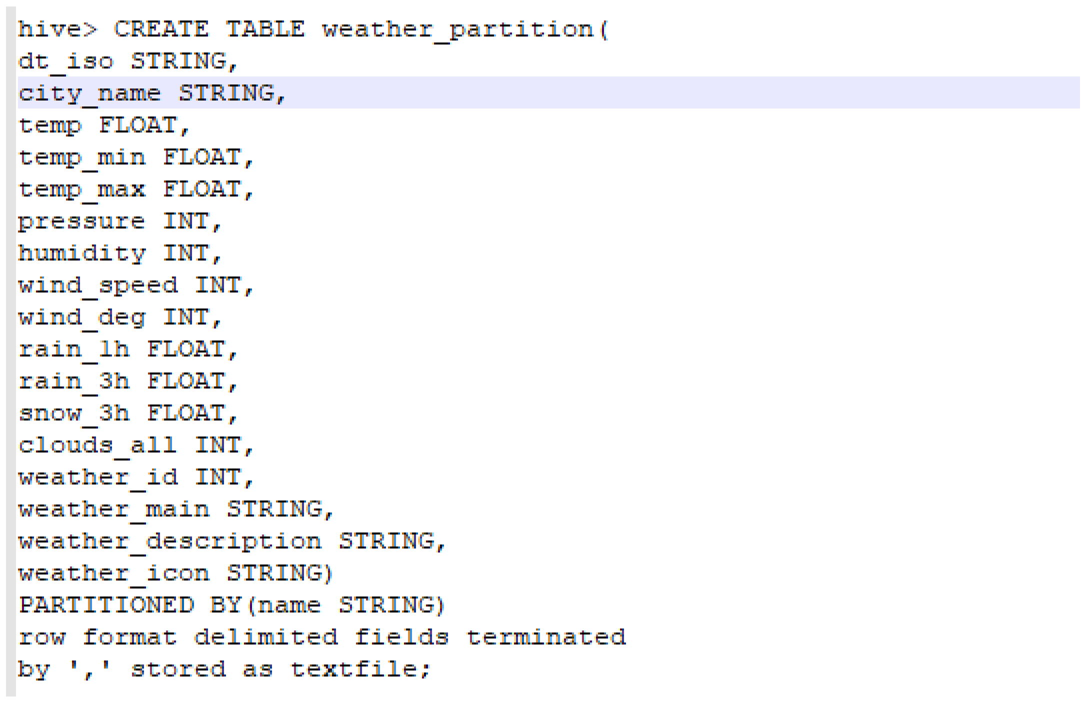
```
Total MapReduce CPU Time Spent: 8 seconds 920 msec
OK
 Barcelona      28316.030604196247
Bilbao  28286.30073713927
Madrid  29125.597647667055
Seville 29662.865883807168
Valencia        28404.889145496534
Time taken: 89.553 seconds, Fetched: 5 row(s)
hive>
```

A slightly more complicated Select query was very slow to run (10 minutes slow).

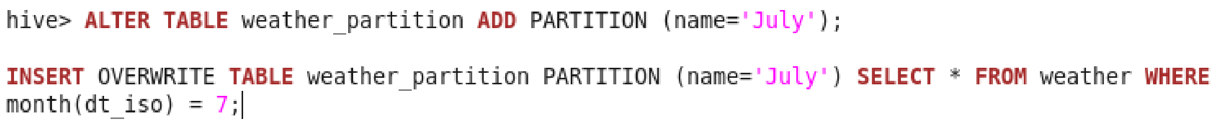


```
SELECT *
    FROM (SELECT w.city_name
            FROM weather AS w
            INNER JOIN energy AS e
            ON w.dt_iso = e.time
            WHERE w.temp < 383.15 OR w.temp > 303.15
            GROUP BY w.city_name)
            AS t1
    INNER JOIN (SELECT w.city_name
            FROM weather AS w
            INNER JOIN energy AS e
            ON w.dt_iso = e.time
            WHERE w.temp > 393.15 AND w.temp < 398.15
            GROUP BY w.city_name)
            AS t2
    ON t1.city_name = t2.city_name
    GROUP BY t1.city_name;
```

Creating and loading partitions into a new table for the weather dataset to look for performance gains:

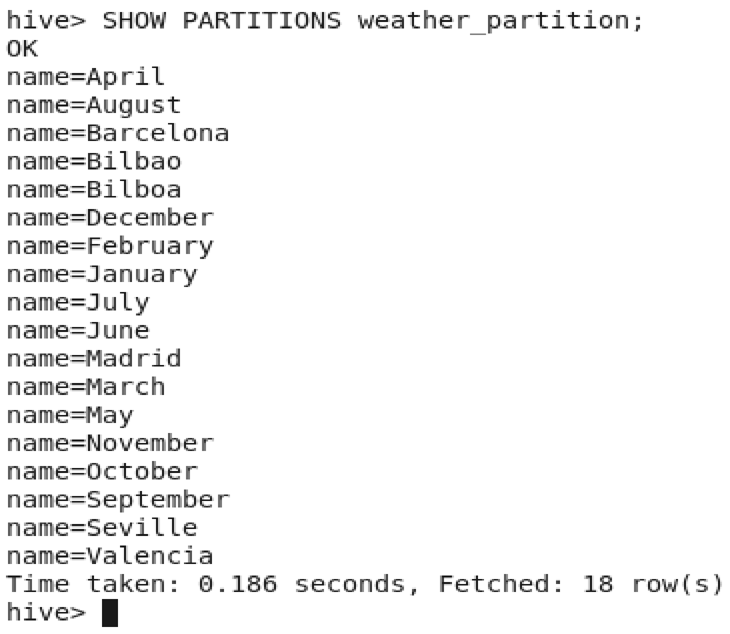


```
hive> CREATE TABLE weather_partition(
dt_iso STRING,
city_name STRING,
temp FLOAT,
temp_min FLOAT,
temp_max FLOAT,
pressure INT,
humidity INT,
wind_speed INT,
wind_deg INT,
rain_1h FLOAT,
rain_3h FLOAT,
snow_3h FLOAT,
clouds_all INT,
weather_id INT,
weather_main STRING,
weather_description STRING,
weather_icon STRING)
PARTITIONED BY(name STRING)
row format delimited fields terminated
by ',' stored as textfile;
```



```
hive> ALTER TABLE weather_partition ADD PARTITION (name='July');

INSERT OVERWRITE TABLE weather_partition PARTITION (name='July') SELECT * FROM weather WHERE
month(dt_iso) = 7;
```

The result of all the partitions:



```
hive> SHOW PARTITIONS weather_partition;
OK
name=April
name=August
name=Barcelona
name=Bilbao
name=Bilboa
name=December
name=February
name=January
name=July
name=June
name=Madrid
name=March
name=May
name=November
name=October
name=September
name=Seville
name=Valencia
Time taken: 0.186 seconds, Fetched: 18 row(s)
hive>
```

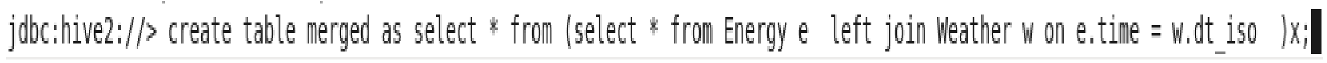
And some queries using the partitions and not using them. This did result in significantly quicker query times (~58seconds down from ~99 seconds).

```
SELECT AVG(temp) FROM weather_partition WHERE city_name = 'Valencia' AND month(dt_iso) = 11;

SELECT AVG(temp) FROM weather_partition WHERE name = 'Valencia' AND month(dt_iso) = 11;
```

**Kyle Son**:

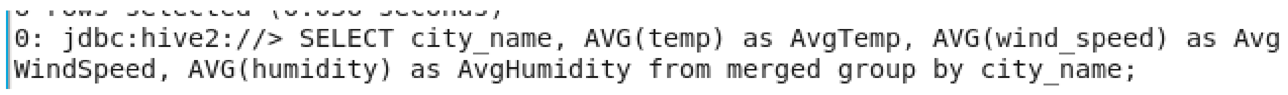
**Hive(Beeline)**

In hive, I used beeline command for the better visualization of the table then, I merged two tables to named merged

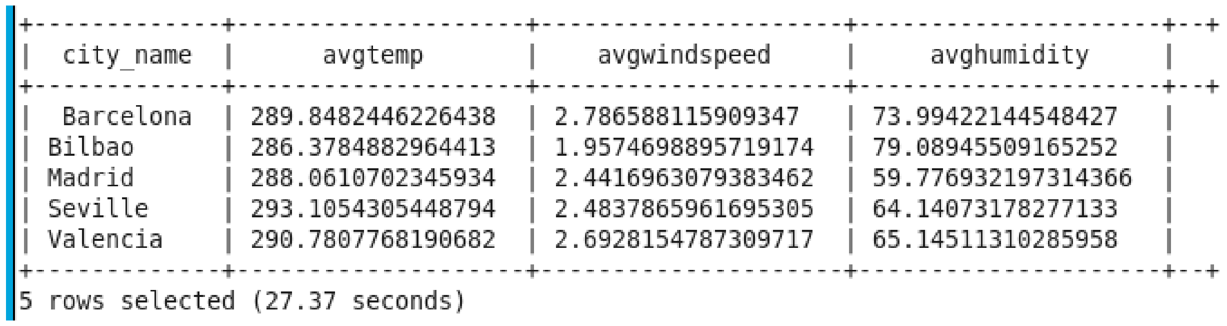


```
jdbc:hive2://> create table merged as select * from (select * from Energy e  left join Weather w on e.time = w.dt_iso  )x;
```

Displays avg temp, windspeed and humidity group by city name

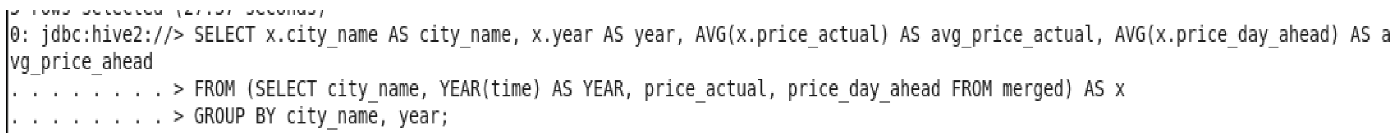


```
0: jdbc:hive2://> SELECT city_name, AVG(temp) as AvgTemp, AVG(wind_speed) as Avg
WindSpeed, AVG(humidity) as AvgHumidity from merged group by city_name;
```

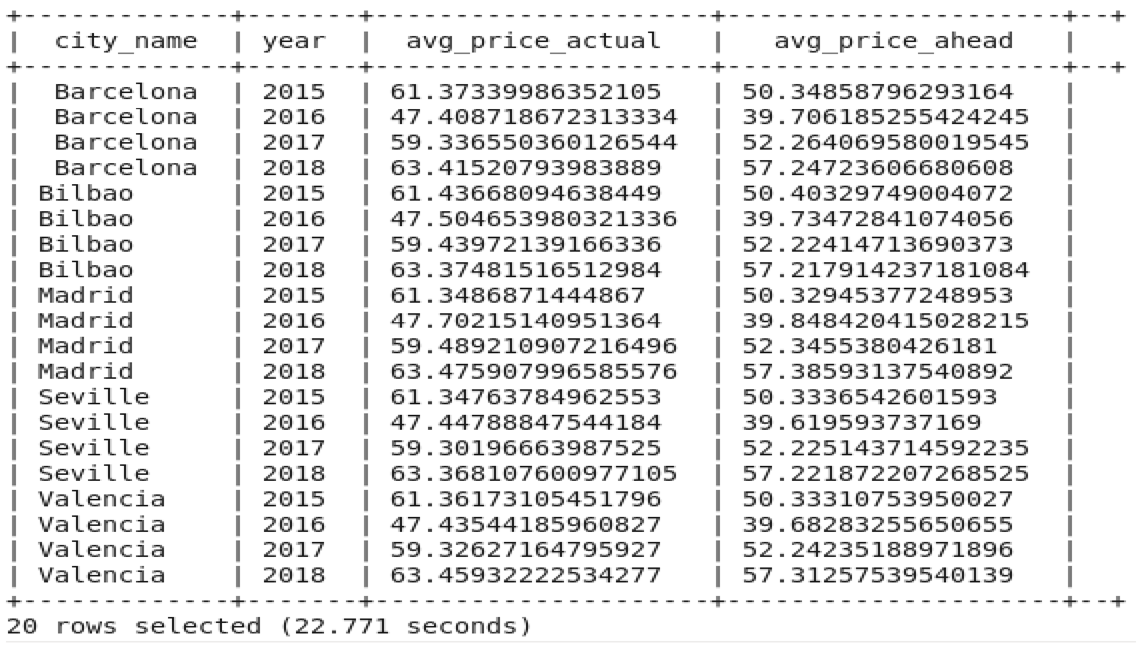


```
+-------------+-------------------+---------------------+---------------------+--+
|  city_name  |      avgtemp      |    avgwindspeed     |     avghumidity     |
+-------------+-------------------+---------------------+---------------------+--+
|  Barcelona  | 289.8482446226438 | 2.786588115909347   | 73.99422144548427   |
| Bilbao      | 286.3784882964413 | 1.9574698895719174  | 79.08945509165252   |
| Madrid      | 288.0610702345934 | 2.4416963079383462  | 59.776932197314366  |
| Seville     | 293.1054305448794 | 2.4837865961695305  | 64.14073178277133   |
| Valencia    | 290.7807768190682 | 2.6928154787309717  | 65.14511310285958   |
+-------------+-------------------+---------------------+---------------------+--+
5 rows selected (27.37 seconds)
```

Show the chronological change of the price group by city name



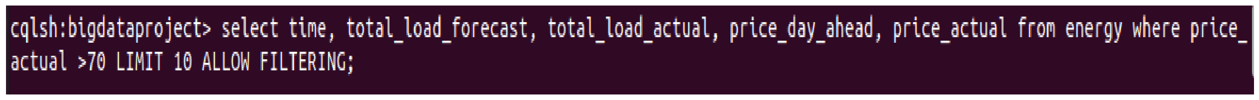
```
0: jdbc:hive2://> SELECT x.city_name AS city_name, x.year AS year, AVG(x.price_actual) AS avg_price_actual, AVG(x.price_day_ahead) AS a
vg_price_ahead
. . . . . . . . > FROM (SELECT city_name, YEAR(time) AS YEAR, price_actual, price_day_ahead FROM merged) AS x
. . . . . . . . > GROUP BY city_name, year;
```



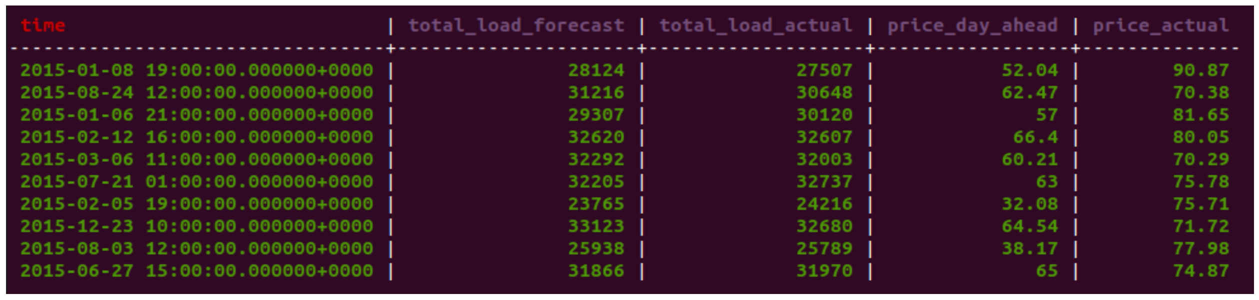
```
+-------------+-------+---------------------+---------------------+--+
|  city_name  | year  |  avg_price_actual   |   avg_price_ahead   |
+-------------+-------+---------------------+---------------------+--+
|  Barcelona  | 2015  | 61.37339986352105   | 50.34858796293164   |
|  Barcelona  | 2016  | 47.408718672313334  | 39.706185255424245  |
|  Barcelona  | 2017  | 59.336550360126544  | 52.264069580019545  |
|  Barcelona  | 2018  | 63.41520793983889   | 57.24723606680608   |
| Bilbao      | 2015  | 61.43668094638449   | 50.40329749004072   |
| Bilbao      | 2016  | 47.504653980321336  | 39.73472841074056   |
| Bilbao      | 2017  | 59.43972139166336   | 52.22414713690373   |
| Bilbao      | 2018  | 63.37481516512984   | 57.217914237181084  |
| Madrid      | 2015  | 61.3486871444867    | 50.32945377248953   |
| Madrid      | 2016  | 47.70215140951364   | 39.848420415028215  |
| Madrid      | 2017  | 59.489210907216496  | 52.3455380426181    |
| Madrid      | 2018  | 63.475907996585576  | 57.38593137540892   |
| Seville     | 2015  | 61.34763784962553   | 50.3336542601593    |
| Seville     | 2016  | 47.44788847544184   | 39.619593737169     |
| Seville     | 2017  | 59.30196663987525   | 52.225143714592235  |
| Seville     | 2018  | 63.368107600977105  | 57.221872207268525  |
| Valencia    | 2015  | 61.36173105451796   | 50.33310753950027   |
| Valencia    | 2016  | 47.43544185960827   | 39.68283255650655   |
| Valencia    | 2017  | 59.32627164795927   | 52.24235188971896   |
| Valencia    | 2018  | 63.45932222534277   | 57.31257539540139   |
+-------------+-------+---------------------+---------------------+--+
20 rows selected (22.771 seconds)
```

## Cassandra

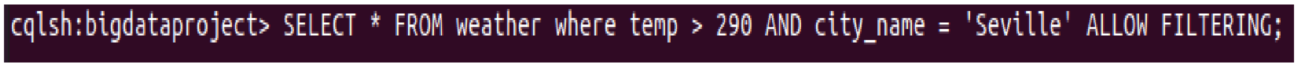
Perform some queries in cassandra, There is a limitation for our project because joining the table is not possible in cassandra



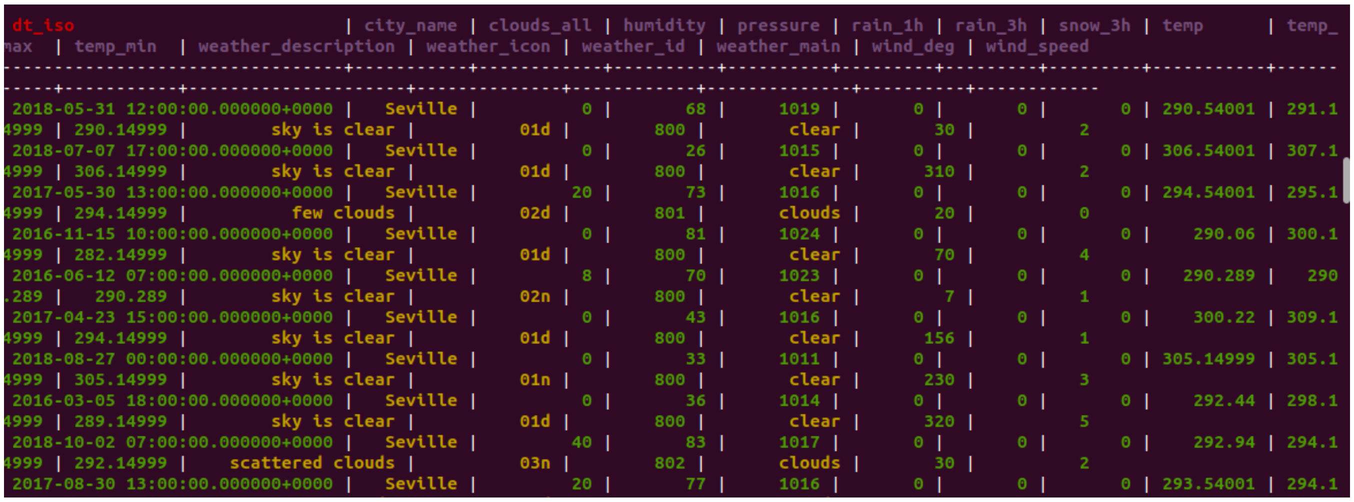
```
cqlsh:bigdataproject> select time, total_load_forecast, total_load_actual, price_day_ahead, price_actual from energy where price_
actual >70 LIMIT 10 ALLOW FILTERING;
```



```
 time                            | total_load_forecast | total_load_actual | price_day_ahead | price_actual
---------------------------------+---------------------+-------------------+-----------------+--------------
 2015-01-08 19:00:00.000000+0000 |               28124 |             27507 |           52.04 |        90.87
 2015-08-24 12:00:00.000000+0000 |               31216 |             30648 |           62.47 |        70.38
 2015-01-06 21:00:00.000000+0000 |               29307 |             30120 |              57 |        81.65
 2015-02-12 16:00:00.000000+0000 |               32620 |             32607 |            66.4 |        80.05
 2015-03-06 11:00:00.000000+0000 |               32292 |             32003 |           60.21 |        70.29
 2015-07-21 01:00:00.000000+0000 |               32205 |             32737 |              63 |        75.78
 2015-02-05 19:00:00.000000+0000 |               23765 |             24216 |           32.08 |        75.71
 2015-12-23 10:00:00.000000+0000 |               33123 |             32680 |           64.54 |        71.72
 2015-08-03 12:00:00.000000+0000 |               25938 |             25789 |           38.17 |        77.98
 2015-06-27 15:00:00.000000+0000 |               31866 |             31970 |              65 |        74.87
```



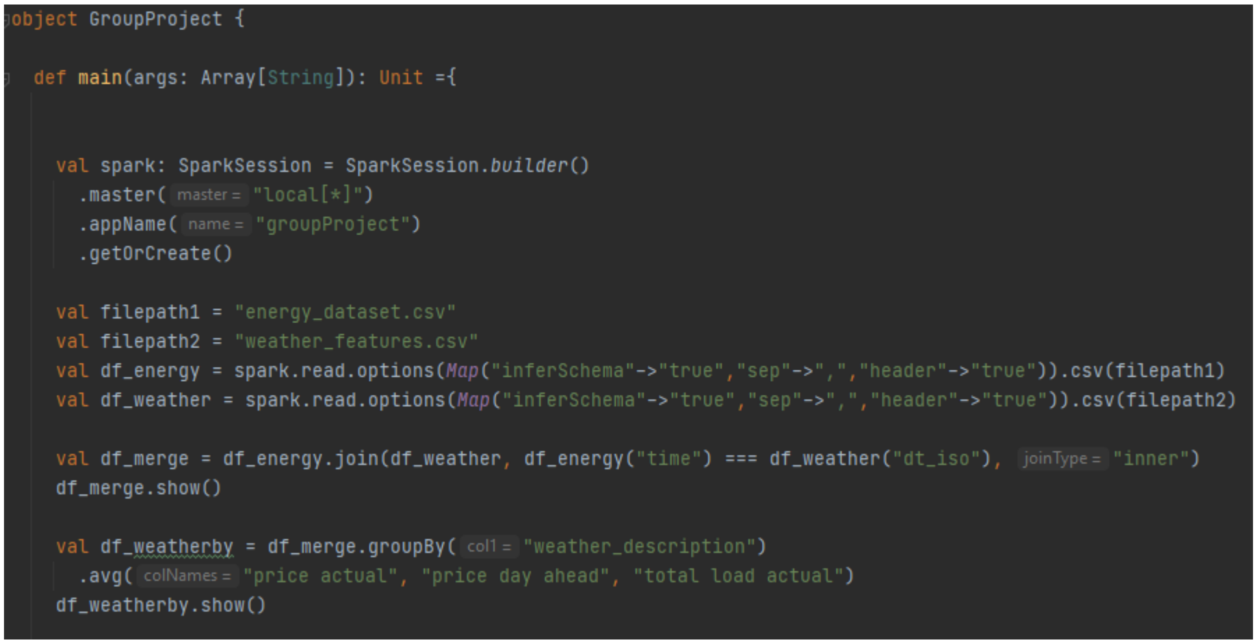
```
cqlsh:bigdataproject> SELECT * FROM weather where temp > 290 AND city_name = 'Seville' ALLOW FILTERING;
```



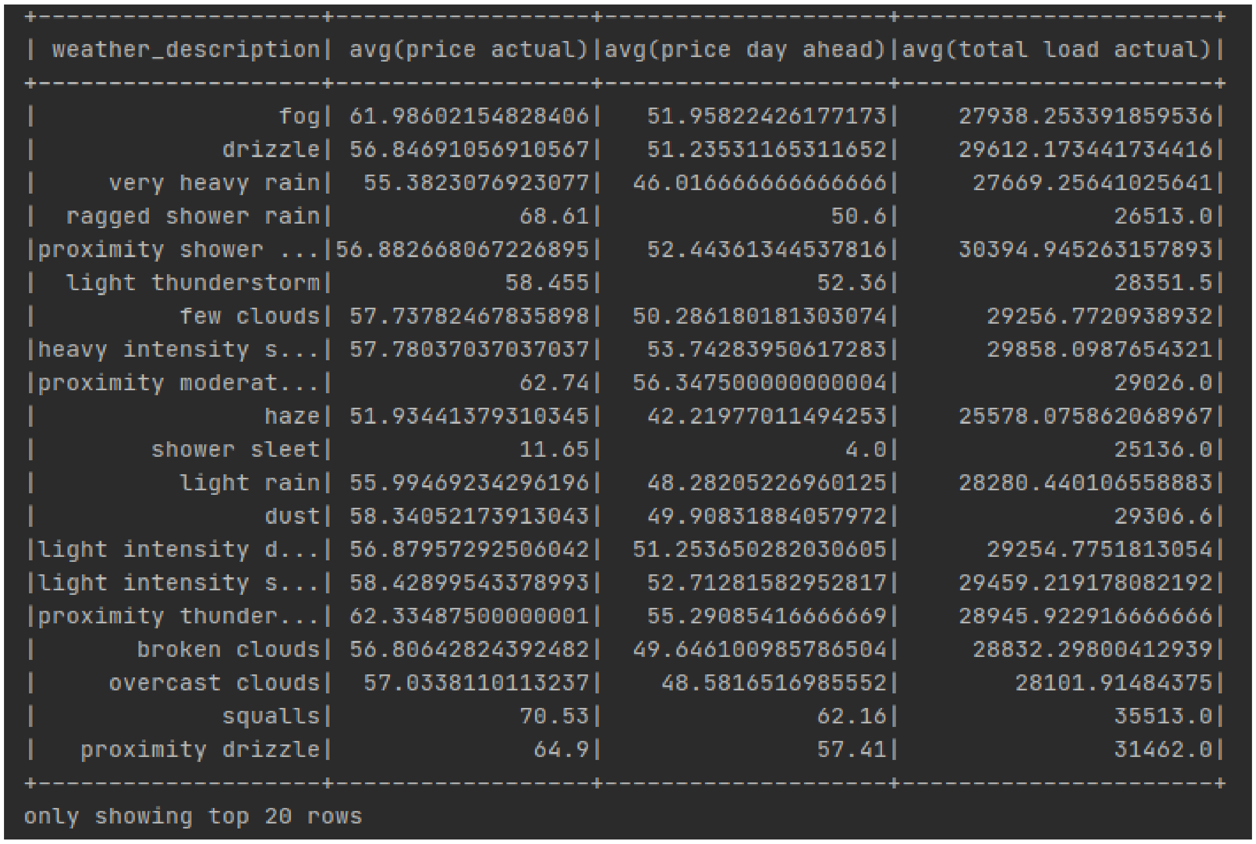
```
 dt_iso                          | city_name | clouds_all | humidity | pressure | rain_1h | rain_3h | snow_3h | temp      | temp_
max   | temp_min   | weather_description | weather_icon | weather_id | weather_main | wind_deg | wind_speed
---------------------------------+-----------+------------+----------+----------+---------+---------+---------+-----------+------
-----+------------+---------------------+--------------+------------+--------------+----------+------------
 2018-05-31 12:00:00.000000+0000 |   Seville |          0 |       68 |     1019 |       0 |       0 |       0 | 290.54001 | 291.1
4999 | 290.14999 |         sky is clear |          01d |        800 |        clear |       30 |          2
 2018-07-07 17:00:00.000000+0000 |   Seville |          0 |       26 |     1015 |       0 |       0 |       0 | 306.54001 | 307.1
4999 | 306.14999 |         sky is clear |          01d |        800 |        clear |      310 |          2
 2017-05-30 13:00:00.000000+0000 |   Seville |         20 |       73 |     1016 |       0 |       0 |       0 | 294.54001 | 295.1
4999 | 294.14999 |           few clouds |          02d |        801 |       clouds |       20 |          0
 2016-11-15 10:00:00.000000+0000 |   Seville |          0 |       81 |     1024 |       0 |       0 |       0 |    290.06 | 300.1
4999 | 282.14999 |         sky is clear |          01d |        800 |        clear |       70 |          4
 2016-06-12 07:00:00.000000+0000 |   Seville |          8 |       70 |     1023 |       0 |       0 |       0 |   290.289 |   290
.289 |   290.289 |         sky is clear |          02n |        800 |        clear |        7 |          1
 2017-04-23 15:00:00.000000+0000 |   Seville |          0 |       43 |     1016 |       0 |       0 |       0 |    300.22 | 309.1
4999 | 294.14999 |         sky is clear |          01d |        800 |        clear |      156 |          1
 2018-08-27 00:00:00.000000+0000 |   Seville |          0 |       33 |     1011 |       0 |       0 |       0 | 305.14999 | 305.1
4999 | 305.14999 |         sky is clear |          01n |        800 |        clear |      230 |          3
 2016-03-05 18:00:00.000000+0000 |   Seville |          0 |       36 |     1014 |       0 |       0 |       0 |    292.44 | 298.1
4999 | 289.14999 |         sky is clear |          01d |        800 |        clear |      320 |          5
 2018-10-02 07:00:00.000000+0000 |   Seville |         40 |       83 |     1017 |       0 |       0 |       0 |    292.94 | 294.1
4999 | 292.14999 |     scattered clouds |          03n |        802 |       clouds |       30 |          2
 2017-08-30 13:00:00.000000+0000 |   Seville |         20 |       77 |     1016 |       0 |       0 |       0 | 293.54001 | 294.1
```

**Spark(Scala)**

Load the csv file in the spark in the intellij, I merged two table by using spark sql join function and then do some queries based on joined table.



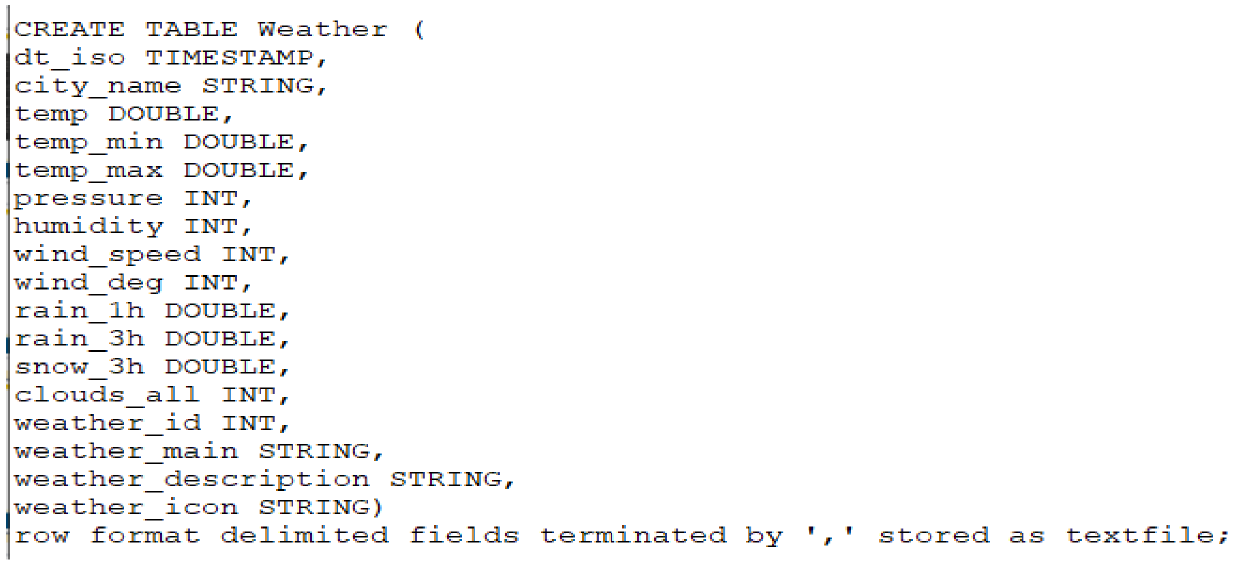
```
object GroupProject {

  def main(args: Array[String]): Unit ={


    val spark: SparkSession = SparkSession.builder()
      .master( master = "local[*]")
      .appName( name = "groupProject")
      .getOrCreate()

    val filepath1 = "energy_dataset.csv"
    val filepath2 = "weather_features.csv"
    val df_energy = spark.read.options(Map("inferSchema"->"true","sep"->",","header"->"true")).csv(filepath1)
    val df_weather = spark.read.options(Map("inferSchema"->"true","sep"->",","header"->"true")).csv(filepath2)

    val df_merge = df_energy.join(df_weather, df_energy("time") === df_weather("dt_iso"),  joinType = "inner")
    df_merge.show()

    val df_weatherby = df_merge.groupBy( col1 = "weather_description")
      .avg( colNames = "price actual", "price day ahead", "total load actual")
    df_weatherby.show()
```



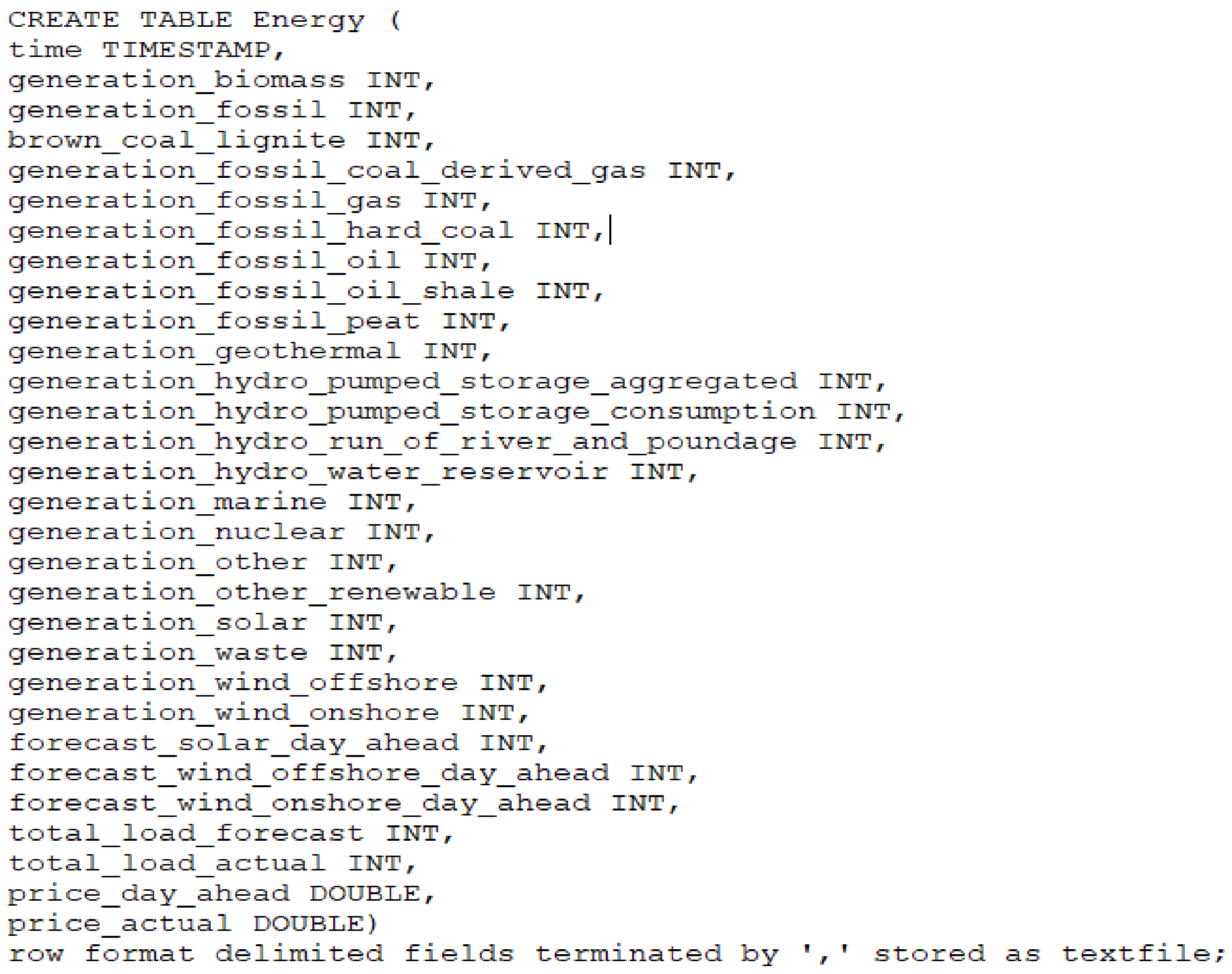
```
+--------------------+------------------+--------------------+---------------------+
| weather_description| avg(price actual)|avg(price day ahead)|avg(total load actual)|
+--------------------+------------------+--------------------+---------------------+
|                 fog| 61.98602154828406|   51.95822426177173|   27938.253391859536|
|             drizzle| 56.84691056910567|   51.23531165311652|   29612.173441734416|
|     very heavy rain|  55.3823076923077|   46.01666666666666|    27669.25641025641|
|   ragged shower rain|             68.61|                50.6|              26513.0|
|proximity shower ...|56.882668067226895|   52.44361344537816|   30394.945263157893|
|   light thunderstorm|            58.455|               52.36|              28351.5|
|          few clouds| 57.73782467835898|  50.286180181303074|     29256.7720938932|
|heavy intensity s...| 57.78037037037037|   53.74283950617283|     29858.0987654321|
|proximity moderat...|             62.74|  56.347500000000004|              29026.0|
|                haze| 51.93441379310345|   42.21977011494253|   25578.075862068967|
|        shower sleet|             11.65|                 4.0|              25136.0|
|          light rain| 55.99469234296196|   48.28205226960125|   28280.440106558883|
|                dust| 58.34052173913043|   49.90831884057972|              29306.6|
|light intensity d...| 56.87957292506042|  51.253650282030605|     29254.7751813054|
|light intensity s...| 58.42899543378993|   52.71281582952817|   29459.219178082192|
|proximity thunder...| 62.33487500000001|   55.29085416666669|   28945.922916666666|
|       broken clouds| 56.80642824392482|  49.646100985786504|    28832.29800412939|
|     overcast clouds|  57.0338110113237|    48.5816516985552|       28101.91484375|
|             squalls|             70.53|               62.16|              35513.0|
|   proximity drizzle|              64.9|               57.41|              31462.0|
+--------------------+------------------+--------------------+---------------------+
only showing top 20 rows
```

**Bill Yerkes**:

Create Tables in Hive:

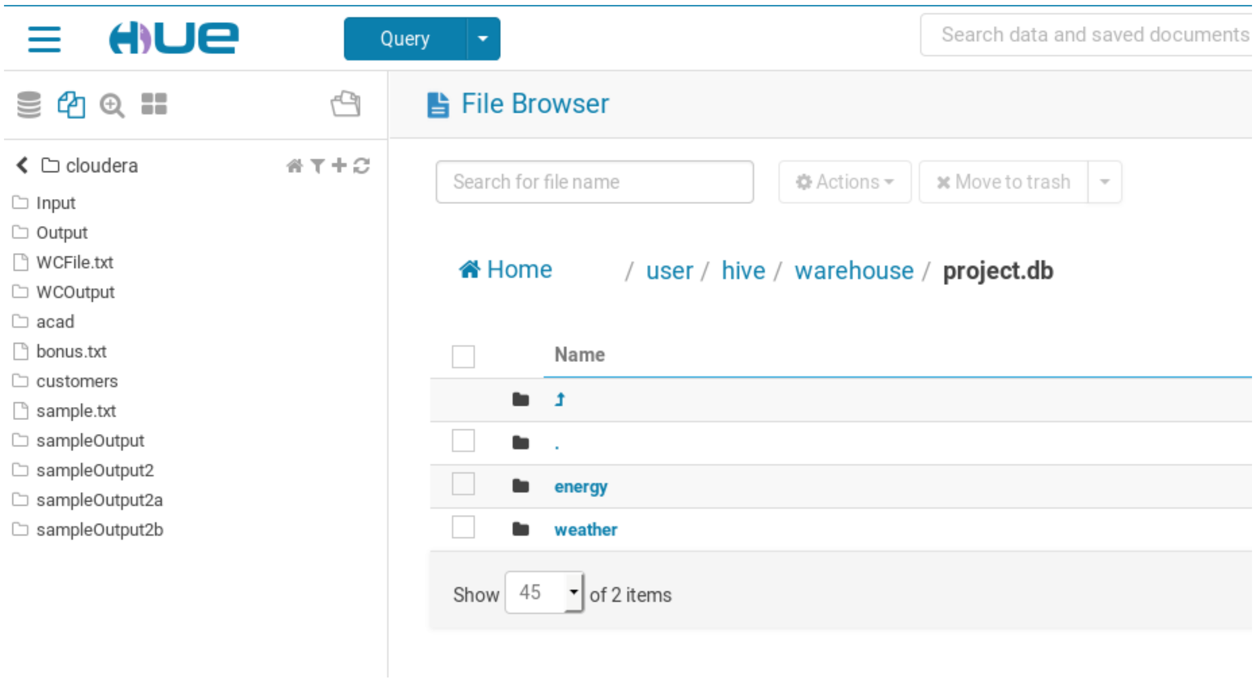


```
CREATE TABLE Weather (
dt_iso TIMESTAMP,
city_name STRING,
temp DOUBLE,
temp_min DOUBLE,
temp_max DOUBLE,
pressure INT,
humidity INT,
wind_speed INT,
wind_deg INT,
rain_1h DOUBLE,
rain_3h DOUBLE,
snow_3h DOUBLE,
clouds_all INT,
weather_id INT,
weather_main STRING,
weather_description STRING,
weather_icon STRING)
row format delimited fields terminated by ',' stored as textfile;
```



```
CREATE TABLE Energy (
time TIMESTAMP,
generation_biomass INT,
generation_fossil INT,
brown_coal_lignite INT,
generation_fossil_coal_derived_gas INT,
generation_fossil_gas INT,
generation_fossil_hard_coal INT,
generation_fossil_oil INT,
generation_fossil_oil_shale INT,
generation_fossil_peat INT,
generation_geothermal INT,
generation_hydro_pumped_storage_aggregated INT,
generation_hydro_pumped_storage_consumption INT,
generation_hydro_run_of_river_and_poundage INT,
generation_hydro_water_reservoir INT,
generation_marine INT,
generation_nuclear INT,
generation_other INT,
generation_other_renewable INT,
generation_solar INT,
generation_waste INT,
generation_wind_offshore INT,
generation_wind_onshore INT,
forecast_solar_day_ahead INT,
forecast_wind_offshore_day_ahead INT,
forecast_wind_onshore_day_ahead INT,
total_load_forecast INT,
total_load_actual INT,
price_day_ahead DOUBLE,
price_actual DOUBLE)
row format delimited fields terminated by ',' stored as textfile;
```

Hive Tables in Hadoop:



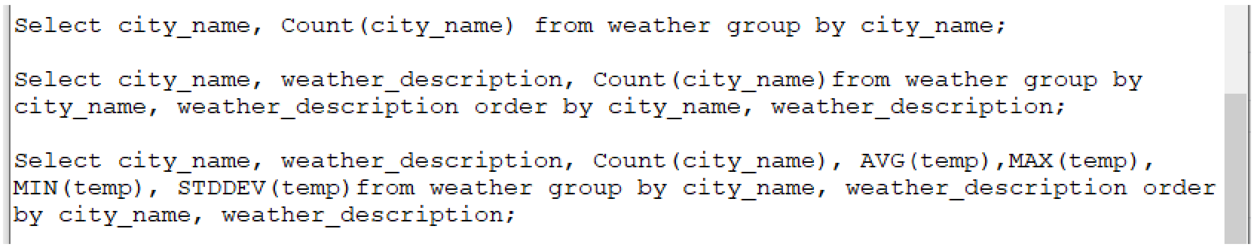
## Load the Data into Hive:

```
#-------------------------------------------------------------------
#Load the Data into Hive
#-------------------------------------------------------------------
load data local inpath '/home/cloudera/Downloads/weather_features.csv' into table Weather;

load data local inpath '/home/cloudera/Downloads/energy_dataset.csv' into table Energy;
```

## Some of the queries on Energy Data Set:

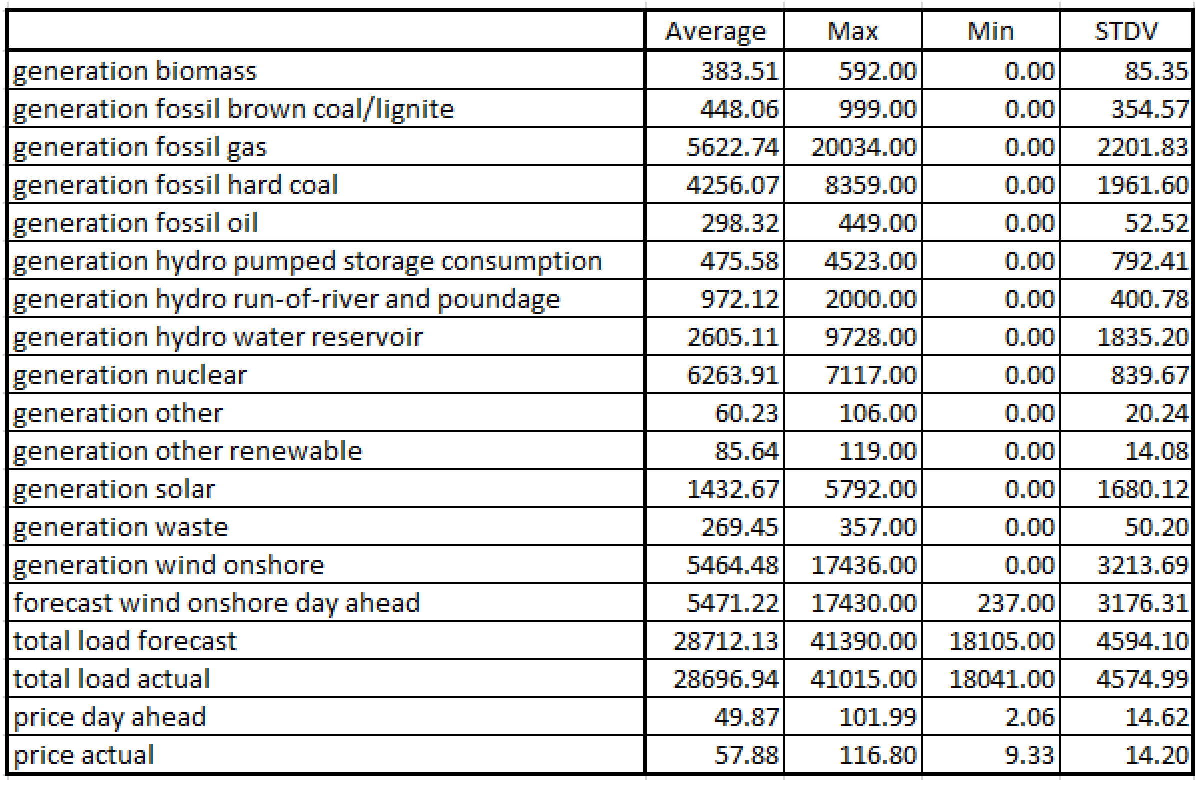
```
Select AVG(generation_biomass),MAX(generation_biomass),MIN(generation_biomass),
STDDEV(generation_biomass) FROM energy;

Select AVG(generation_fossil),MAX(generation_fossil),MIN(generation_fossil),
STDDEV(generation_fossil) FROM energy;

Select AVG(brown_coal_lignite),MAX(brown_coal_lignite),MIN(brown_coal_lignite),
STDDEV(brown_coal_lignite) FROM energy;
```

## Some of the queries on Weather Data Set:



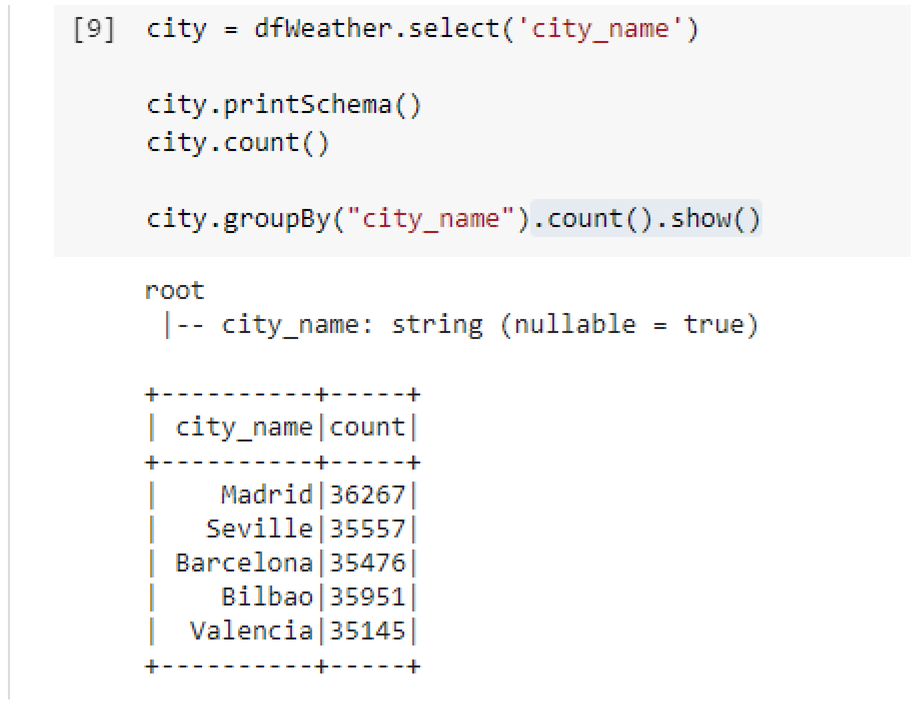
```
Select city_name, Count(city_name) from weather group by city_name;

Select city_name, weather_description, Count(city_name)from weather group by
city_name, weather_description order by city_name, weather_description;

Select city_name, weather_description, Count(city_name), AVG(temp),MAX(temp),
MIN(temp), STDDEV(temp)from weather group by city_name, weather_description order
by city_name, weather_description;
```

## Some of the metrics on the Data



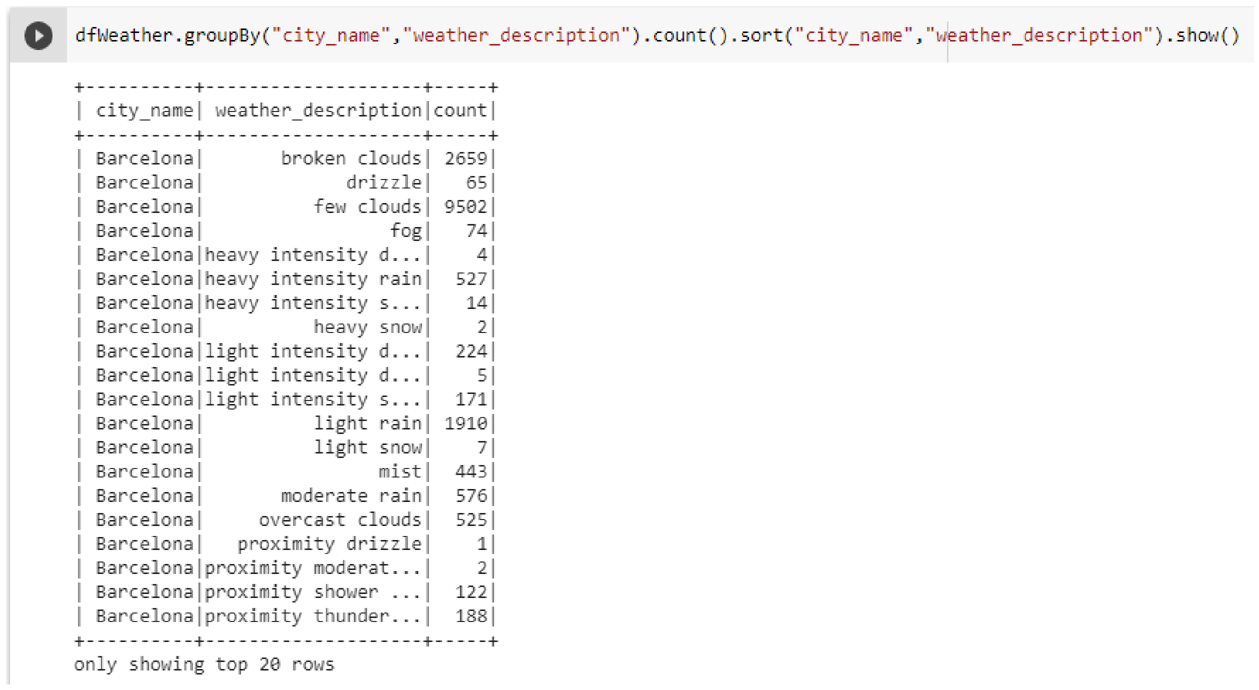
| | Average | Max | Min | STDV |
|---|---|---|---|---|
| generation biomass | 383.51 | 592.00 | 0.00 | 85.35 |
| generation fossil brown coal/lignite | 448.06 | 999.00 | 0.00 | 354.57 |
| generation fossil gas | 5622.74 | 20034.00 | 0.00 | 2201.83 |
| generation fossil hard coal | 4256.07 | 8359.00 | 0.00 | 1961.60 |
| generation fossil oil | 298.32 | 449.00 | 0.00 | 52.52 |
| generation hydro pumped storage consumption | 475.58 | 4523.00 | 0.00 | 792.41 |
| generation hydro run-of-river and poundage | 972.12 | 2000.00 | 0.00 | 400.78 |
| generation hydro water reservoir | 2605.11 | 9728.00 | 0.00 | 1835.20 |
| generation nuclear | 6263.91 | 7117.00 | 0.00 | 839.67 |
| generation other | 60.23 | 106.00 | 0.00 | 20.24 |
| generation other renewable | 85.64 | 119.00 | 0.00 | 14.08 |
| generation solar | 1432.67 | 5792.00 | 0.00 | 1680.12 |
| generation waste | 269.45 | 357.00 | 0.00 | 50.20 |
| generation wind onshore | 5464.48 | 17436.00 | 0.00 | 3213.69 |
| forecast wind onshore day ahead | 5471.22 | 17430.00 | 237.00 | 3176.31 |
| total load forecast | 28712.13 | 41390.00 | 18105.00 | 4594.10 |
| total load actual | 28696.94 | 41015.00 | 18041.00 | 4574.99 |
| price day ahead | 49.87 | 101.99 | 2.06 | 14.62 |
| price actual | 57.88 | 116.80 | 9.33 | 14.20 |

## Query and Results in Pyspark



```
[9] city = dfWeather.select('city_name')

    city.printSchema()
    city.count()

    city.groupBy("city_name").count().show()
```

```
root
 |-- city_name: string (nullable = true)

+----------+-----+
| city_name|count|
+----------+-----+
|    Madrid|36267|
|   Seville|35557|
| Barcelona|35476|
|    Bilbao|35951|
|  Valencia|35145|
+----------+-----+
```



```
dfWeather.groupBy("city_name","weather_description").count().sort("city_name","weather_description").show()
```

```
+----------+--------------------+-----+
| city_name| weather_description|count|
+----------+--------------------+-----+
| Barcelona|       broken clouds| 2659|
| Barcelona|             drizzle|   65|
| Barcelona|          few clouds| 9502|
| Barcelona|                 fog|   74|
| Barcelona|heavy intensity d...|    4|
| Barcelona|heavy intensity rain|  527|
| Barcelona|heavy intensity s...|   14|
| Barcelona|          heavy snow|    2|
| Barcelona|light intensity d...|  224|
| Barcelona|light intensity d...|    5|
| Barcelona|light intensity s...|  171|
| Barcelona|          light rain| 1910|
| Barcelona|          light snow|    7|
| Barcelona|                mist|  443|
| Barcelona|       moderate rain|  576|
| Barcelona|     overcast clouds|  525|
| Barcelona|   proximity drizzle|    1|
| Barcelona|proximity moderat...|    2|
| Barcelona|proximity shower ...|  122|
| Barcelona|proximity thunder...|  188|
+----------+--------------------+-----+
only showing top 20 rows
```

**Data and Parameters:**

**[Hourly energy demand generation and weather | Kaggle]**

This dataset contains 4 years of electrical consumption, generation, pricing, and weather data for Spain. Consumption and generation data were retrieved from ENTSOE a public portal for Transmission Service Operator (TSO) data. Settlement prices were obtained from the Spanish TSO Red Electric España. Weather data was purchased as part of a personal project from the Open Weather API for the 5 largest cities in Spain and made public here.

**GitHub Link:**

BillYerkes/CSEE5590_GroupProject (github.com)

**Project Management:**

**Implementation Status Report**

**Work completed:**

Joe Goldsich:

Analyze Weather Data Set using Hive, Hadoop and MapReduce.

Anna Johnson

Analyze Data in MySQL, using Sqoop and Hadoop

Kyle Son

Analyze Merged data in using Hive, Hadoop and Cassandra

Bill Yerkes

Analyze Energy Data Set using Hive, Hadoop and MapReduce.

**Work to be completed:**

Determine tools to be able to make predictions on the relationship between weather and energy.

Clean the data removing or correcting items so as to be able to make a more accurate prediction with the constructed model.

Create a model for predicting energy demand based upon weather information.

Test and verify the model.

Review the results and generate a presentation.

**Issues or Concerns:**

The date / time fields for the data are stored currently as text / string. These values will need to be converted to data / time. Additionally the data is stored as Greenwich Meridian time with an off set.

The group also needs to get exposure to how to use GraphX / MLLib to be able to make the predictions based upon the data and to create visualizations to present the results.

The group will need to determine how to move forward using Spark. Determination of using Scala or Python as the programming language.

**Conclusion:** The team has familiarized themselves with the tools learned over the first half of the semester. The team has also investigated the two datasets to be able to better understand how to construct our solution. As the team gains greater knowledge about the tools associated with Big Data Programming, we feel confident that we should be able to successfully accomplish our goals by the end of the semester.

**The scientist:**

UMKC Students / CSEE 5590 Big Data Programming. .

Anna Johnson, Joe Goldsich, Jongkook Son, and Bill Yerkes

**Users**

There are two main users for our application. Consumers of energy utility companies and producers of the energy being consumed.

**The Society**

Thanks to our application, the overall total utility/energy cost for our society would decrease because each subject would be able to act appropriately according to the prediction of the application. Producers would be able to expand or decrease their production line based on the weather forecast. Consumers of energy would be able to avoid huge amounts of electricity bills because of a more efficient system.

**Video Presentation:**

**https://youtu.be/DXZD-4CaSlI**

**References:**

**https://www.kaggle.com/nicholasjhana/energy-consumption-generation-prices-and-weather**

**Thousands caught off guard by rolling blackouts in Kansas City metro Monday (fox4kc.com)**

**https://www.texastribune.org/2021/02/17/texas-power-grid-failures/**

**https://www.forbes.com/sites/arielcohen/2021/02/19/texas-energy-crisis-is-an-epic-resilience-and-leadership-failure/?sh=46d08806eee8**

**California rolling blackouts during summer heat wave caused by 3 main factors, report says | Fox Business**