UMKC
UNIVERSITY OF MISSOURI
KANSAS CITY

Anna Johnson Kyle Son
Joe Goldsich Bill Yerkes

Analysis for determination of a relationship between energy demand and weather.

CSEE5590 – Big Data Programming

Content

- Introduction:
  - Goals and Objectives
  - Motivation
  - Significance
- Iteration 2:
  - Joe Goldsich: Hive - Hadoop - MapReduce (Weather Data Set)
  - Anna Johnson: Sqoop - Hadoop - MySQL
  - Kyle Son: Cassandra - Hive -Hadoop (Merge Table)
  - Bill Yerkes: Hive - Hadoop - MapReduce (Energy Data Set)
- Data Set Link
  - https://www.kaggle.com/nicholasjhana/energy-consumption-generation-prices-and-weather

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

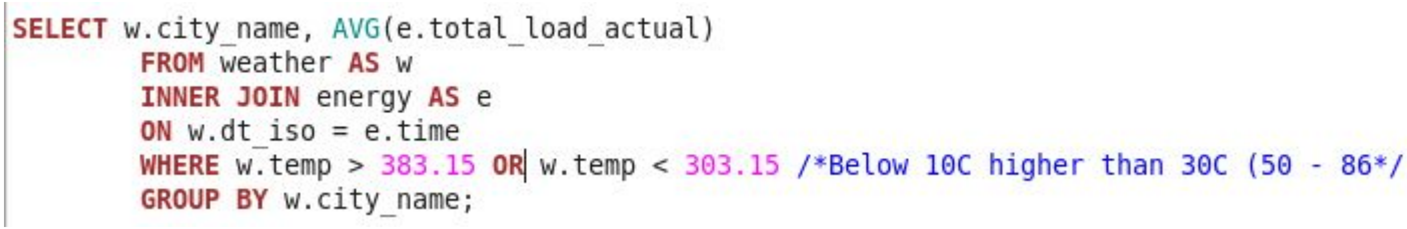
Significance

- 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.

Joe Goldsich

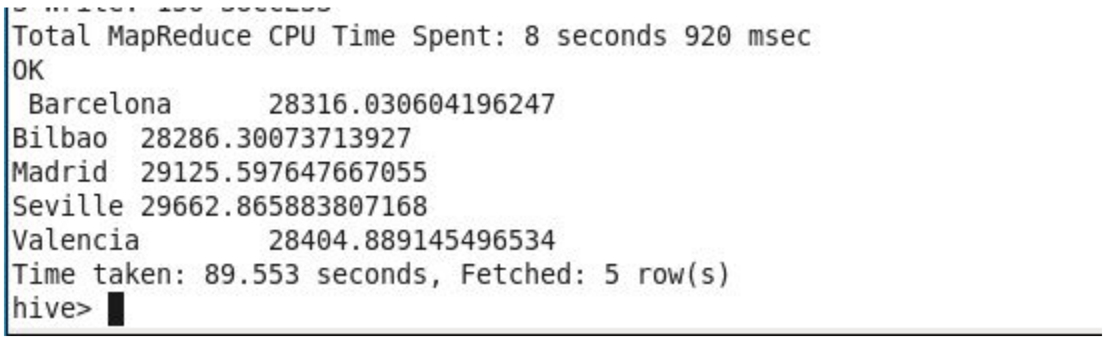
- Using Hive, I have practiced making queries on our data sets that might return useful information.
- First, I made simple queries like looking for a correlation between total energy load at different temperature ranges.
- When attempting more complicated queries there was a considerable dip in performance. So I then tried to take advantage of data partitioning that Hive allows.

Grab the energy use at temperatures above 30 C or below 10 C (86F, 50F)



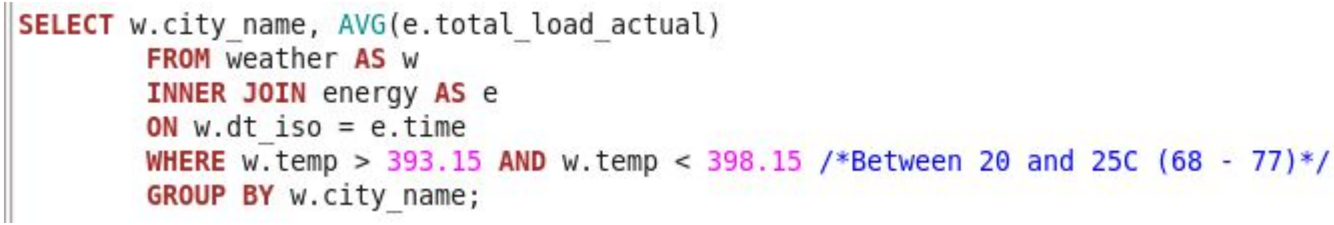
```
SELECT w.city_name, AVG(e.total_load_actual)
        FROM weather AS w
        INNER JOIN energy AS e
        ON w.dt_iso = e.time
        WHERE w.temp > 383.15 OR w.temp < 303.15 /*Below 10C higher than 30C (50 - 86*/
        GROUP BY w.city_name;
```

This produced:



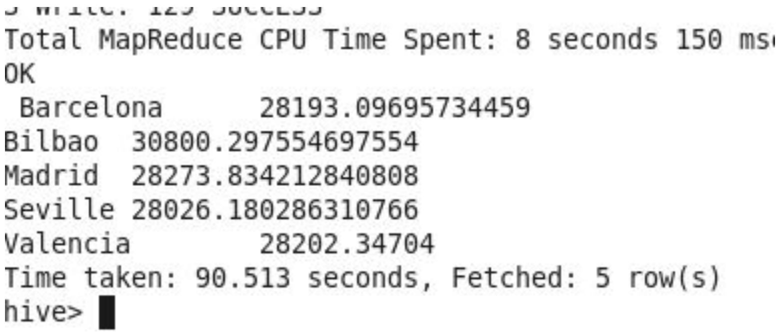
```
S Write: 150 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 920 msec
OK
 Barcelona      28316.030604196247
Bilbao  28286.30073713927
Madrid  29125.597647667055
Seville 29662.865883807168
Valencia        28404.889145496534
Time taken: 89.553 seconds, Fetched: 5 row(s)
hive> █
```

And then compare with temperatures between 20 C and 25 C (68 F to 77 F)



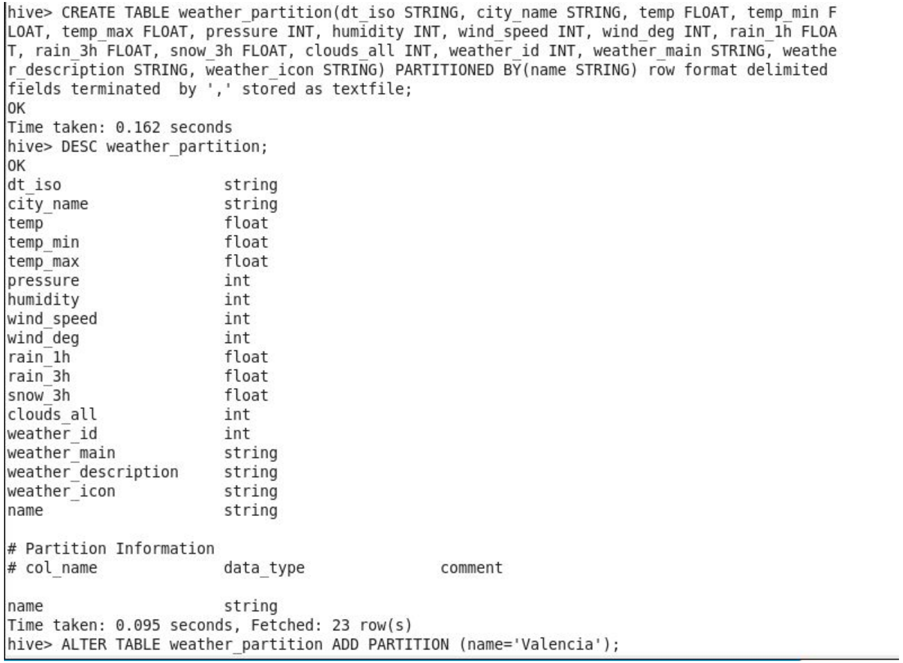
```
SELECT w.city_name, AVG(e.total_load_actual)
        FROM weather AS w
        INNER JOIN energy AS e
        ON w.dt_iso = e.time
        WHERE w.temp > 393.15 AND w.temp < 398.15 /*Between 20 and 25C (68 - 77)*/
        GROUP BY w.city_name;
```

And this produced:



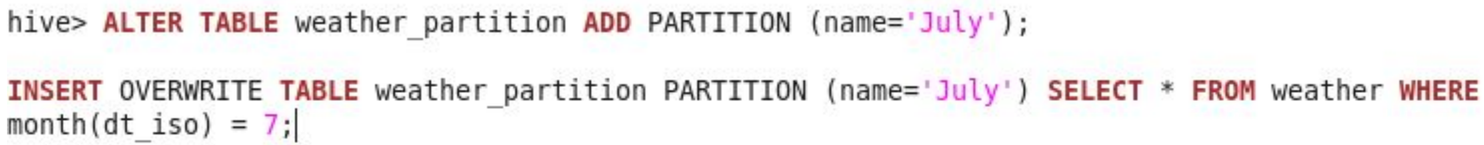
```
S Write: 125 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 150 ms
OK
 Barcelona      28193.09695734459
Bilbao  30800.297554697554
Madrid  28273.834212840808
Seville 28026.180286310766
Valencia        28202.34704
Time taken: 90.513 seconds, Fetched: 5 row(s)
hive> 
```

While attempting to conduct slightly more complex queries (nested selects and joins) ran into some significant performance issues (one query took over 10 minutes to execute). So I then tried to partition the data and see if that made for better performance.

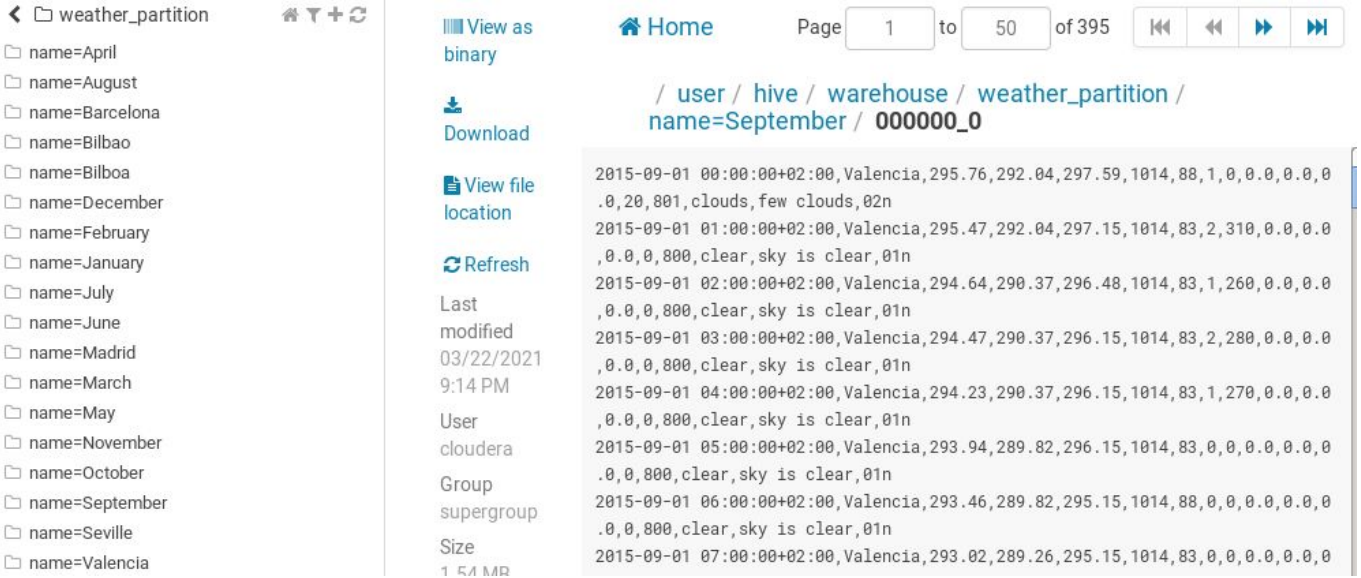


```
hive> CREATE TABLE weather_partition(dt_iso STRING, city_name STRING, temp FLOAT, temp_min F
LOAT, temp_max FLOAT, pressure INT, humidity INT, wind_speed INT, wind_deg INT, rain_1h FLOA
T, rain_3h FLOAT, snow_3h FLOAT, clouds_all INT, weather_id INT, weather_main STRING, weathe
r_description STRING, weather_icon STRING) PARTITIONED BY(name STRING) row format delimited
fields terminated  by ',' stored as textfile;
OK
Time taken: 0.162 seconds
hive> DESC weather_partition;
OK
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
name                    string

# Partition Information
# col_name              data_type               comment

name                    string
Time taken: 0.095 seconds, Fetched: 23 row(s)
hive> ALTER TABLE weather_partition ADD PARTITION (name='Valencia');
```

Some partitions examples:



```
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;
```



```
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
```

Last modified 03/22/2021 9:14 PM

User cloudera

Group supergroup

Size 1.54 MB

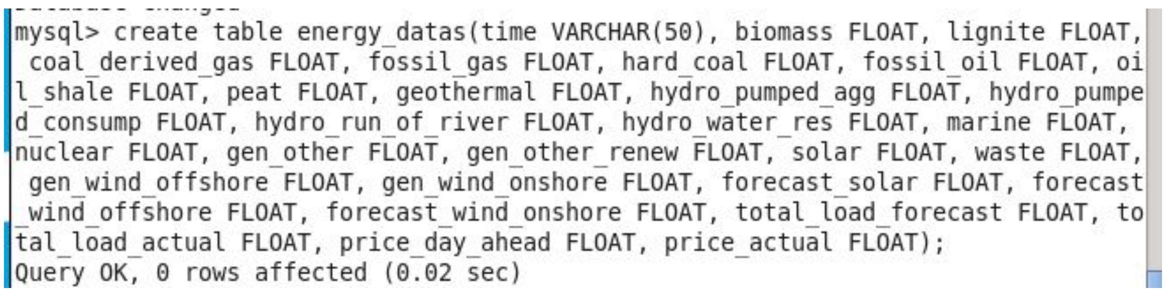
/ user / hive / warehouse / weather_partition / name=September / 000000_0

```
2015-09-01 00:00:00+02:00,Valencia,295.76,292.04,297.59,1014,88,1,0,0.0,0.0,0
.0,20,801,clouds,few clouds,02n
2015-09-01 01:00:00+02:00,Valencia,295.47,292.04,297.15,1014,83,2,310,0.0,0.0
,0.0,0,800,clear,sky is clear,01n
2015-09-01 02:00:00+02:00,Valencia,294.64,290.37,296.48,1014,83,1,260,0.0,0.0
,0.0,0,800,clear,sky is clear,01n
2015-09-01 03:00:00+02:00,Valencia,294.47,290.37,296.15,1014,83,2,280,0.0,0.0
,0.0,0,800,clear,sky is clear,01n
2015-09-01 04:00:00+02:00,Valencia,294.23,290.37,296.15,1014,83,1,270,0.0,0.0
,0.0,0,800,clear,sky is clear,01n
2015-09-01 05:00:00+02:00,Valencia,293.94,289.82,296.15,1014,83,0,0,0.0,0.0,0
.0,0,800,clear,sky is clear,01n
2015-09-01 06:00:00+02:00,Valencia,293.46,289.82,295.15,1014,88,0,0,0.0,0.0,0
.0,0,800,clear,sky is clear,01n
2015-09-01 07:00:00+02:00,Valencia,293.02,289.26,295.15,1014,83,0,0,0.0,0.0,0
```

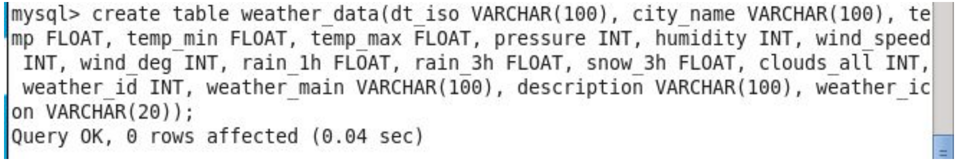
Anna Johnson

- In MySQL, I created tables for both the energy and weather features datasets
- I populated the tables by loading the data from the csv files
- I used MySQL to run queries on the tables to better understand the data
- I then used Sqoop to import the tables into HDFS
- Sqoop can be used to transfer our databases between the Hadoop/ Hive ecosystem and the relational database system of MySQL, which provides different functionality for querying the data.

Creating Tables in MySQL

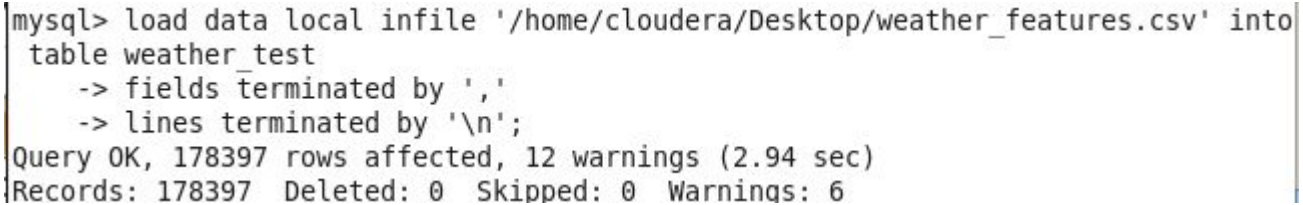


```
mysql> create table energy_datas(time VARCHAR(50), biomass FLOAT, lignite FLOAT,
 coal_derived_gas FLOAT, fossil_gas FLOAT, hard_coal FLOAT, fossil_oil FLOAT, oi
l_shale FLOAT, peat FLOAT, geothermal FLOAT, hydro_pumped_agg FLOAT, hydro_pumpe
d_consump FLOAT, hydro_run_of_river FLOAT, hydro_water_res 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)
```



```
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)
```

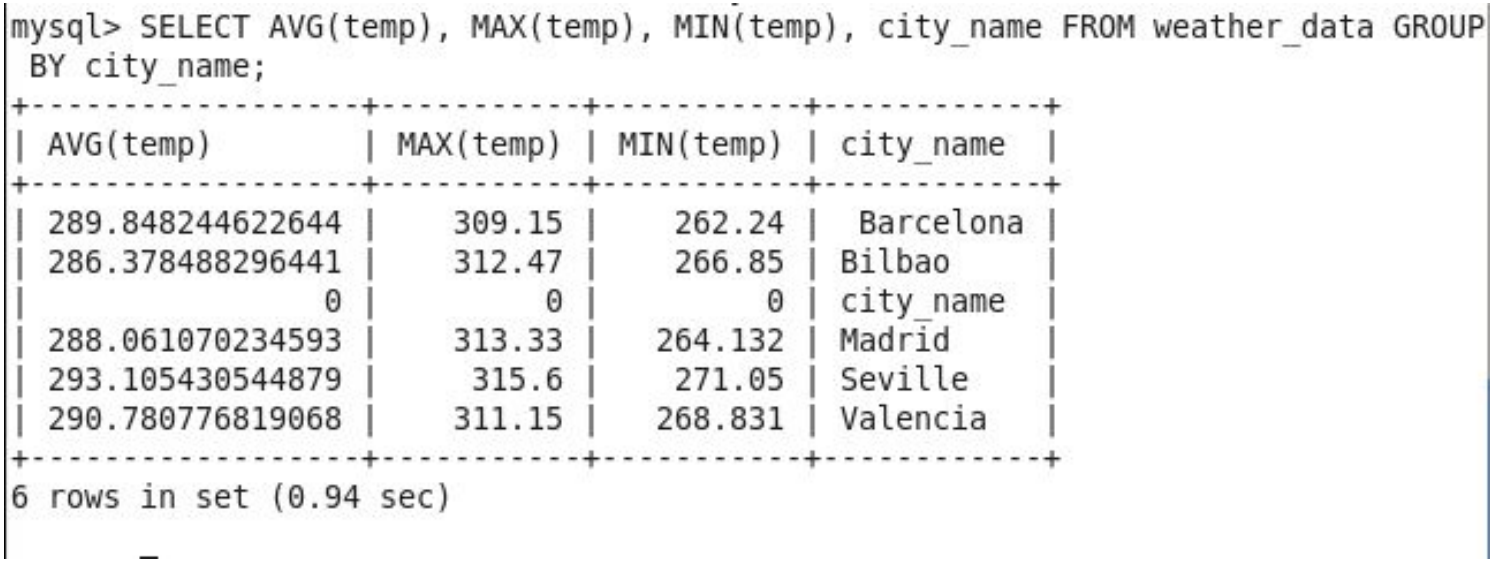
Populating tables with data:



```
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
```

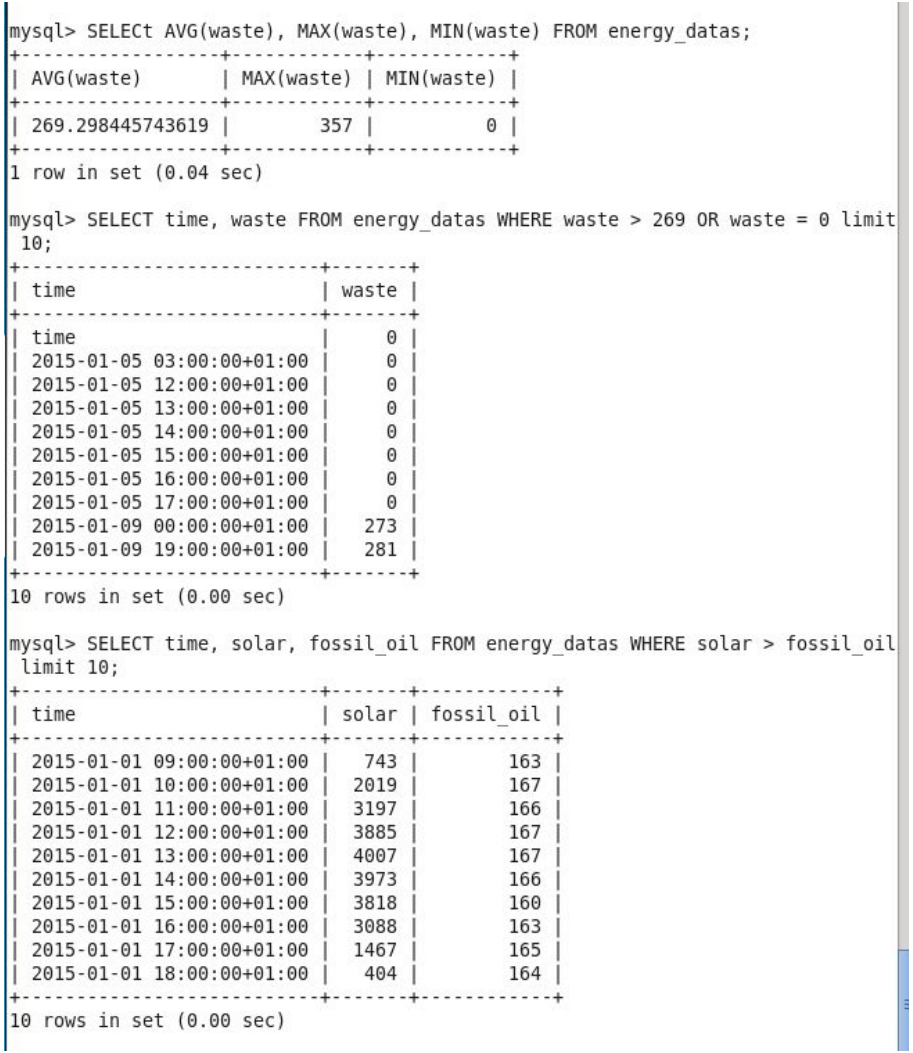
Example Queries in MySQL:

Weather data:



```
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)
```

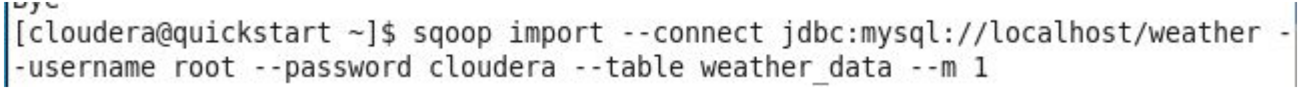
Energy data:



```
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)
```

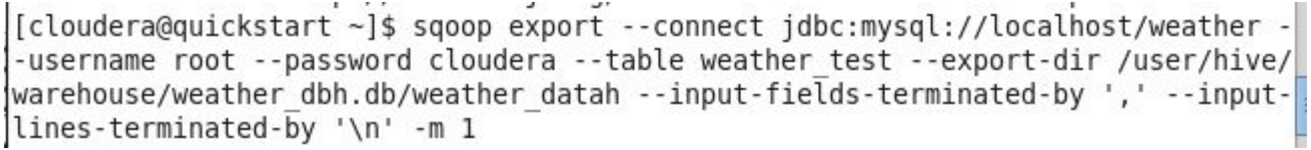
Transferring Data with Sqoop

Importing from MySQL to HDFS:



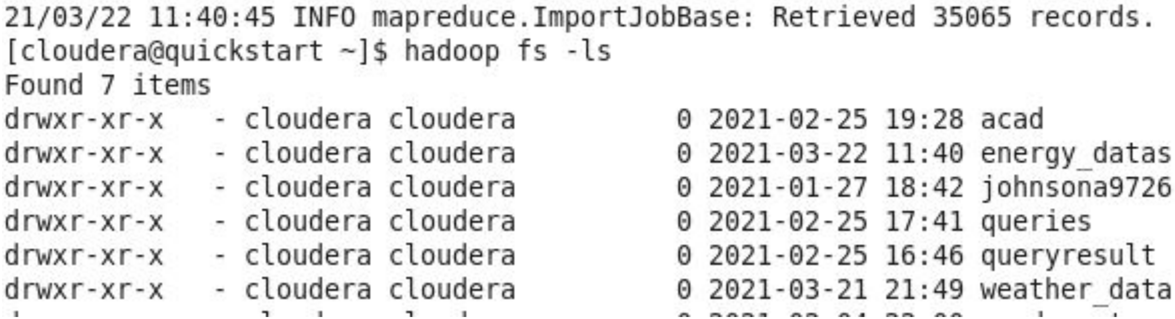
```
Bye
[cloudera@quickstart ~]$ sqoop import --connect jdbc:mysql://localhost/weather -
-username root --password cloudera --table weather_data --m 1
```

Exporting between Hive and MySQL:



```
[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
```

energy_datas and weather_data in HDFS after importing from MySQL:

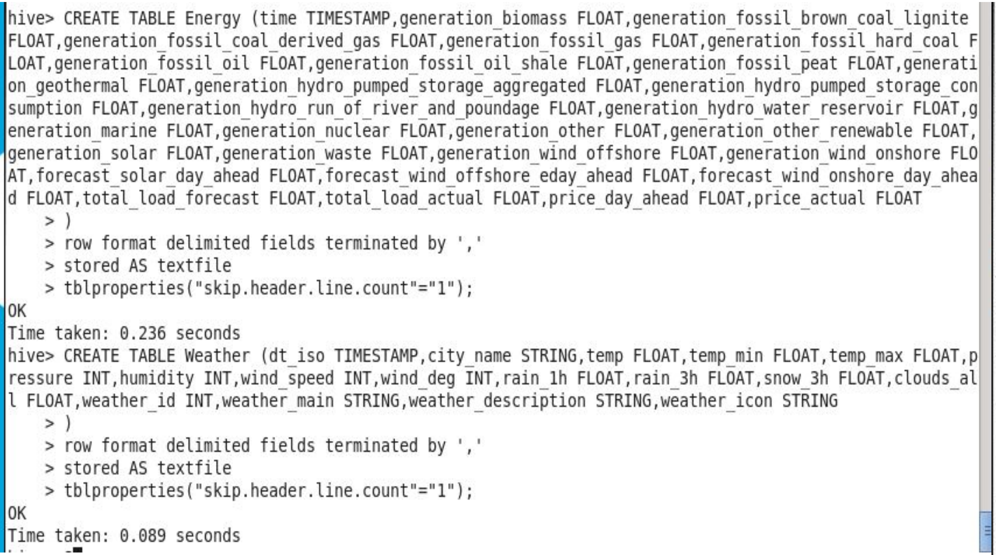


```
21/03/22 11:40:45 INFO mapreduce.ImportJobBase: Retrieved 35065 records.
[cloudera@quickstart ~]$ hadoop fs -ls
Found 7 items
drwxr-xr-x   - cloudera cloudera          0 2021-02-25 19:28 acad
drwxr-xr-x   - cloudera cloudera          0 2021-03-22 11:40 energy_datas
drwxr-xr-x   - cloudera cloudera          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
```

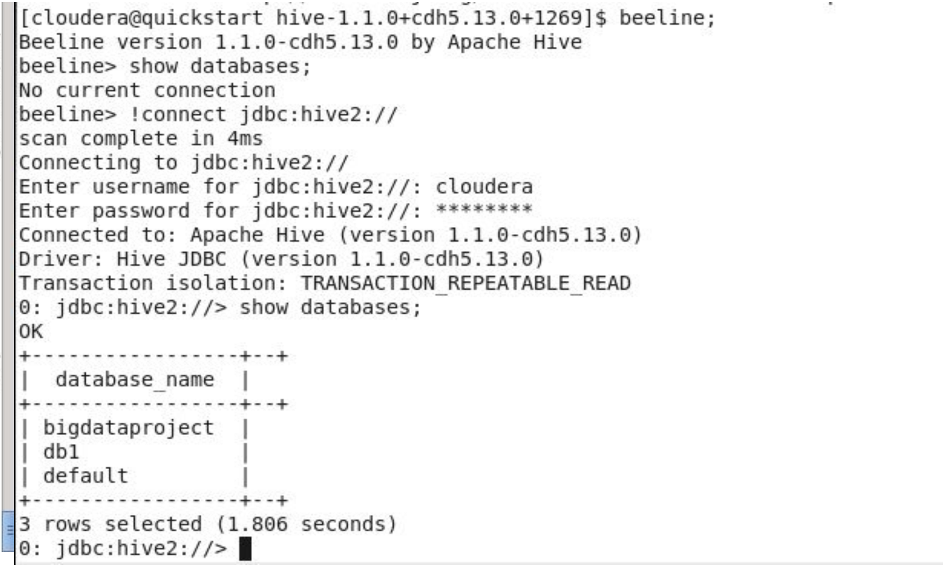
Kyle Son

- Instead of Hive CLI, I used HIVE beeline for better visualization of table
- In Hive, Join two tables on (dt_iso = time) and perform queries on it
- Using cassandra, perform some queries on each table
- It is not possible to join two tables in cassandra
- I used sparksql on intellij with scala for query the data

HIVE(Beeline)



```
hive> CREATE TABLE Energy (time TIMESTAMP,generation_biomass FLOAT,generation_fossil_brown_coal_lignite
FLOAT,generation_fossil_coal_derived_gas FLOAT,generation_fossil_gas FLOAT,generation_fossil_hard_coal F
LOAT,generation_fossil_oil FLOAT,generation_fossil_oil_shale FLOAT,generation_fossil_peat FLOAT,generati
on_geothermal FLOAT,generation_hydro_pumped_storage_aggregated FLOAT,generation_hydro_pumped_storage_con
sumption FLOAT,generation_hydro_run_of_river_and_poundage FLOAT,generation_hydro_water_reservoir FLOAT,g
eneration_marine FLOAT,generation_nuclear FLOAT,generation_other FLOAT,generation_other_renewable FLOAT,
generation_solar FLOAT,generation_waste FLOAT,generation_wind_offshore FLOAT,generation_wind_onshore FLO
AT,forecast_solar_day_ahead FLOAT,forecast_wind_offshore_eday_ahead FLOAT,forecast_wind_onshore_day_ahea
d FLOAT,total_load_forecast FLOAT,total_load_actual FLOAT,price_day_ahead FLOAT,price_actual FLOAT
    > )
    > row format delimited fields terminated by ','
    > stored AS textfile
    > tblproperties("skip.header.line.count"="1");
OK
Time taken: 0.236 seconds
hive> CREATE TABLE Weather (dt_iso TIMESTAMP,city_name STRING,temp FLOAT,temp_min FLOAT,temp_max FLOAT,p
ressure INT,humidity INT,wind_speed INT,wind_deg INT,rain_1h FLOAT,rain_3h FLOAT,snow_3h FLOAT,clouds_al
l FLOAT,weather_id INT,weather_main STRING,weather_description STRING,weather_icon STRING
    > )
    > row format delimited fields terminated by ','
    > stored AS textfile
    > tblproperties("skip.header.line.count"="1");
OK
Time taken: 0.089 seconds
```

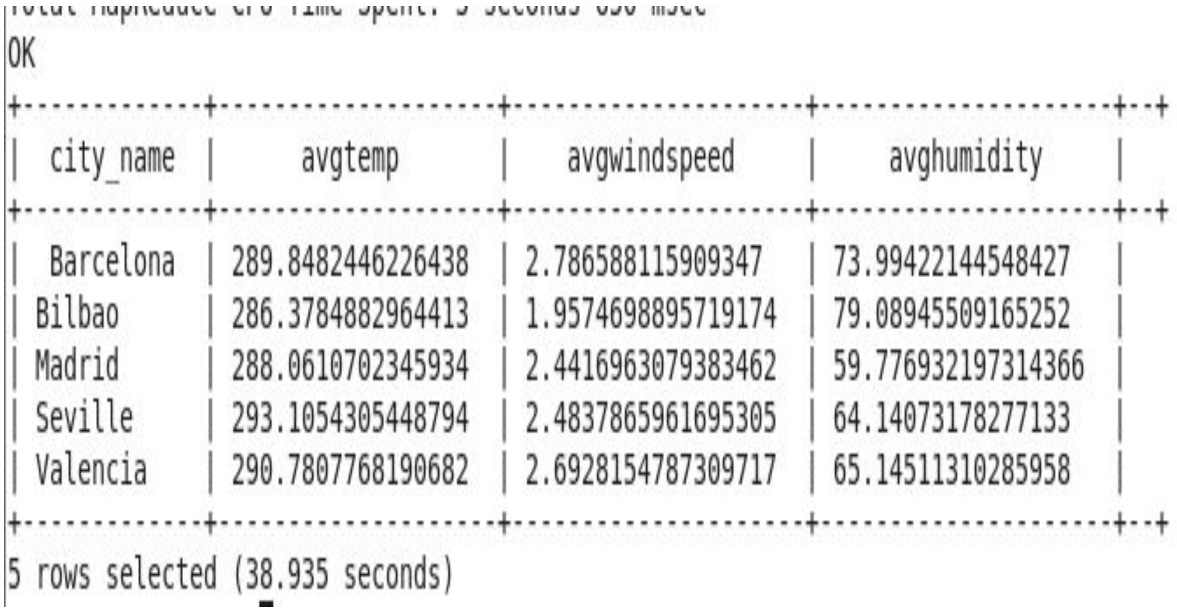


```
[cloudera@quickstart hive-1.1.0+cdh5.13.0+1269]$ beeline;
Beeline version 1.1.0-cdh5.13.0 by Apache Hive
beeline> show databases;
No current connection
beeline> !connect jdbc:hive2://
scan complete in 4ms
Connecting to jdbc:hive2://
Enter username for jdbc:hive2://: cloudera
Enter password for jdbc:hive2://: ********
Connected to: Apache Hive (version 1.1.0-cdh5.13.0)
Driver: Hive JDBC (version 1.1.0-cdh5.13.0)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://> show databases;
OK
+----------------+--+
|  database_name  |
+----------------+--+
| bigdataproject  |
| db1             |
| default         |
+----------------+--+
3 rows selected (1.806 seconds)
0: jdbc:hive2://>
```

create table merged as select * from (select * from Energy e left join Weather w on e.time = w.dt_iso)x

=> **From This query, I created a merged table of energy and weather**

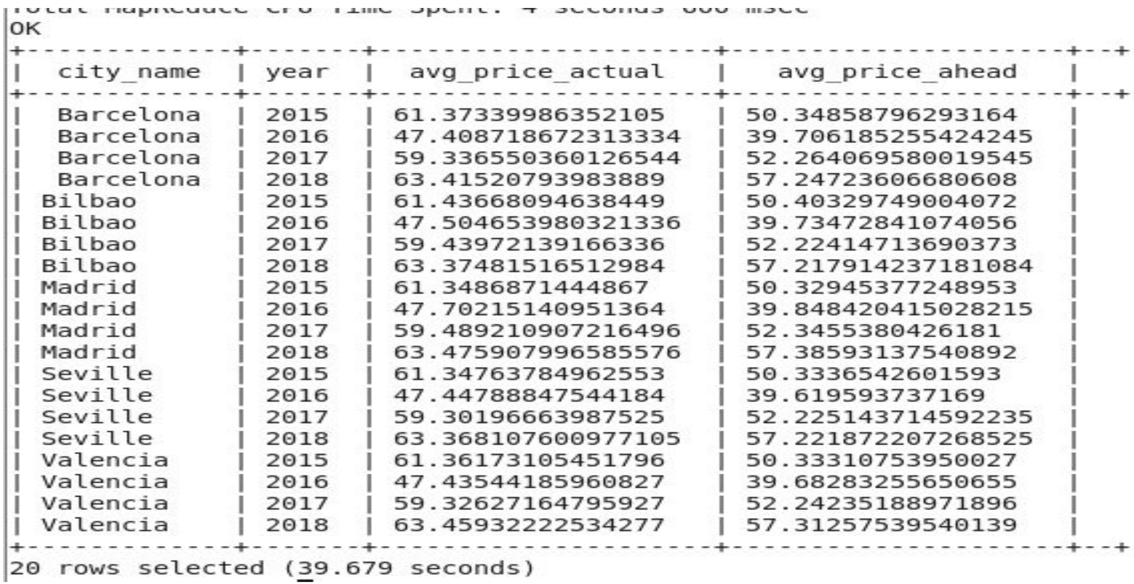
SELECT city_name, AVG(temp) as AvgTemp, AVG(wind_speed) as AvgWindSpeed, AVG(humidity) as AvgHumidity
from merged group by city_name;



```
Total MapReduce CPU Time Spent: 5 seconds 830 msec
OK
+------------+--------------------+--------------------+--------------------+--+
|  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 (38.935 seconds)
```

=> From This query, I displayed each city’s average temperature windspeed and humidity group by city name

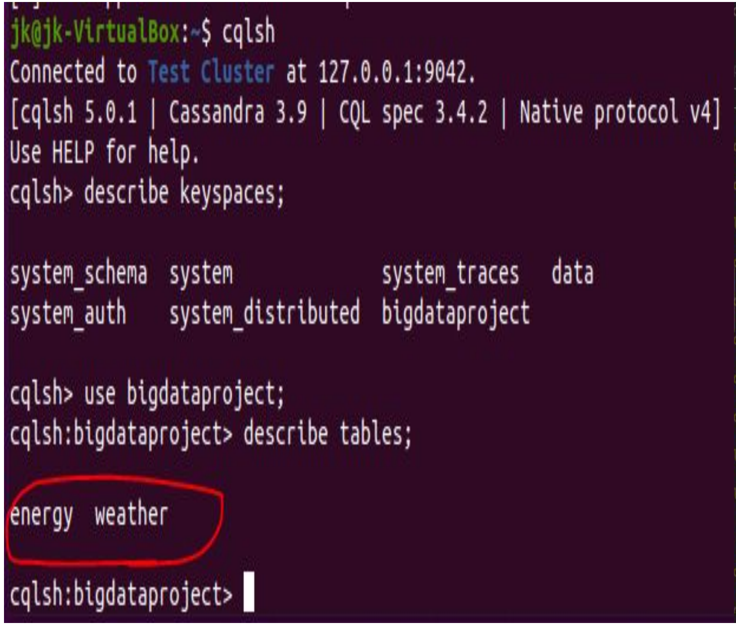
SELECT x.city_name AS city_name, x.year AS year, AVG(x.price_actual) AS avg_price_actual, AVG(x.price_day_ahead) AS avg_price_ahead
FROM (SELECT city_name, YEAR(time) AS YEAR, price_actual, price_day_ahead FROM merged) AS x
GROUP BY city_name, year;



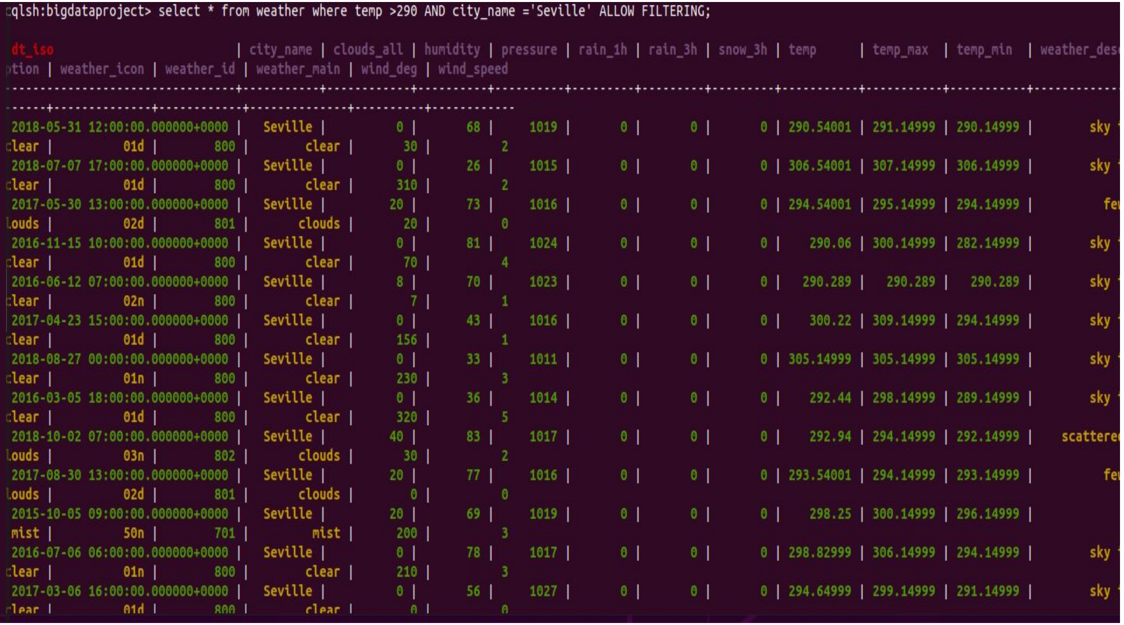
```
Total MapReduce CPU Time Spent: 4 seconds [illegible] msec
OK
+-------------+-------+---------------------+---------------------+--+
|  city_name  | year  |  avg_price_actual   |   avg_price_ahead   |
+-------------+-------+---------------------+---------------------+--+
|  Barcelona  | 2015  | 61.37339986352105   | 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 (39.679 seconds)
```

=> From This query, I displayed avg actual price and ahead price group by city and year to show the chronological change of the electricity price

CASSANDRA

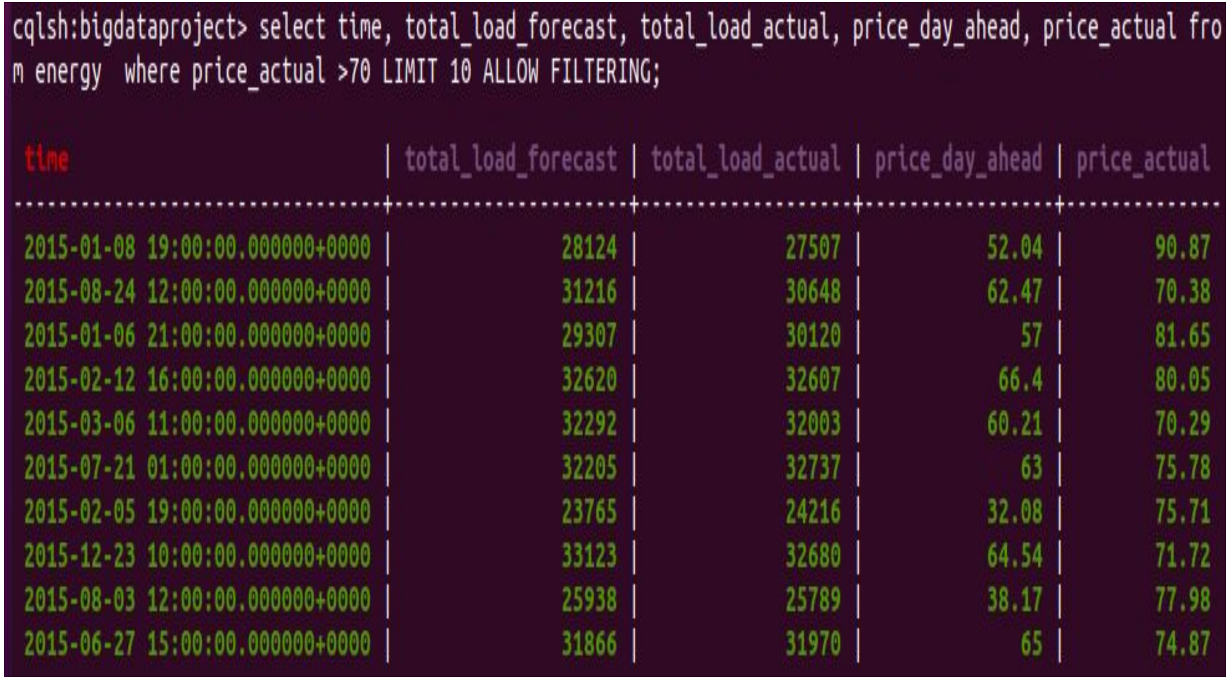


```
jk@jk-VirtualBox:~$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.9 | CQL spec 3.4.2 | Native protocol v4]
Use HELP for help.
cqlsh> describe keyspaces;

system_schema  system              system_traces   data
system_auth    system_distributed  bigdataproject

cqlsh> use bigdataproject;
cqlsh:bigdataproject> describe tables;

energy  weather

cqlsh:bigdataproject>
```



```
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_desc
ption | 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 | 291.14999 | 290.14999 |          sky
clear |          01d |        800 |        clear |       30 |          2
 2018-07-07 17:00:00.000000+0000 |   Seville |          0 |       26 |     1015 |       0 |       0 |       0 | 306.54001 | 307.14999 | 306.14999 |          sky
clear |          01d |        800 |        clear |      310 |          2
 2017-05-30 13:00:00.000000+0000 |   Seville |         20 |       73 |     1016 |       0 |       0 |       0 | 294.54001 | 295.14999 | 294.14999 |           fe
louds |          02d |        801 |       clouds |       20 |          0
 2016-11-15 10:00:00.000000+0000 |   Seville |          0 |       81 |     1024 |       0 |       0 |       0 |    290.06 | 300.14999 | 282.14999 |          sky
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
clear |          02n |        800 |        clear |        7 |          1
 2017-04-23 15:00:00.000000+0000 |   Seville |          0 |       43 |     1016 |       0 |       0 |       0 |    300.22 | 309.14999 | 294.14999 |          sky
clear |          01d |        800 |        clear |      156 |          1
 2018-08-27 00:00:00.000000+0000 |   Seville |          0 |       33 |     1011 |       0 |       0 |       0 | 305.14999 | 305.14999 | 305.14999 |          sky
clear |          01n |        800 |        clear |      230 |          3
 2016-03-05 18:00:00.000000+0000 |   Seville |          0 |       36 |     1014 |       0 |       0 |       0 |    292.44 | 298.14999 | 289.14999 |          sky
clear |          01d |        800 |        clear |      320 |          5
 2018-10-02 07:00:00.000000+0000 |   Seville |         40 |       83 |     1017 |       0 |       0 |       0 |    292.94 | 294.14999 | 292.14999 |    scattere
louds |          03n |        802 |       clouds |       30 |          2
 2017-08-30 13:00:00.000000+0000 |   Seville |         20 |       77 |     1016 |       0 |       0 |       0 | 293.54001 | 294.14999 | 293.14999 |           fe
louds |          02d |        801 |       clouds |        0 |          0
 2015-10-05 09:00:00.000000+0000 |   Seville |         20 |       69 |     1019 |       0 |       0 |       0 |    298.25 | 300.14999 | 296.14999 |
 mist |          50n |        701 |         mist |      200 |          3
 2016-07-06 06:00:00.000000+0000 |   Seville |          0 |       78 |     1017 |       0 |       0 |       0 | 298.82999 | 306.14999 | 294.14999 |          sky
clear |          01n |        800 |        clear |      210 |          3
 2017-03-06 16:00:00.000000+0000 |   Seville |          0 |       56 |     1027 |       0 |       0 |       0 | 294.64999 | 299.14999 | 291.14999 |          sky
clear |          01d |        800 |        clear |        0 |          0
```

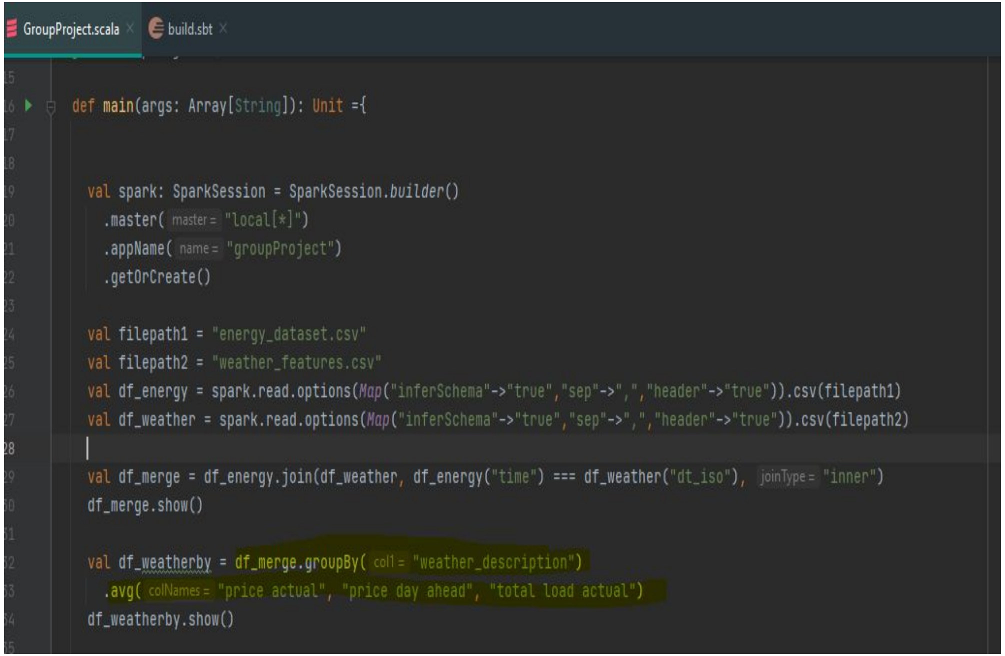
=> Created two separate table in cassandra, and perform query for each tables



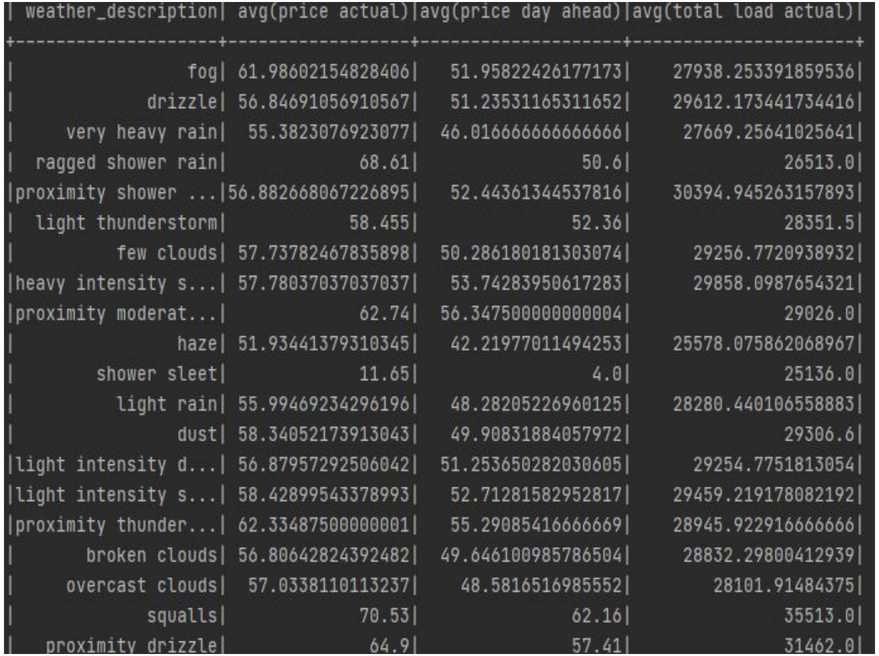
```
cqlsh:bigdataproject> select time, total_load_forecast, total_load_actual, price_day_ahead, price_actual fro
m energy  where price_actual >70 LIMIT 10 ALLOW FILTERING;

 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
```

=> From This query, I only display the data where actual price is over 70

SPARKSQL(Intellij)



```scala
  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|  46.016666666666666|     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|
```

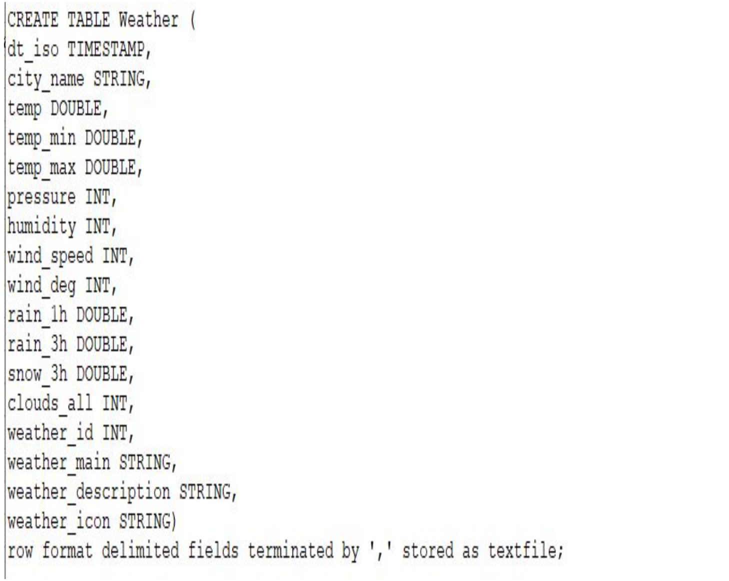
=> From This query, I group by weather description and display price information to show the correlation between them

Bill Yerkes

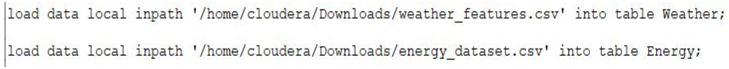
- Utilized Hive (Hadoop and MapReduce) to analyse the Energy Data
  - Created Tables
  - Imported Data
  - Ran Queries (MapReduce) to get metrics

- Technologies not utilized/required (May used them in the future)
  - Cassandra (NoSQL DB: do not foresee using.)
  - Sqoop (Data Migration: probably will use to move to SQL DB for reports.)
  - Solr (Search Engine: do not foresee using.)

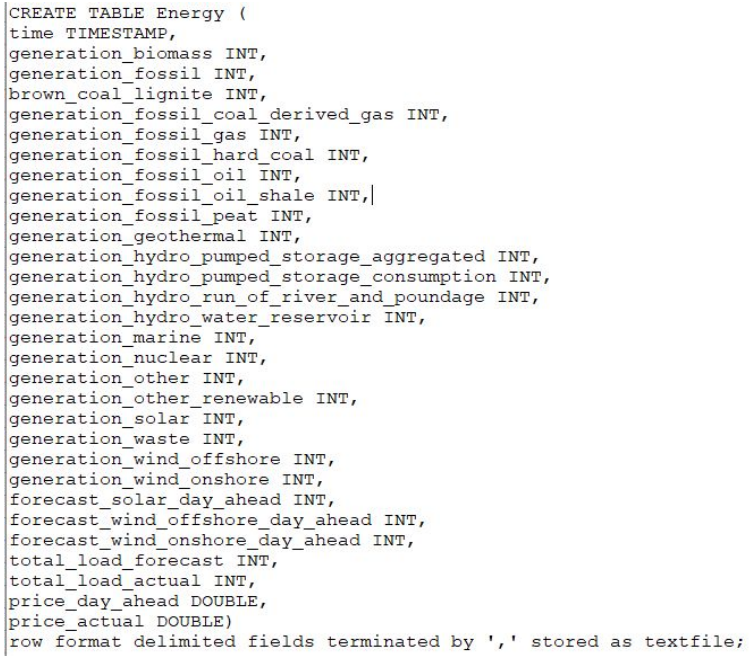
Bill Yerkes



```
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;
```



```
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;
```

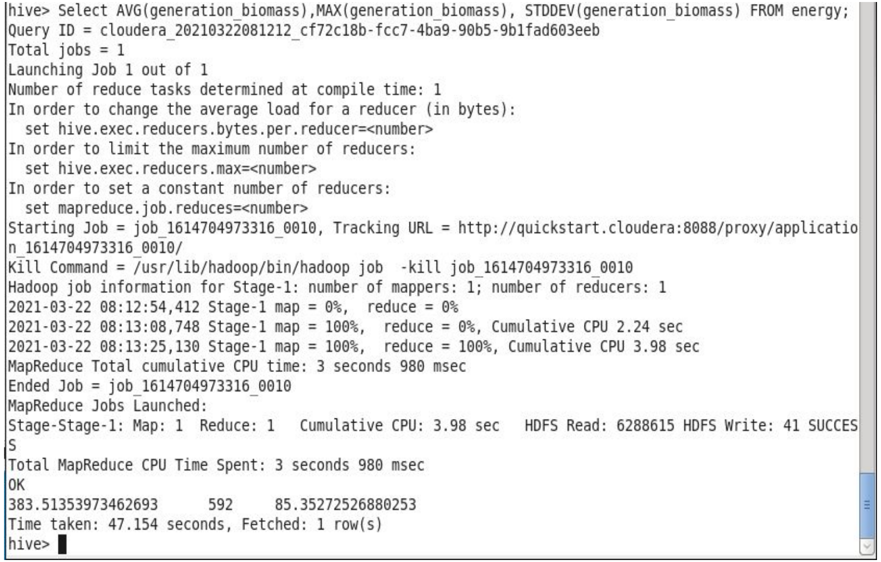


```
CREATE TABLE Energy (
time TIMESTAMP,
generation_biomass INT,
generation_fossil INT,
brown_coal_lignite INT,
generation_fossil_coal_derived_gas INT,
generation_fossil_gas INT,
generation_fossil_hard_coal INT,
generation_fossil_oil INT,
generation_fossil_oil_shale INT,
generation_fossil_peat INT,
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;
```

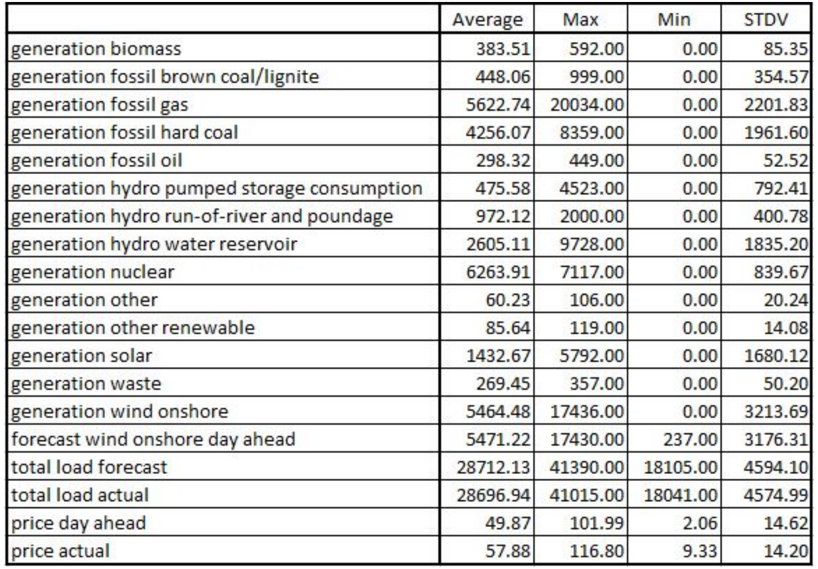
Bill Yerkes

Generate Metrics via Hive (MapReduce)

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



```
hive> Select AVG(generation_biomass),MAX(generation_biomass), STDDEV(generation_biomass) FROM energy;
Query ID = cloudera_20210322081212_cf72c18b-fcc7-4ba9-90b5-9b1fad603eeb
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1614704973316_0010, Tracking URL = http://quickstart.cloudera:8088/proxy/applicatio
n_1614704973316_0010/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1614704973316_0010
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2021-03-22 08:12:54,412 Stage-1 map = 0%,  reduce = 0%
2021-03-22 08:13:08,748 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.24 sec
2021-03-22 08:13:25,130 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.98 sec
MapReduce Total cumulative CPU time: 3 seconds 980 msec
Ended Job = job_1614704973316_0010
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.98 sec   HDFS Read: 6288615 HDFS Write: 41 SUCCES
S
Total MapReduce CPU Time Spent: 3 seconds 980 msec
OK
383.51353973462693      592     85.35272526880253
Time taken: 47.154 seconds, Fetched: 1 row(s)
hive> 
```



	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

Next Steps

- Group will pool information about analysis of Data.
- Investigate migration of current functions from Hive to Spark.
- Investigate future class technologies on how to perform predictions and generate visualizations of data.
- Join datasets based on time column and look for the relationships between the two

Q & A

THANK YOU