Exercise 3

SparkSQL

Prior Knowledge
Unix Command Line Shell
Simple Python
Apache Spark in Jupyter (from previous exercise)

Learning Objectives
SparkSQL
Reading CSV files in Spark

Software Requirements
(see separate document for installation of these)

- Apache Spark
- Jupyter

1. Let's create a new directory for our work:

   ```
   cd ~
   mkdir sql
   cd sql
   ```

2. We need to download some data to work with:
   ```
   wget https://freo.me/winddata15 -O wind2015.zip
   ```

   You should see something like:

   ```
   wind2015.zip        100%[===================>]   4.97M  4.35MB/s    in 1.1s
   ```

3. Now unzip the files:
   ```
   unzip wind2015.zip
   ```

   You should see :
   ```
   Archive:  wind2015.zip
     inflating: SF04.csv
     inflating: SF15.csv
     inflating: SF17.csv
     inflating: SF18.csv
     inflating: SF36.csv
     inflating: SF37.csv
   ```


4. Now start Jupyter:

   ```
   jupyter notebook
   ```

5. Give the notebook a useful name

6. Now create a cell with our line to configure tab completion:

   ```
   %config IPCompleter.greedy=True
   ```

7. Run that cell.

8. Now, create a new cell which will have our main code in it.

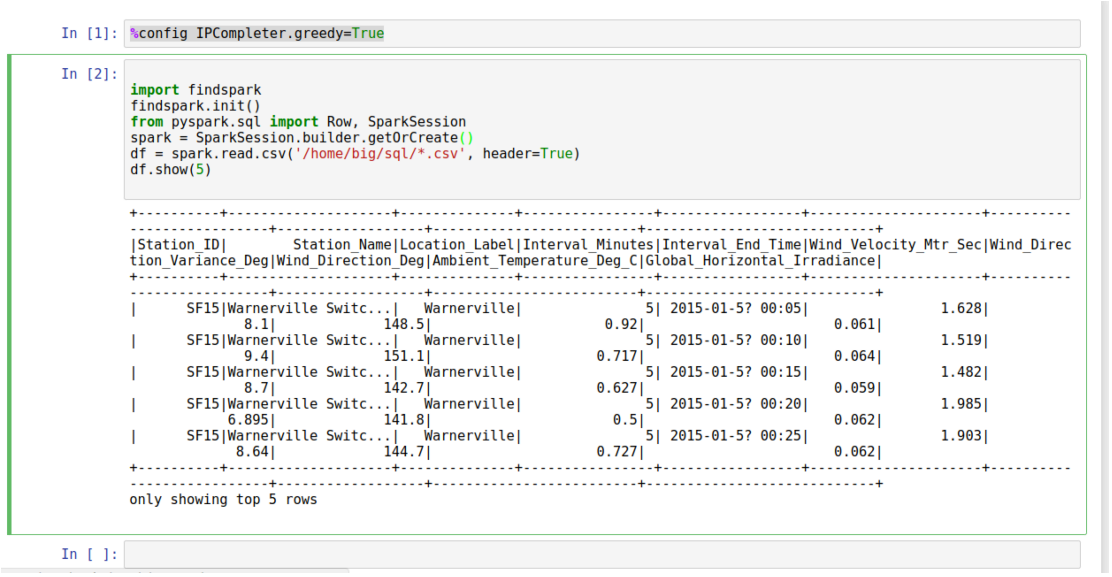
   Type the following into the new cell (you don't need to type the comments):

   ```
   # these two lines make spark work in Jupyter
   import findspark
   findspark.init()
   # This tells us that we are working with Spark DataFrames
   from pyspark.sql import Row, SparkSession
   spark = SparkSession.builder.getOrCreate()

   # read the wind data from CSV files
   df = spark.read.csv('/home/big/sql/*.csv', header=True)
   # show the top 5 rows
   df.show(5)
   ```

9. The df object we have is not an RDD, but instead a DataFrame. This is basically a SQL motivated construct that is similar to a Pandas or R dataframe (but not exactly the same!)

10. Run the cell. You should see:



```
In [1]: %config IPCompleter.greedy=True

In [2]: import findspark
        findspark.init()
        from pyspark.sql import Row, SparkSession
        spark = SparkSession.builder.getOrCreate()
        df = spark.read.csv('/home/big/sql/*.csv', header=True)
        df.show(5)
```

```
+----------+--------------------+--------------+----------------+-----------------+--------------------+----------
----------------+-----------------+-------------------------+--------------------------+
|Station_ID|        Station_Name|Location_Label|Interval_Minutes|Interval_End_Time|Wind_Velocity_Mtr_Sec|Wind_Direc
tion_Variance_Deg|Wind_Direction_Deg|Ambient_Temperature_Deg_C|Global_Horizontal_Irradiance|
+----------+--------------------+--------------+----------------+-----------------+--------------------+----------
----------------+-----------------+-------------------------+--------------------------+
|      SF15|Warnerville Switc...|   Warnerville|               5| 2015-01-5? 00:05|                1.628|
               8.1|             148.5|                     0.92|                     0.061|
|      SF15|Warnerville Switc...|   Warnerville|               5| 2015-01-5? 00:10|                1.519|
               9.4|             151.1|                    0.717|                     0.064|
|      SF15|Warnerville Switc...|   Warnerville|               5| 2015-01-5? 00:15|                1.482|
               8.7|             142.7|                    0.627|                     0.059|
|      SF15|Warnerville Switc...|   Warnerville|               5| 2015-01-5? 00:20|                1.985|
             6.895|             141.8|                      0.5|                     0.062|
|      SF15|Warnerville Switc...|   Warnerville|               5| 2015-01-5? 00:25|                1.903|
              8.64|             144.7|                    0.727|                     0.062|
+----------+--------------------+--------------+----------------+-----------------+--------------------+----------
----------------+-----------------+-------------------------+--------------------------+
only showing top 5 rows

In [ ]:
```

11. This is data from weather stations in San Francisco showing the wind speed, direction and temperature throughout 2015.

12. Before we do this as SQL, we are going to look at the data using the same map/reduce model we used previously. To do this, we will convert the he DataFrame into an RDD, allowing us to do functional programming on it (map/reduce/etc)

    Note that this doesn't copy the data, but just exposes the rdd which is already hiding inside the dataframe.

    ```
    winds = df.rdd
    ```

13. Let's do the normal step of mapping the data into a simple <K,V> pair. Each column in the row can be accessed by the syntax e.g. row.Station_ID

    We can therefore map our RDD with the following:

    ```
    mapped = winds.map(lambda s: (s.Station_ID, s.Wind_Velocity_Mtr_Sec))
    ```

14. We can simply calculate the maximum values with this reducer:

    ```
    maxes = mapped.reduceByKey(lambda a, b: a if (a>b) else b)
    ```

15. And once again collect / print:

    ```
    for (k,v) in maxes.collect(): print k,v
    ```

16. Comment out the df.show(5) line.

17. Now run the cell again. You should see:

```
import findspark
findspark.init()
from pyspark.sql import Row, SparkSession
spark = SparkSession.builder.getOrCreate()
df = spark.read.csv('/home/big/sql/*.csv', header=True)
# df.show(5)
winds = df.rdd
mapped = winds.map(lambda s: (s.Station_ID, s.Wind_Velocity_Mtr_Sec))
maxes = mapped.reduceByKey(lambda a, b: a if (a>b) else b)
for (k,v) in maxes.collect(): print k,v
```

```
SF36 9.71
SF18 9.85
SF37 7.079
SF04 9.92
SF15 7.92
SF17 5.767
```

18. You can also turn the response of a collect into a Python Map, which is handy. Try this:

```
print maxes.collectAsMap()
print maxes.collectAsMap()['SF04']
```

PART B – Using SQL

19. There is an easier way to do all this if you are willing to write some SQL. Comment out the lines from `winds = df.rdd` down to the end of the cell. Hint: Select the lines and then hit **Ctrl-/**

20. First we need to give our DataFrame a table name:
`df.registerTempTable('wind')`

21. Now we can use a simple SQL statement against our data.

```
spark.sql("SELECT Station_ID, avg(Wind_Velocity_Mtr_Sec) as
avg,max(Wind_Velocity_Mtr_Sec) as max from wind group by
Station_ID").show()
```


22. Now run the cell and you should see something like:

```
+----------+------------------+-----+
|Station_ID|               avg|  max|
+----------+------------------+-----+
|      SF37| 2.260403505500663|7.079|
|      SF15|1.8214145677504483| 7.92|
|      SF04| 2.300981748124102| 9.92|
|      SF17|0.5183500253485376|5.767|
|      SF18|2.2202234391695437| 9.85|
|      SF36| 2.464172530911313| 9.71|
+----------+------------------+-----+
```

23. One thing you might like is that you can convert from a Spark Dataframe to a Pandas dataframe just by calling **toPandas**()
    Note that when you do this, you are collecting the results back from a cluster to a single server (the master).

24. Recap. We have:
    a. Used Spark to read in CSV files
    b. Explored Map/Reduce on those CSV files
    c. Used SQL to query the data.
25. We are going to make this into a standalone program now. Copy the python code and paste into a file called wind.py

    You can use Atom, PyCharm, nano or some other editor.

26. It should look like this:

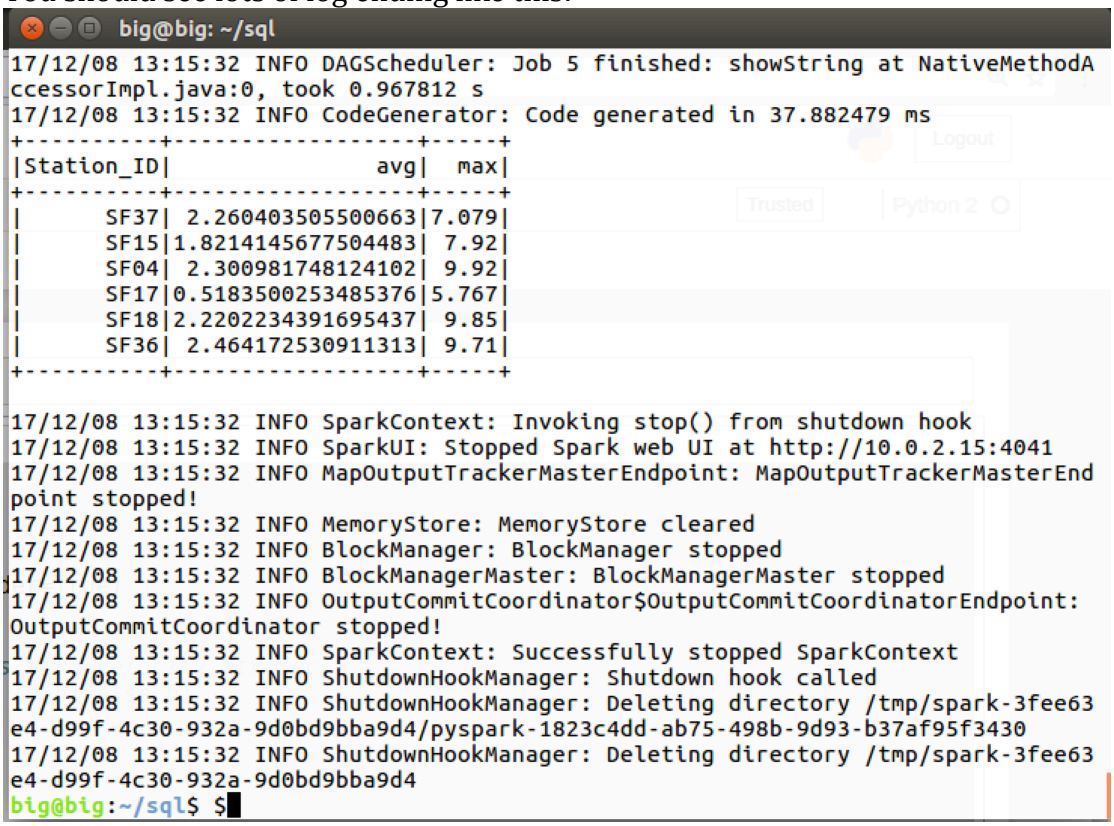
```
import findspark
findspark.init()
from pyspark.sql import Row, SparkSession
spark = SparkSession.builder.getOrCreate()
df = spark.read.csv('/home/big/sql/*.csv', header=True)

df.registerTempTable('wind')
spark.sql("SELECT Station_ID, avg(Wind_Velocity_Mtr_Sec) as avg,max(Wind_
```

27. Try running it as a standalone program:

```
~/spark/bin/spark-submit wind.py
```


28. You should see lots of log ending like this:



We are going to use this in the next exercise. That's all for now.

Congratulations, this lab is complete.