**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

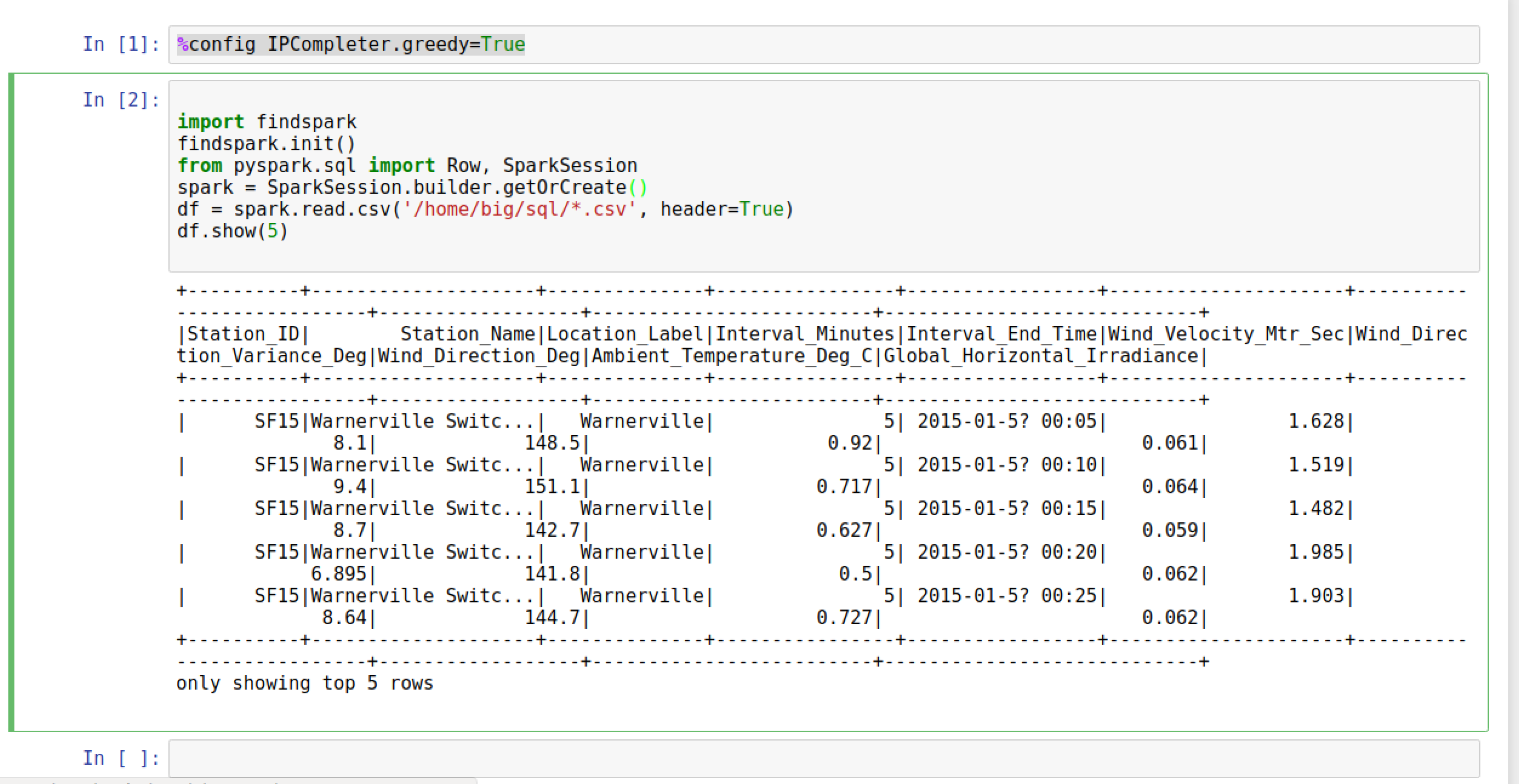
1. Now start Jupyter:  
     
   jupyter notebook
2. Give the notebook a useful name
3. Now create a cell with our line to configure tab completion:

%config IPCompleter.greedy=True

1. Run that cell.
2. 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)

1. 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!)
2. Run the cell. You should see:



1. This is data from weather stations in San Francisco showing the wind speed, direction and temperature throughout 2015.
2. 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
3. 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))
4. We can simply calculate the maximum values with this reducer:  
     
   maxes = mapped.reduceByKey(lambda a, b: a if (a>b) else b)
5. And once again collect / print:  
     
   for (k,v) in maxes.collect(): print k,v
6. Comment out the df.show(5) line.
7. Now run the cell again. You should see:

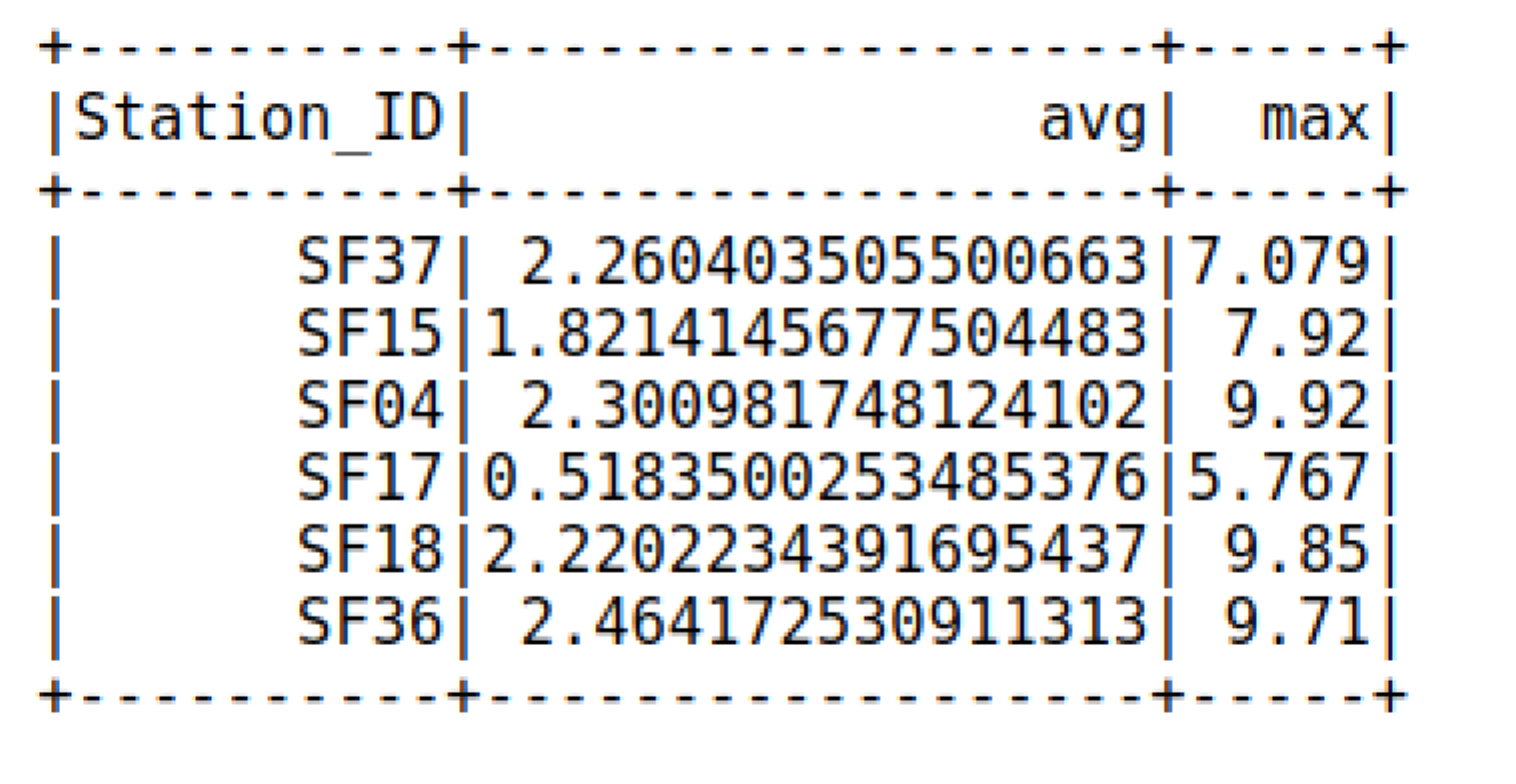
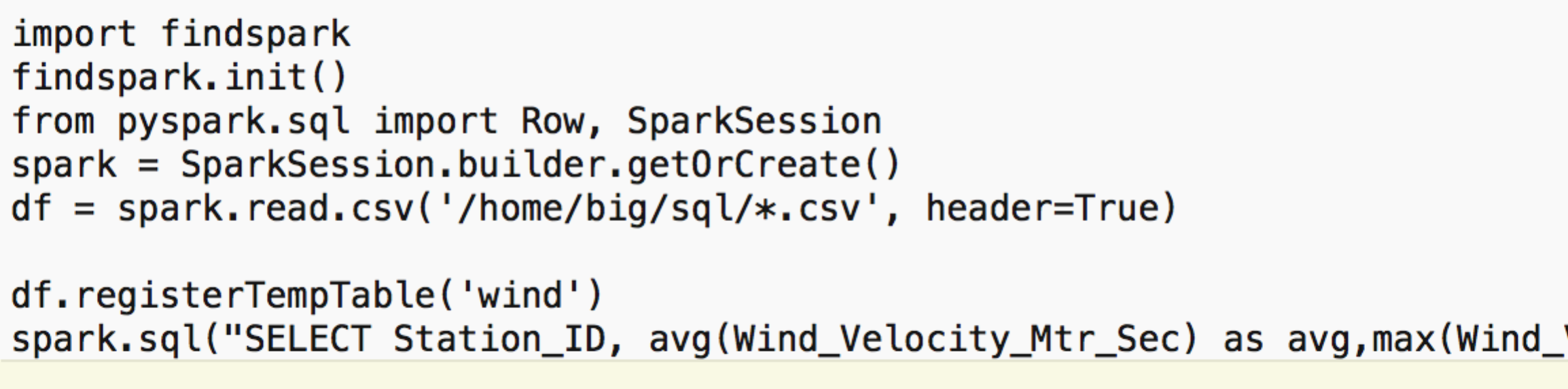
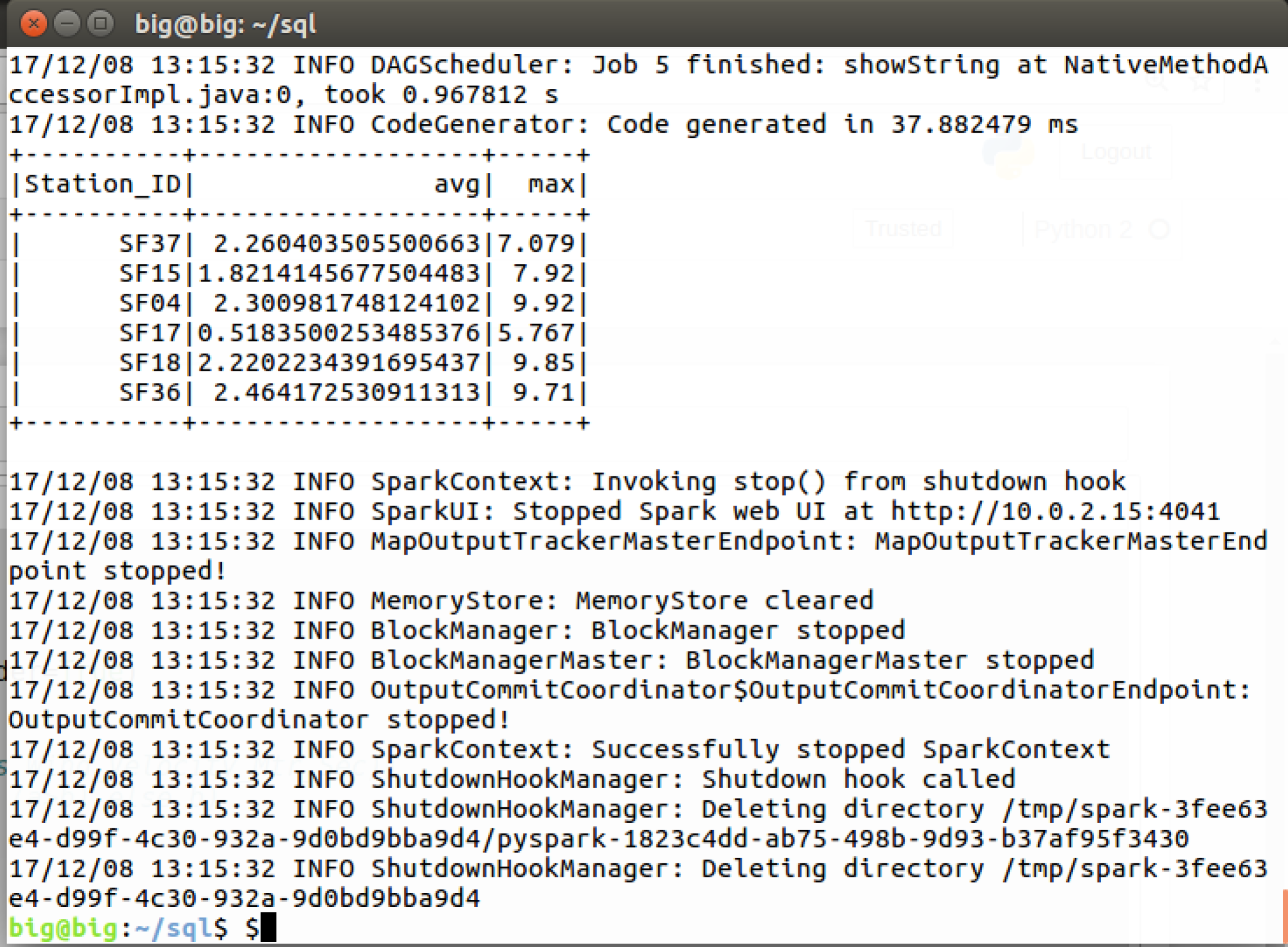


1. You can also turn the response of a collect into a Python Map, which is handy. Try this:  
     
   print maxes.collectAsMap()  
   print maxes.collectAsMap()['SF04']

**PART B – Using SQL**

1. 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-/**

1. First we need to give our DataFrame a table name:  
   df.registerTempTable('wind')
2. 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()
3. Now run the cell and you should see something like:  
     
   
4. 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).
5. Recap. We have:
   1. Used Spark to read in CSV files
   2. Explored Map/Reduce on those CSV files
   3. Used SQL to query the data.
6. 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.
7. It should look like this:  
   
8. Try running it as a standalone program:  
     
   ~/spark/bin/spark-submit wind.py
9. 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.