Assignment 3: Hadoop, Spark and Flink

By

Rachael Joan Dias

0651897

Addressed to,

Professor Brian Srivastava

Trent University

AMOD-5410H-A-2019GW-PTBO Big Data

**Objective:** Hadoop’s Mapreduce programming model was implemented on a dataset to answer a specific set of questions. Using Spark’s machine learning capabilities, linear regression was performed on a dataset to predict values of life expectancy. Flink’s real-time streaming was used to process text data.

1. **Data Gathering**

**Hadoop:** For the Hadoop part of the assignment I used the UCI Heart Disease dataset from Kaggle. The data consisted of 14attributes, there were 303 records totally. [1]

**Spark:** I used the (WHO) Life Expectancy dataset from Kaggle and applied linear regression to predict the average life expectancy based on a few factors. [2]

**Flink:** I created a text file implemented some of the DataStream and DataSet API functions on it.

1. **Hadoop**

**Note:** Since Hadoop was not working on my system, I used my friend’s laptop to execute my code, Prof Sri has given us permission to do so.

Prior to executing MapReduce below commands were run to start Hadoop, create input and output directories

**Command 1:** Start-all.cmd

This will start Hadoop the namenode, datanode, resource manager and node manager

**Command 2:** hadoop fs -mkdir /input

This command will create an input directory by the name of input

**Command 3:** hadoop fs -put C:\Users\17059\Desktop\BigData\heart.txt /input

Once the input directory is created, we need to place the file in the input directory, the above command will put the file in the input directory

**Command 4:** hadoop fs -mkdir /output

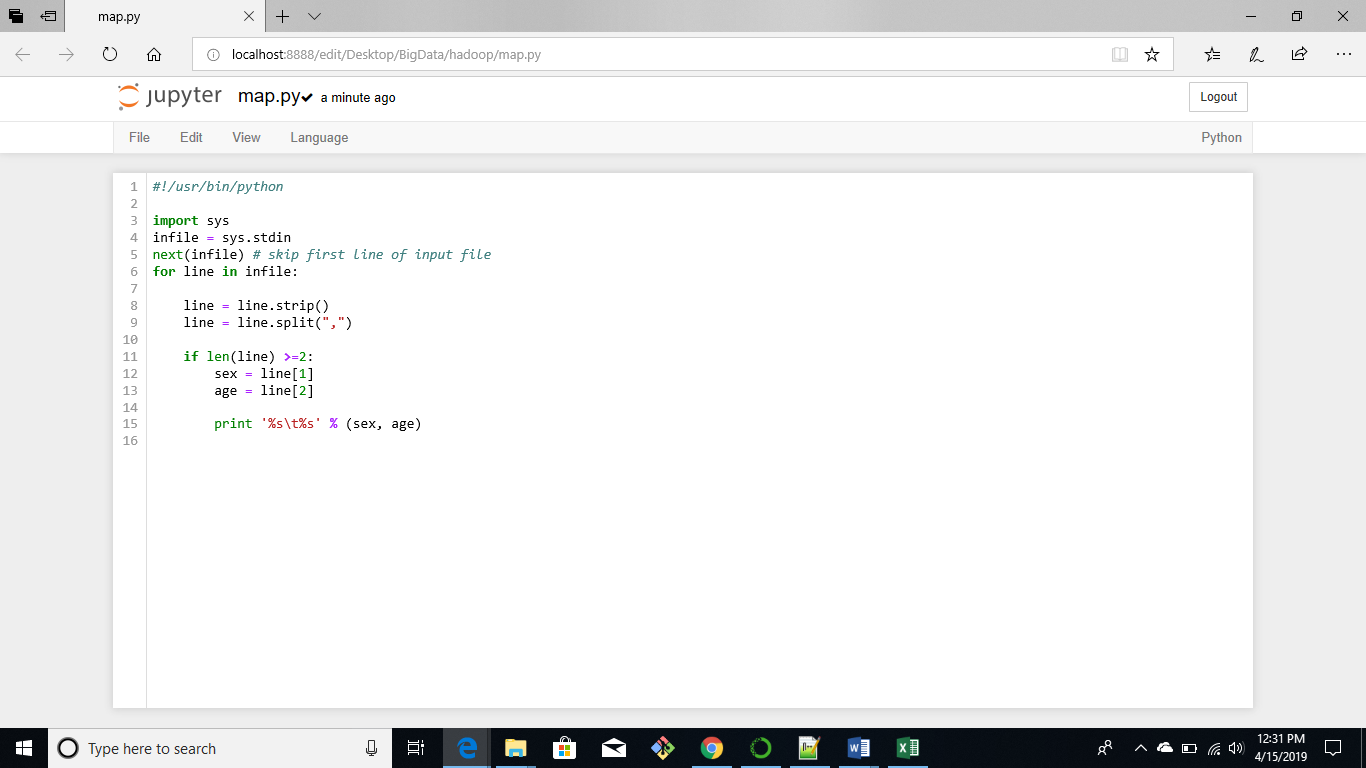
Now, that we have the file in the input directory, the next step is to create an output directory where we can see the output of the MapReduce program

I used the “sex” and “age” columns from UCI Heart Disease dataset. Using, a mapper and reducer program I was able to answer 3 specific questions related to the gender and age.

1. **Average age among males and females in the dataset**

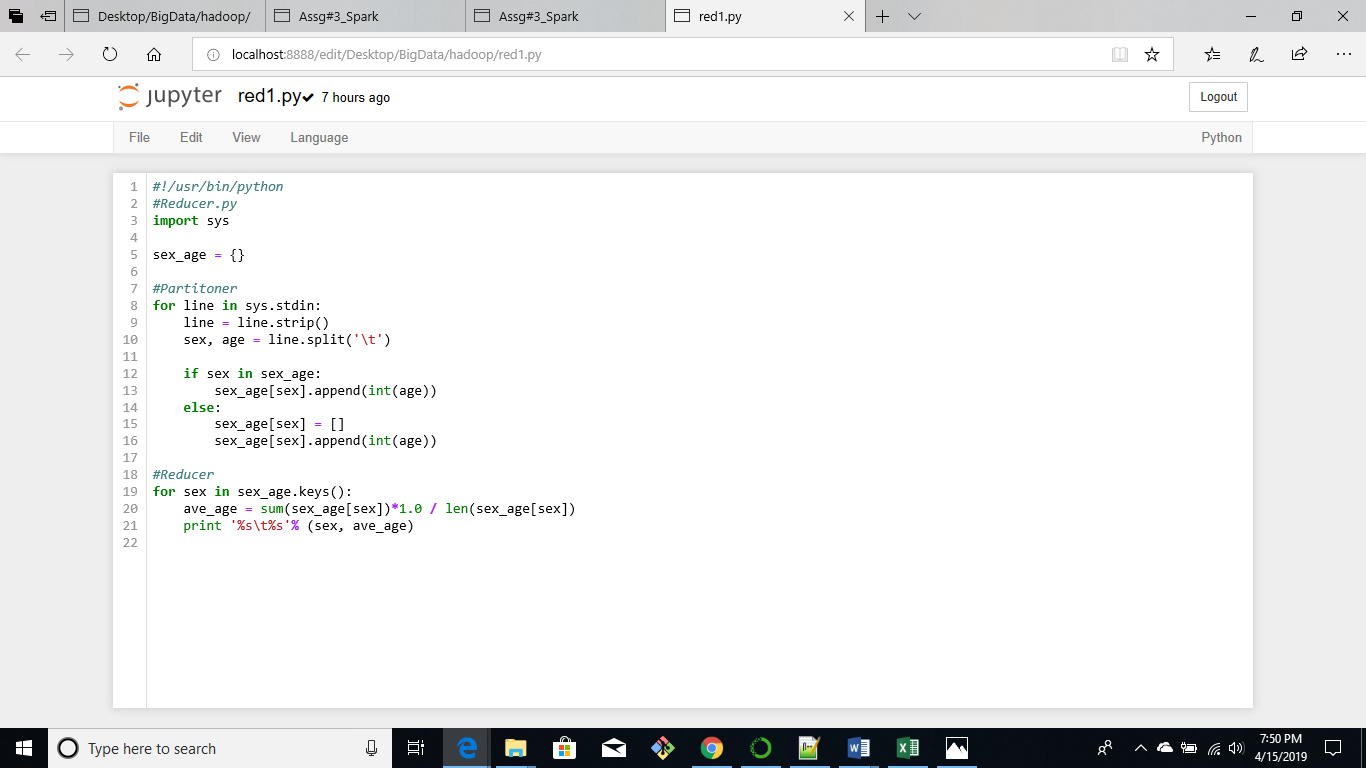
I used next(infile) to skip the first line of the file since this is the header.

Below is a screenshot of the mapper script. The mapper script reads from stdin and prints to stdout line by line, since we only need the “sex” and “age” attributes we can select them by indexing line[1] and line[2]. These are printed out as key-value pairs

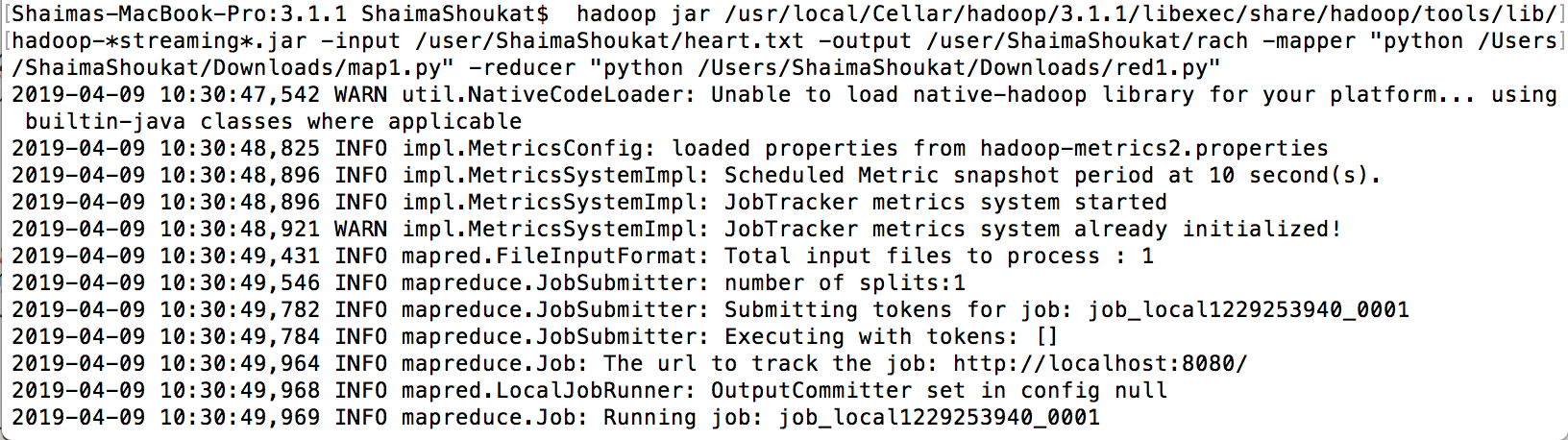


An empty dictionary is initialized by specifying sex\_age{}, which will store sex and age as key-value pairs.

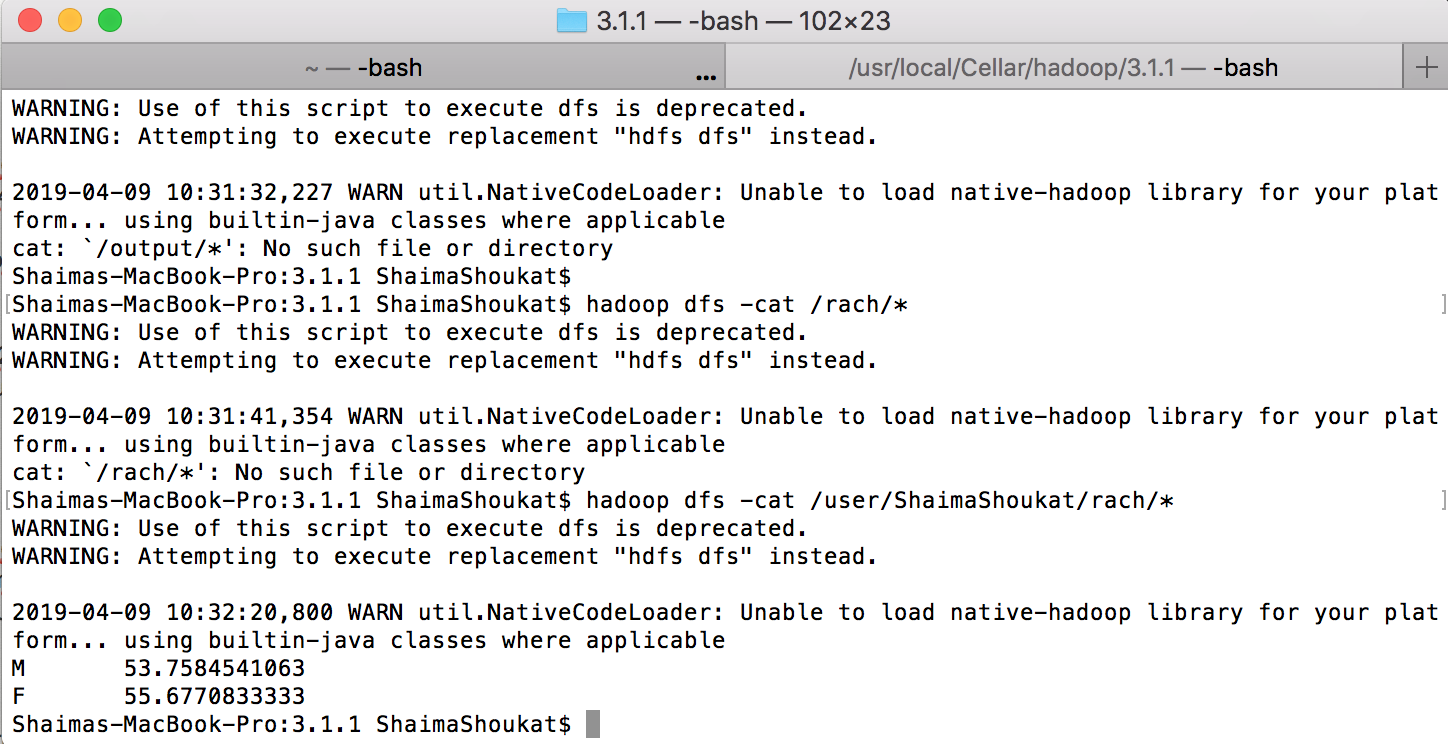
The reducer script will sum all the ages present in the dictionary and divide by the total number of elements present in the dictionary. This will return the average age



Below Hadoop streaming command was executed to run the MapReduce job



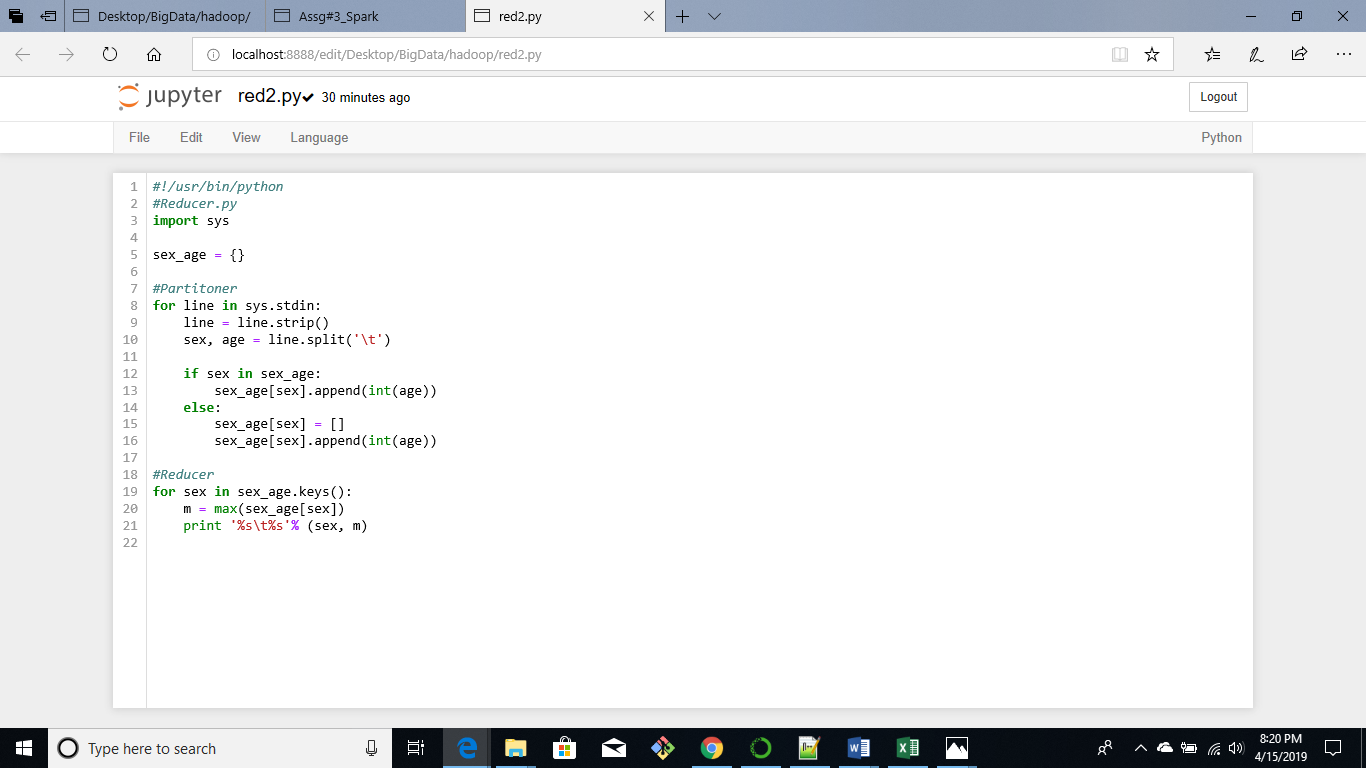
The MapReduce job gives the following output, the average age among males is 53.75 years while the average age among females is 55.67 years.



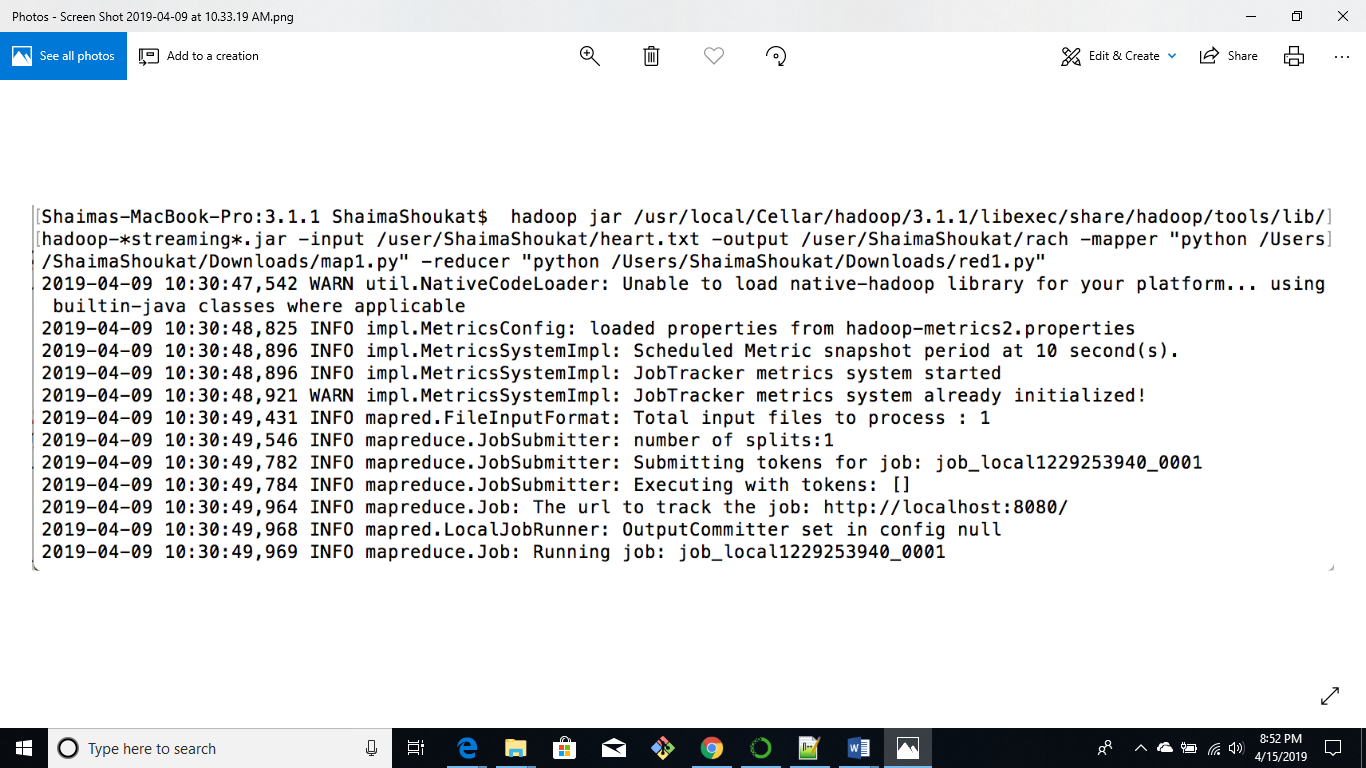
**Scripts:** map.py and red1.py

1. **Maximum age among males and females in the dataset**

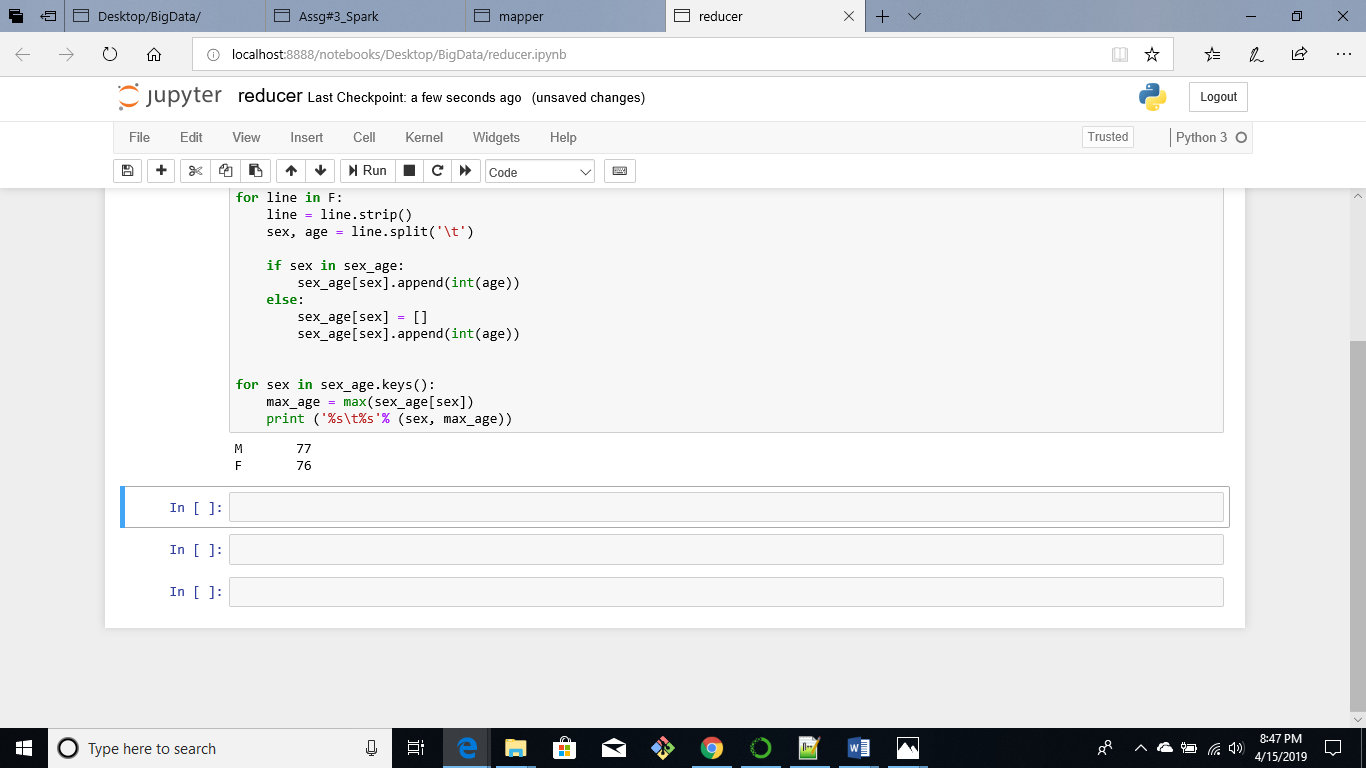
In a similar way, the maximum age among males and females can be found using the same mapper script and changing the reducer by replacing it with the max() function. The max() function will return the largest age among males and females in the dictionary.



Below Hadoop Streaming command was executed to run the MapReduce job



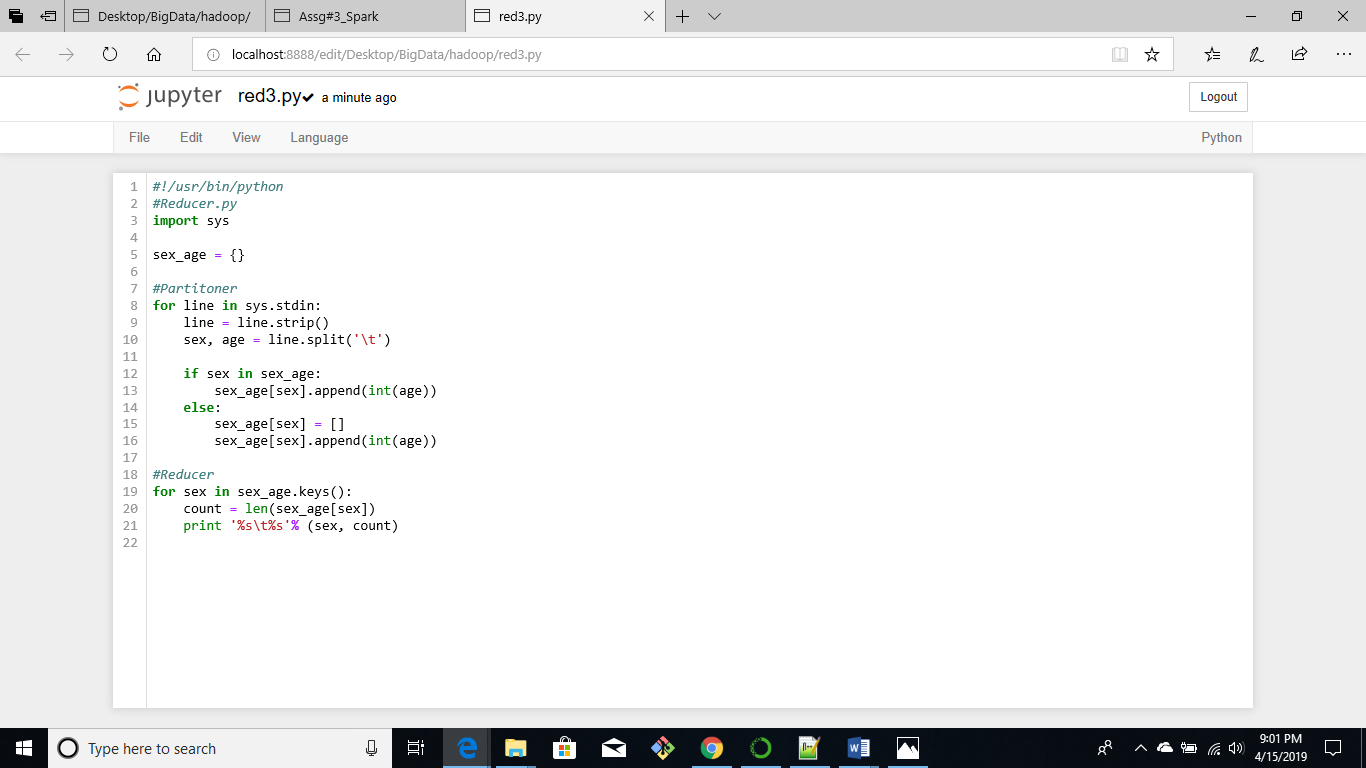
The maximum age among males and females is 77 and 76 years respectively



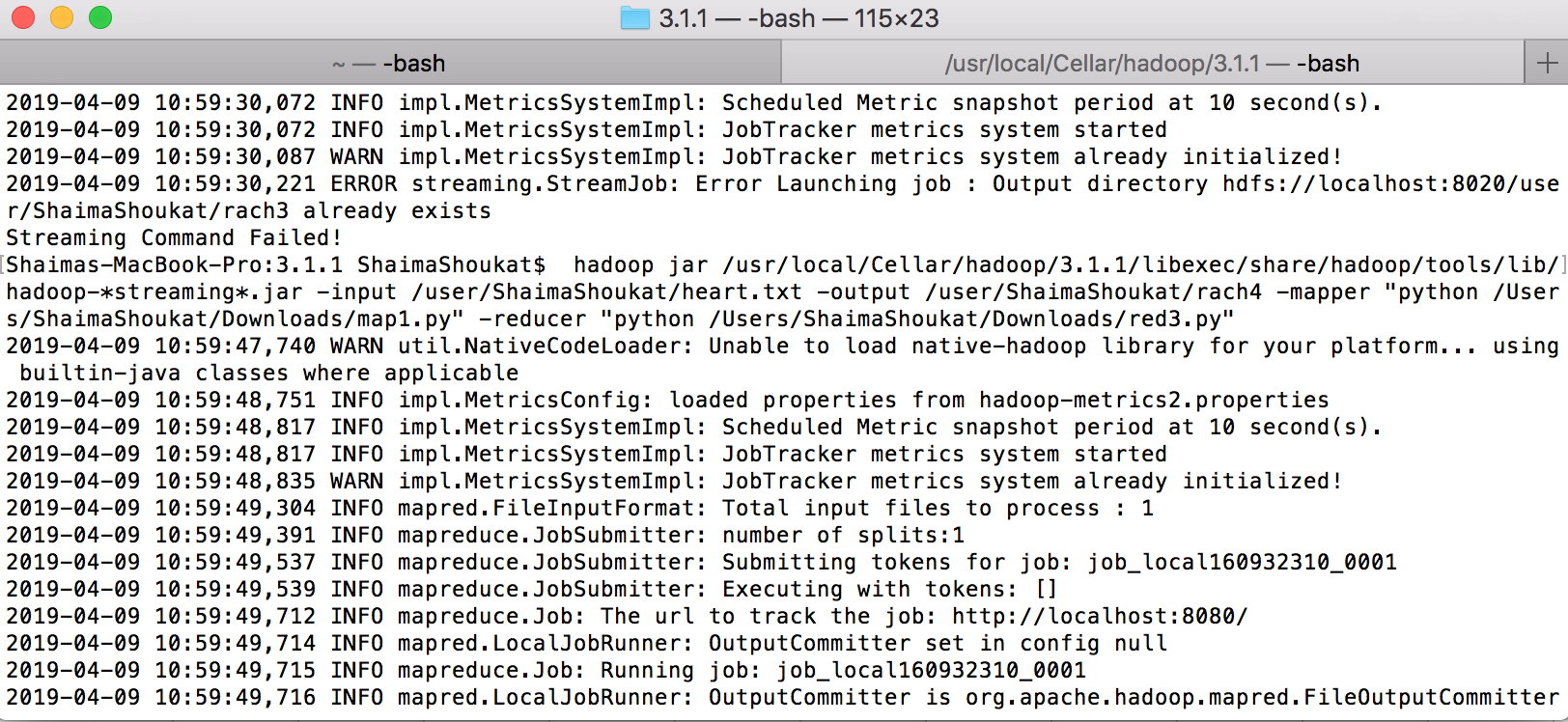
**Scripts:** map.py and red2.py

1. **Count of males and females in the dataset**

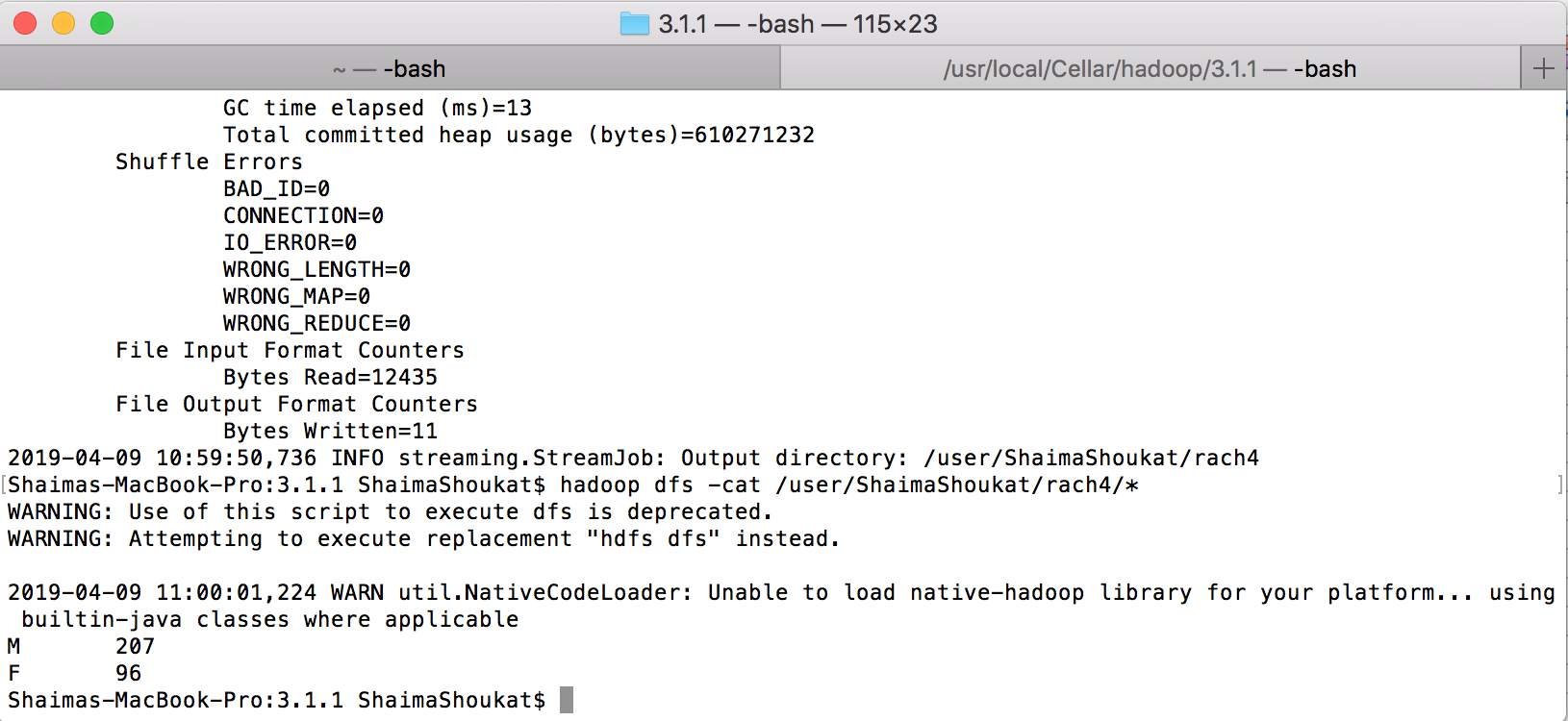
The reducer script is modified to count the total number of males and females in the dataset, by applying the len() function to the dictionary we can count the total number of males and females present



Command for MapReduce job



From the output below we can see that there are 207 and 96 females.



**Scripts:** map.py and red3.py

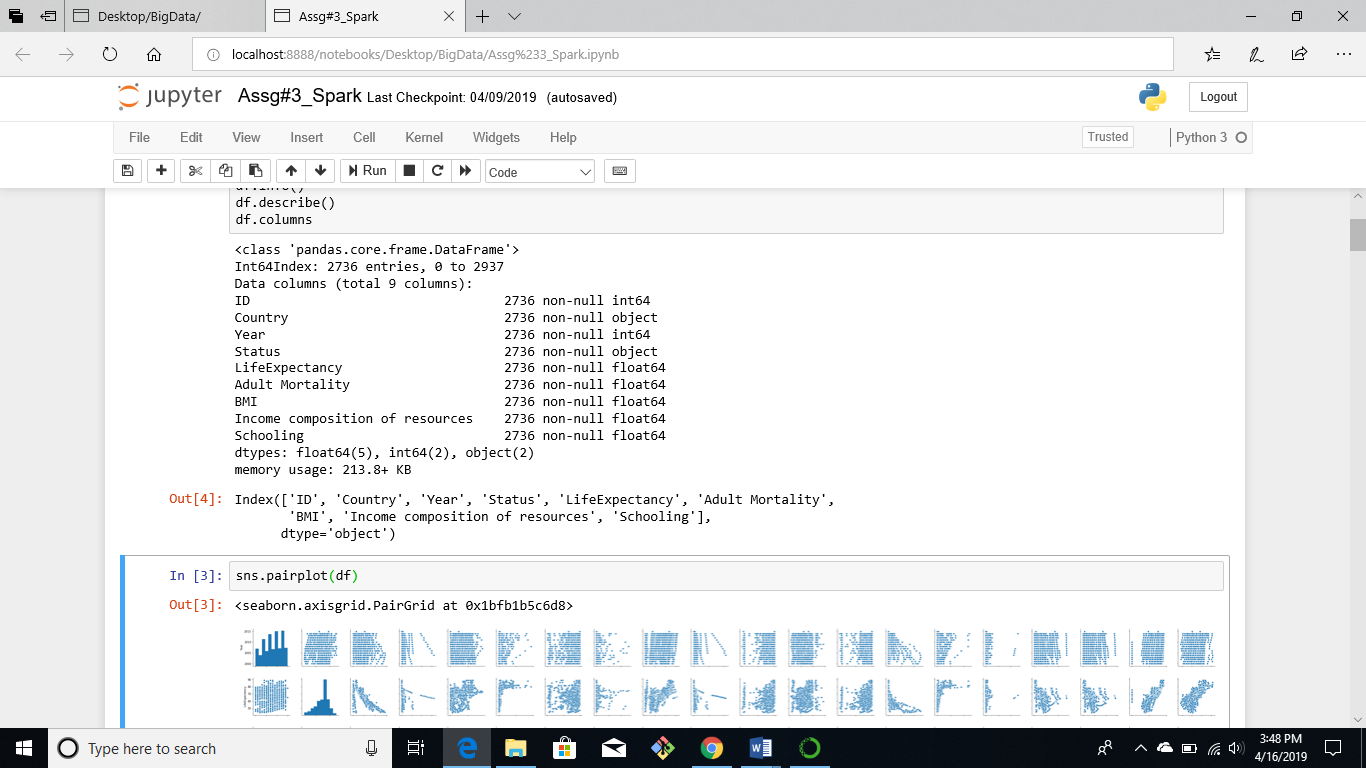
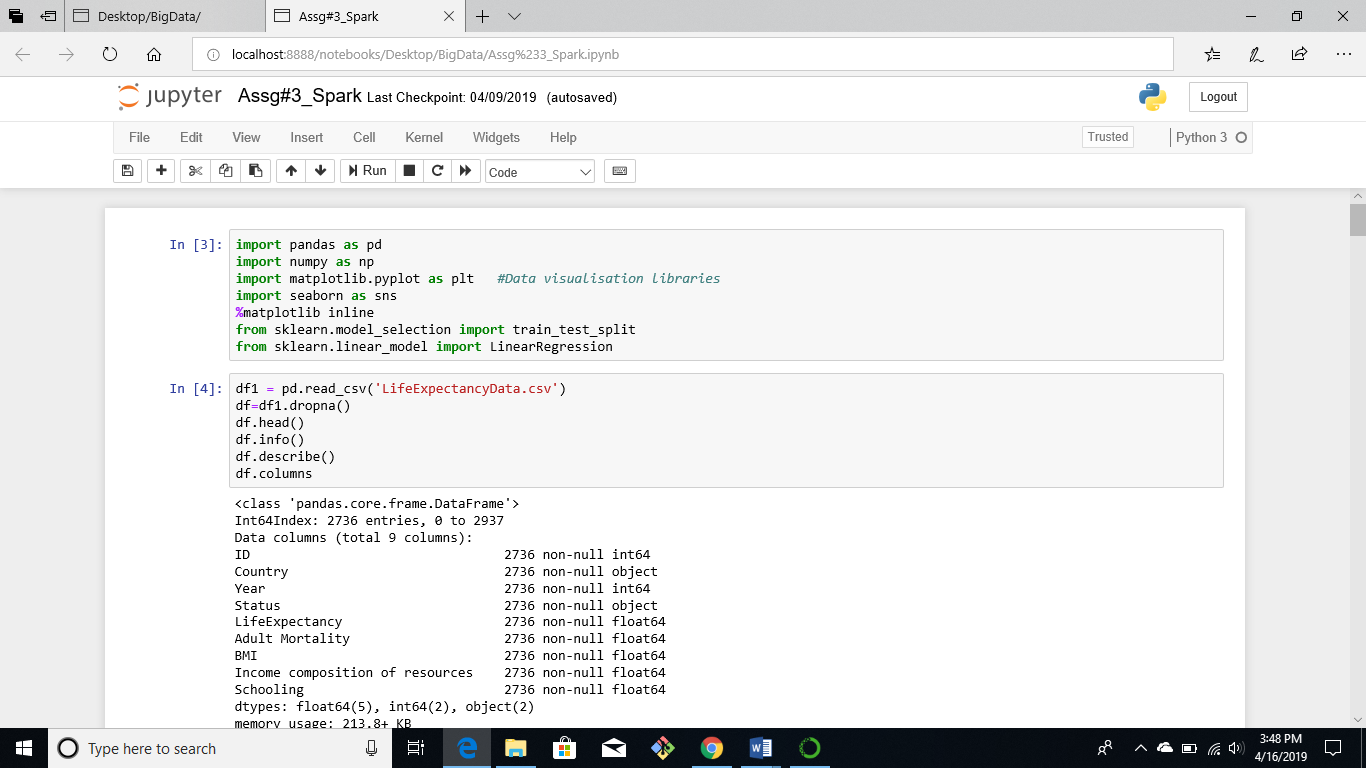
Hadoop is especially useful when performing computation on large datasets, because the MapReduce programming model stores and distributes the data across a cluster of computers.

1. **Spark**

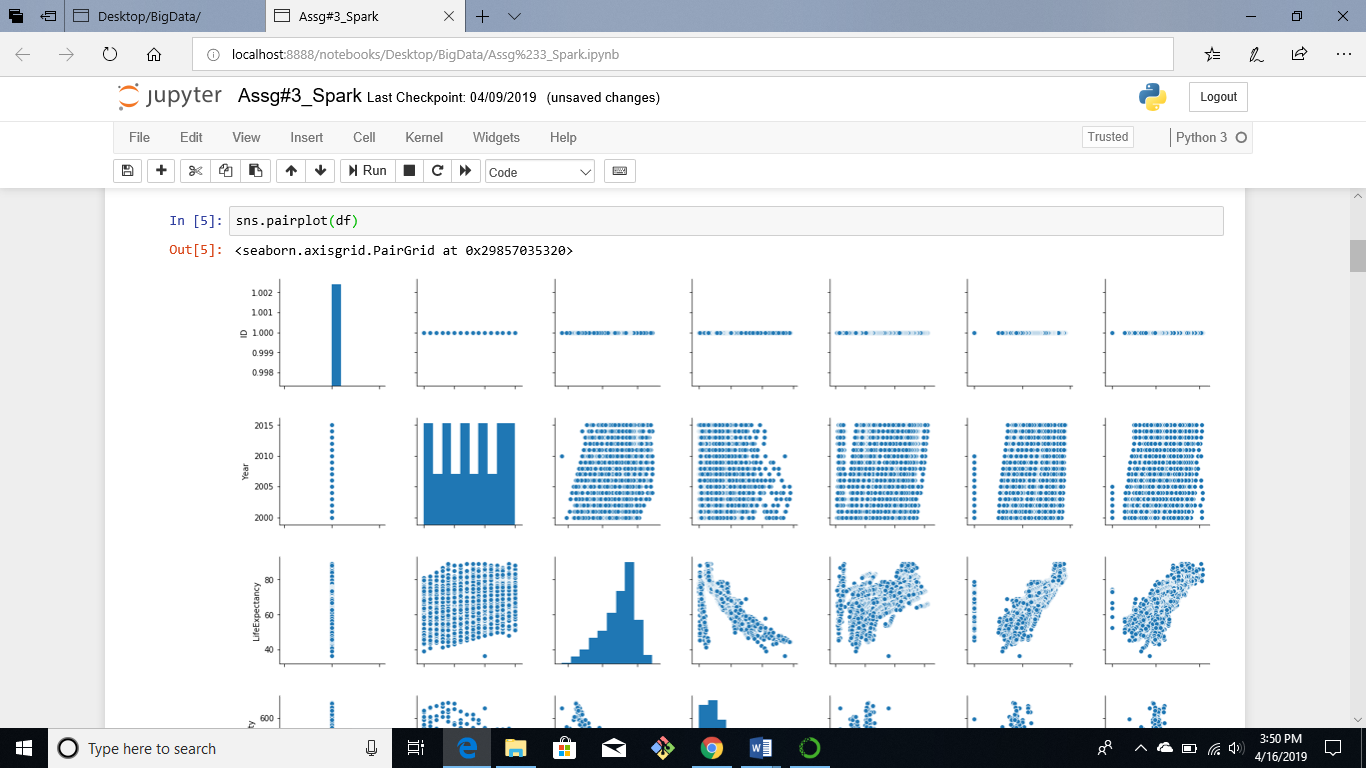
**Regression and Visualization:**

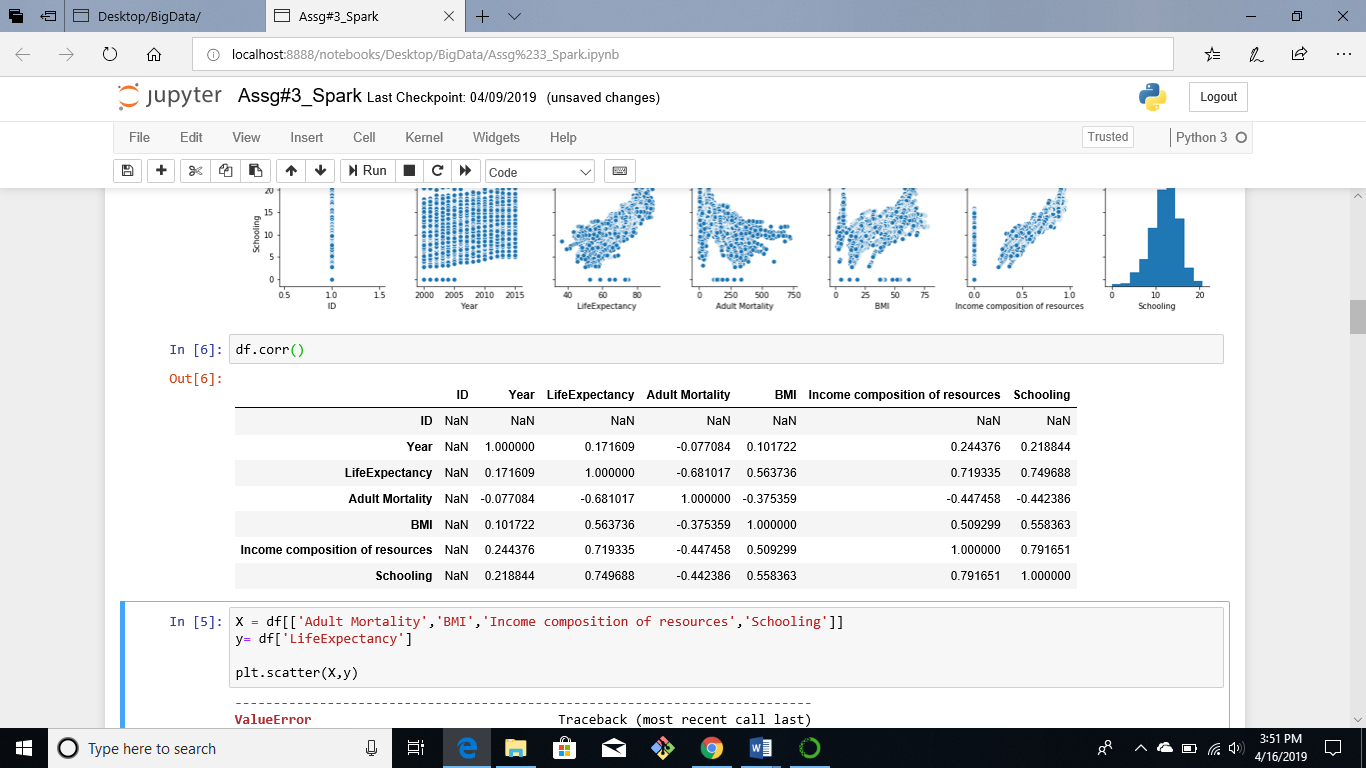
Linear regression predicts the value of dependent variable ‘Y’ based on variable ‘X’ assuming there is a strong correlation between ‘X’ and ‘Y’. The life expectancy dataset consists of the “LifeExpectancy” attribute which is the average number of years an individual is expected to live for different countries between 2000 and 2015. Using, linear regression variables such as adult mortality, BMI, income and schooling can be used to predict the life expectancy for different countries.

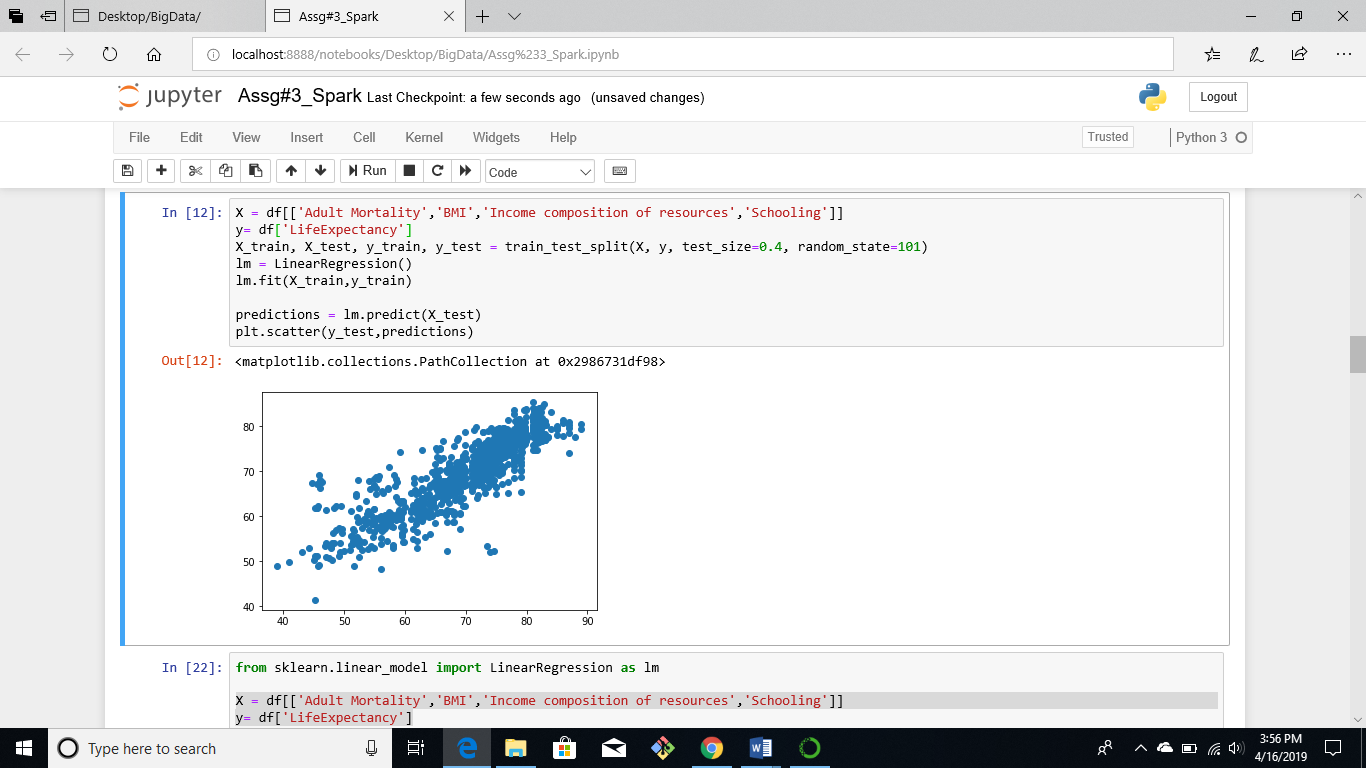
We start our analysis by importing the libraries below to visualize the relationships and then, build a linear model to predict the life expectancy of an individual. First, we read the .csv file using the pandas function read\_csv(). Then, since we want to simplify our analysis we drop all rows containing Null values using the dropna() function.



From, the correlation plots we observe that there is a strong linear relationship between the “LifeExpectancy” variable and Adult mortality, BMI, income and schooling. Therefore, we can use these variables to build a linear regression model.

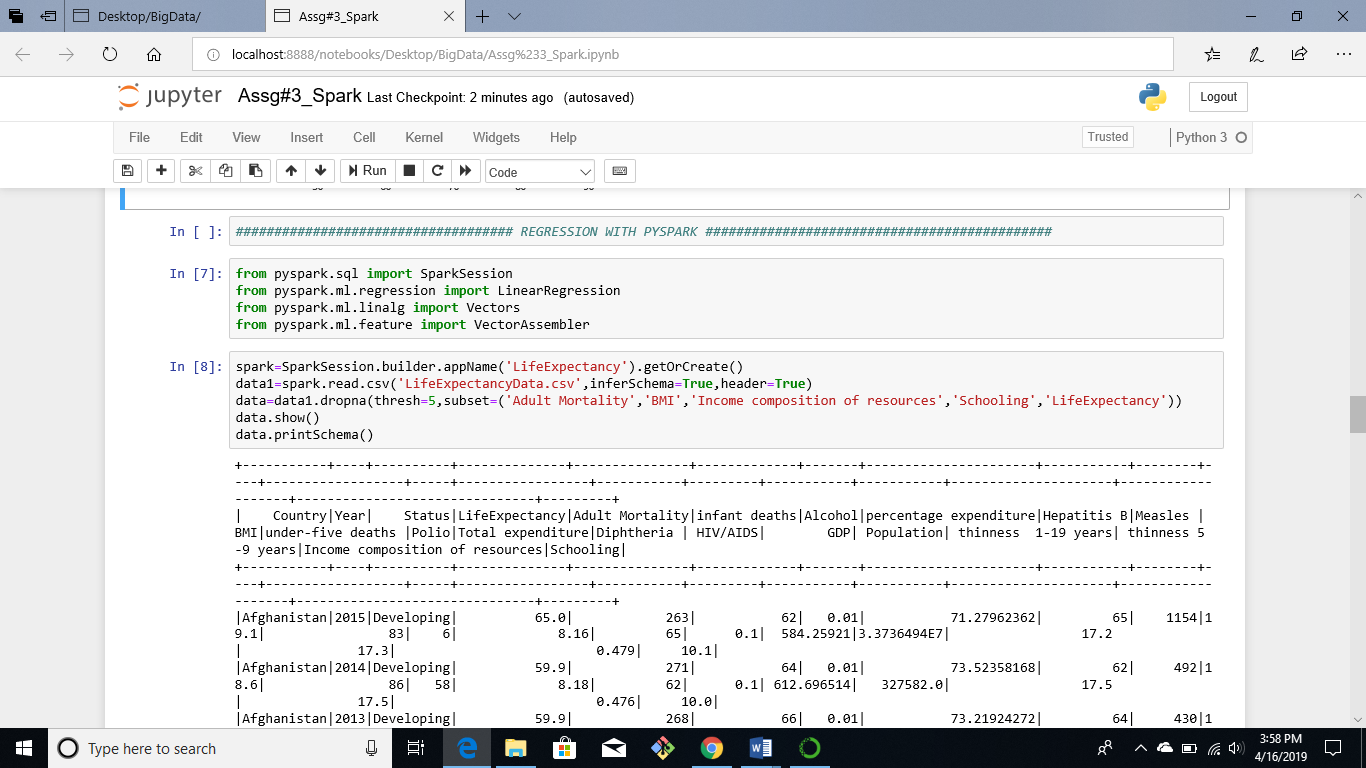




First, we define the outcome variable “y” and the independent variables “X”, then we split the dataset into test and train. Using the LinearRegression() function a linear model is fit with the data which can be used to make predictions. 

A scatter of the actual value and predictions based on the data.

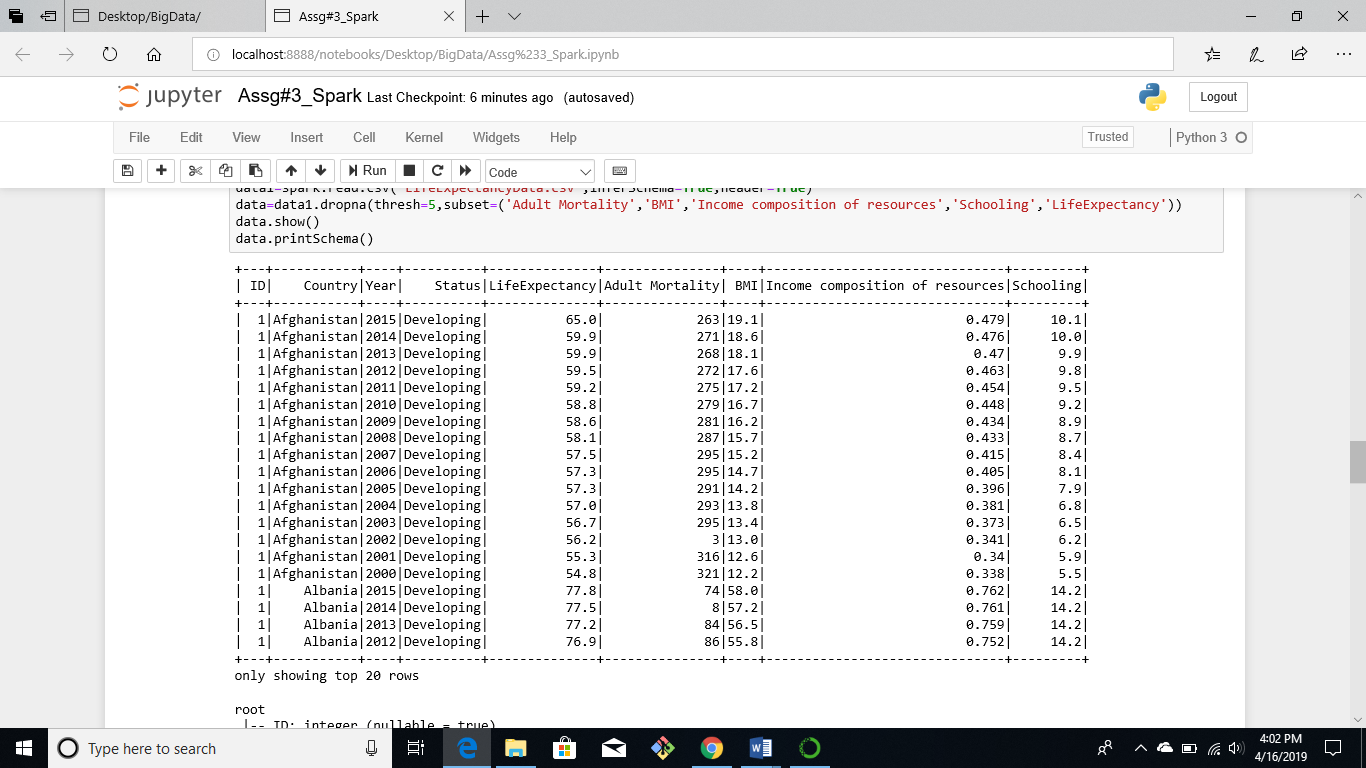
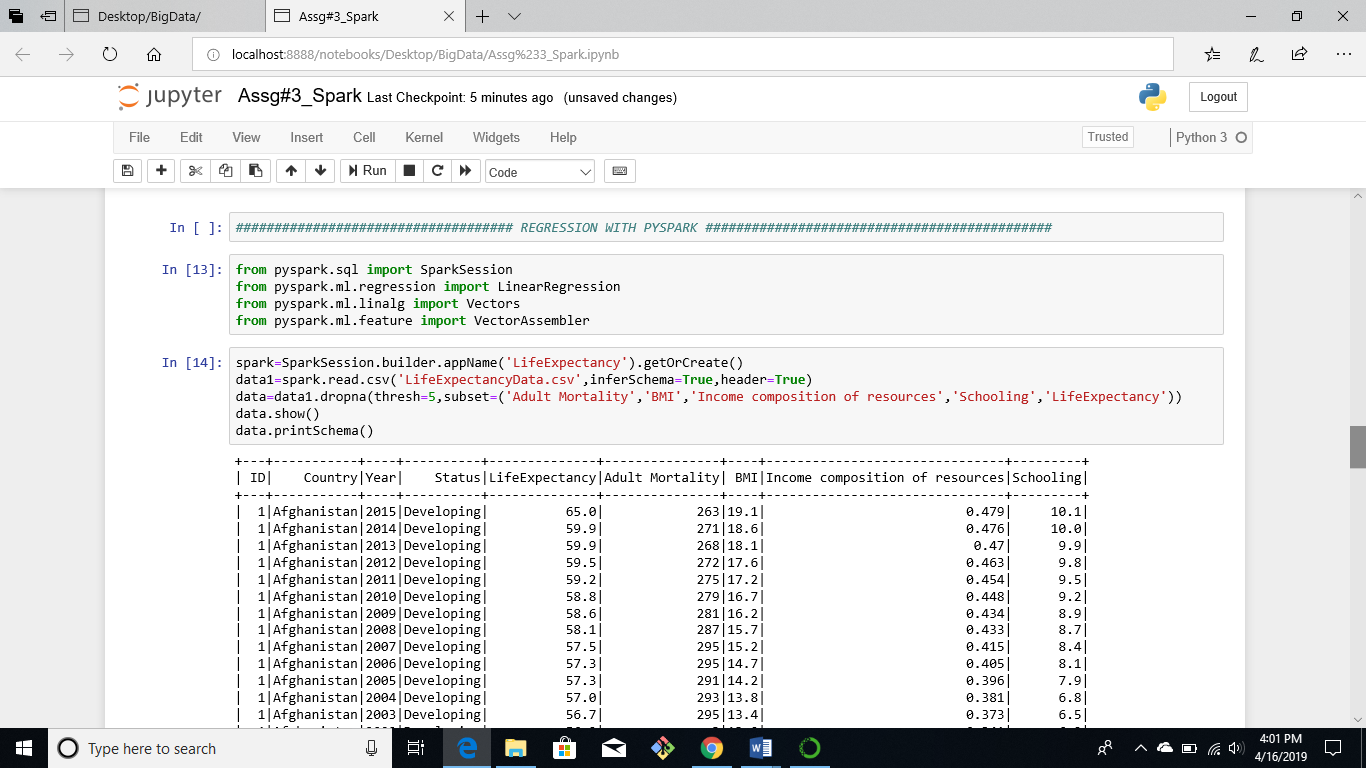
1. **Machine Learning using Spark**



We import SparkSession then intialize the session by using the command “spark=SparkSession.builder.appName('LifeExpectancy').getOrCreate()”

Then, we import the LinearRegression model from the machine learning library. We read the dataset using the read.csv() command inferSchema=True insures that the datatype is maintained while reading it from the csv. In order to make the analysis easier, we drop all rows having Null values.

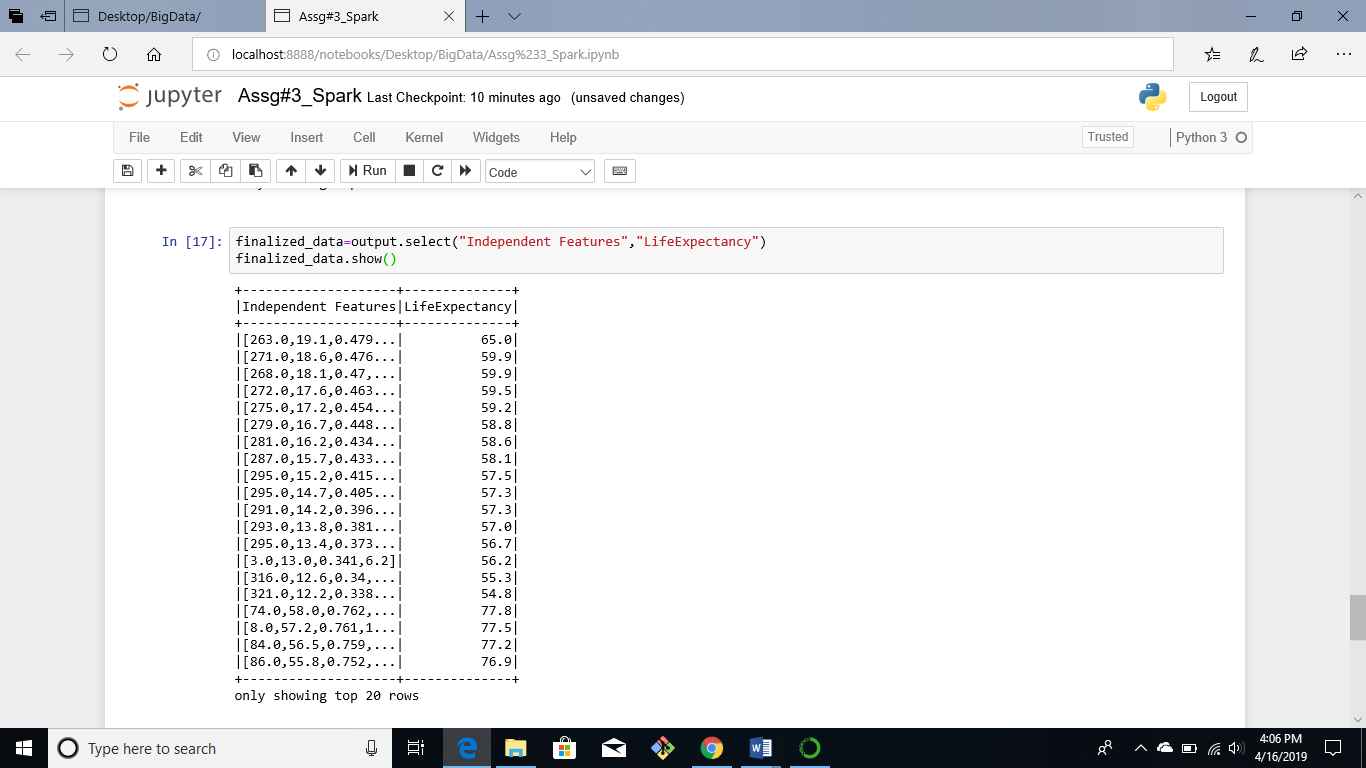
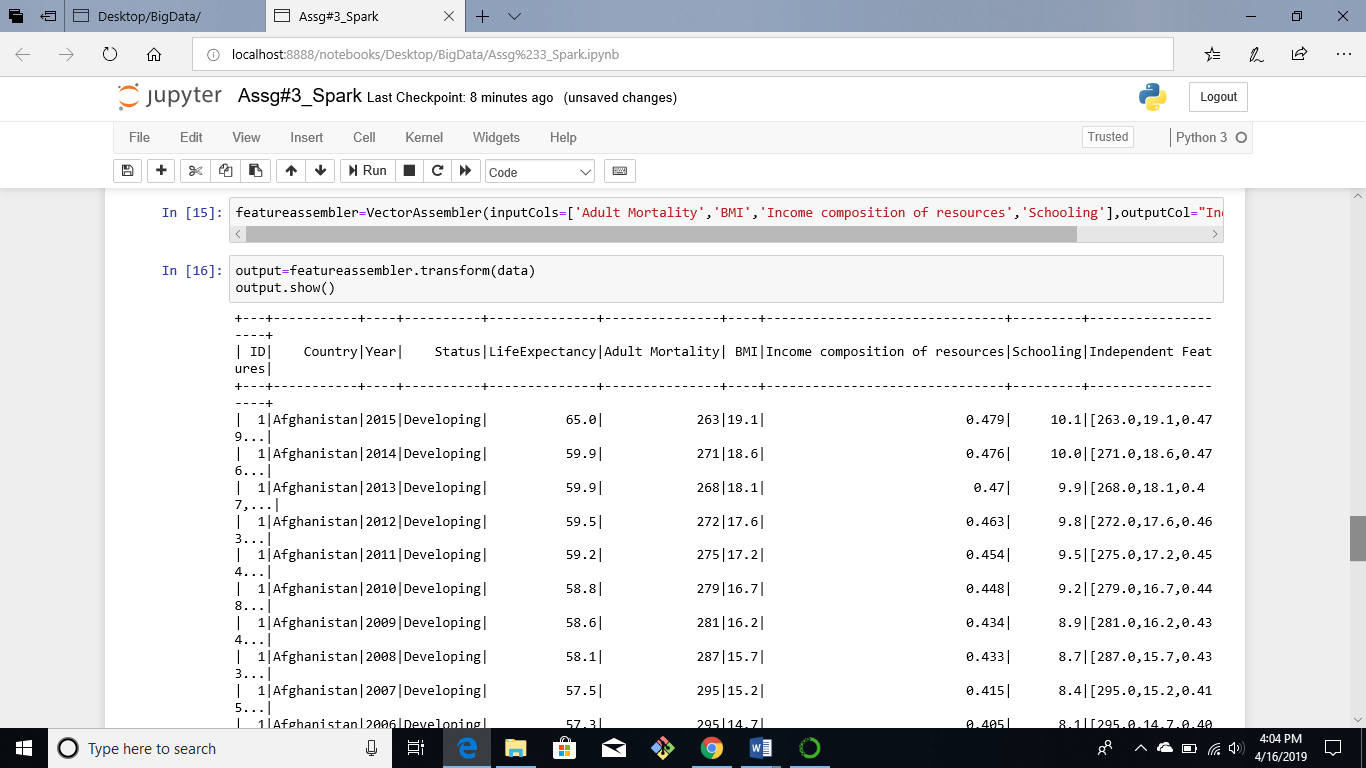
data.show() will print the data frame and data.printSchema() will print the datatypes of the columns in the dataframe.



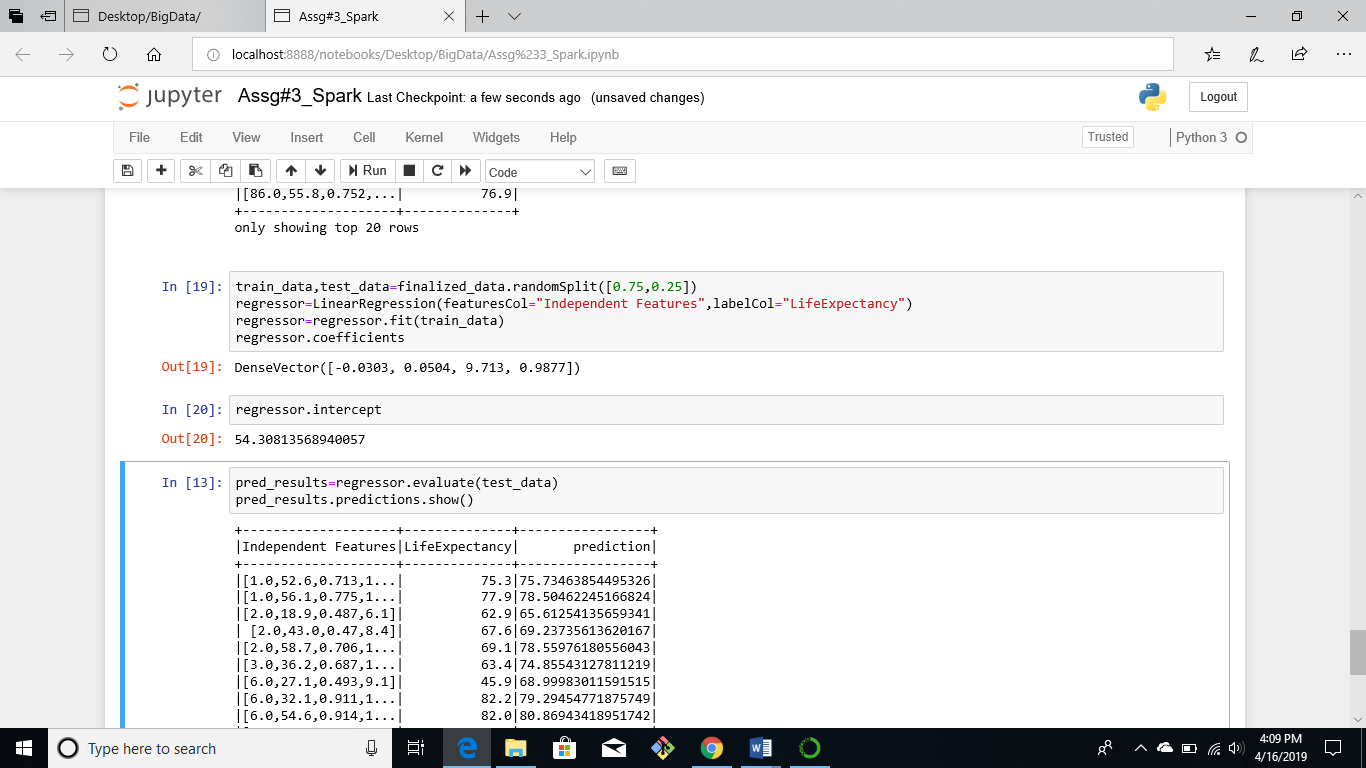


Pyspark requires the input variables to be represented as a vector, the VectorAssembler() groups all the independent variables as a vector, all the inputcols are and outputcol have to be provided to the VectorAssembeler()

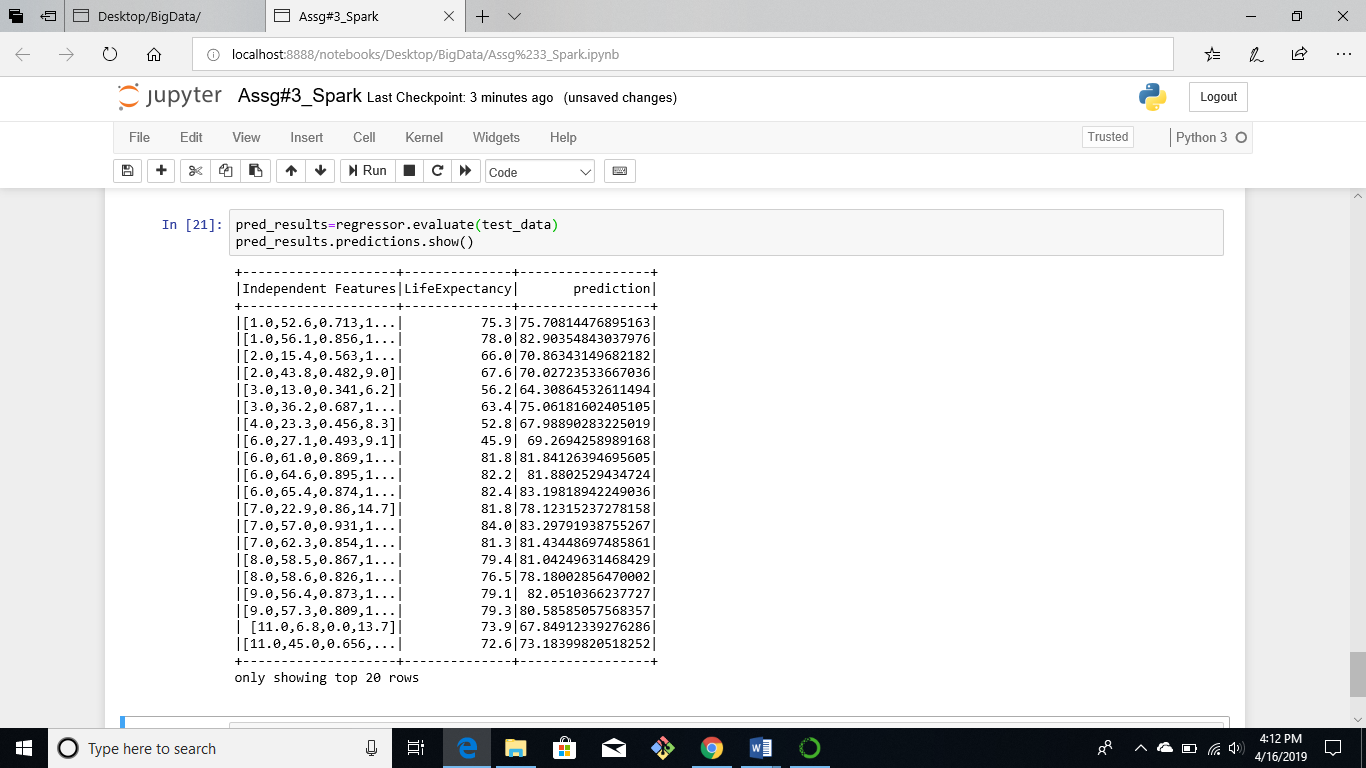
From the screenshot we can see that the independent features have been transformed into a vector for corresponding values of LifeExpectancy.



Next, we divide the data into test and training instances. We select the LinearRegression() model and specify the independent and dependent features. Then, we fit the data to the model.



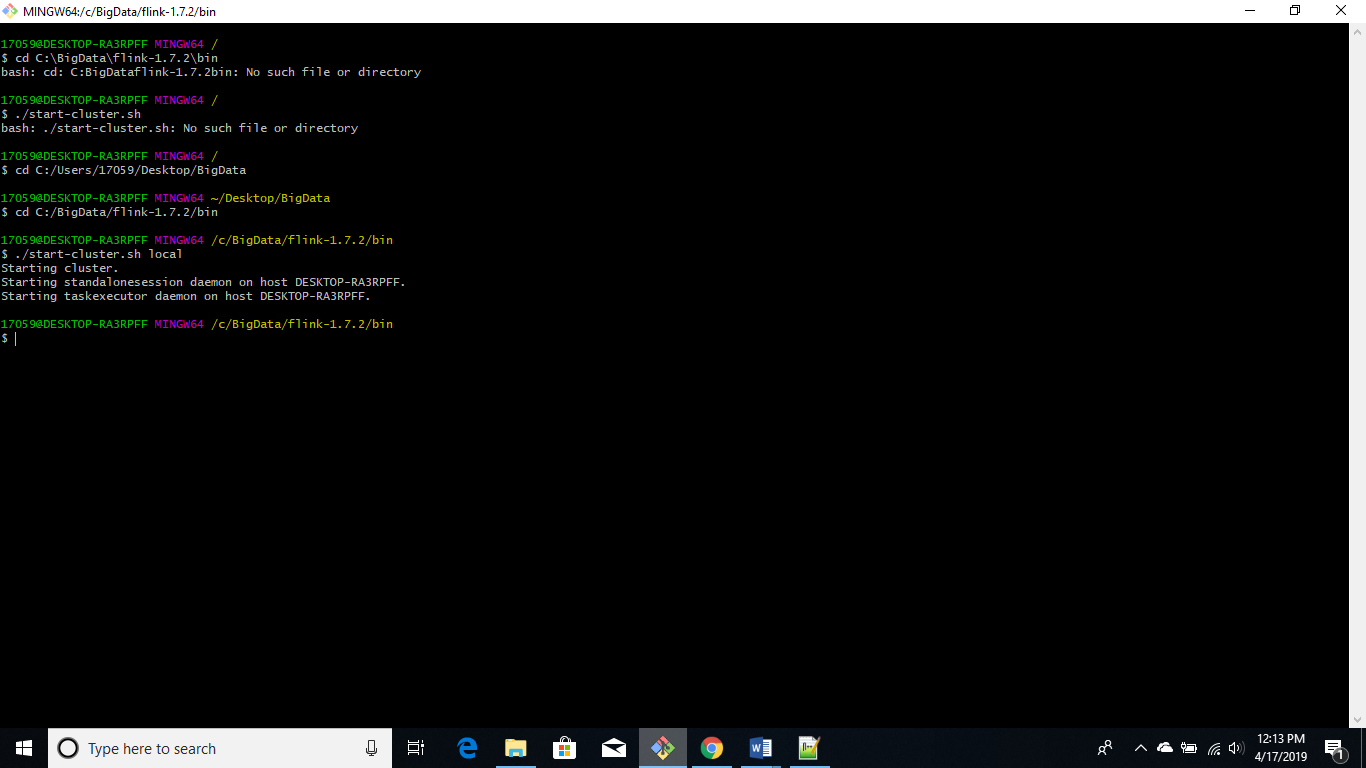
We can check the evaluate the prediction based on the test\_data by using the evaluate function. The table below shows the Independent Feature, the actual value of life expectancy and the prediction based on the model



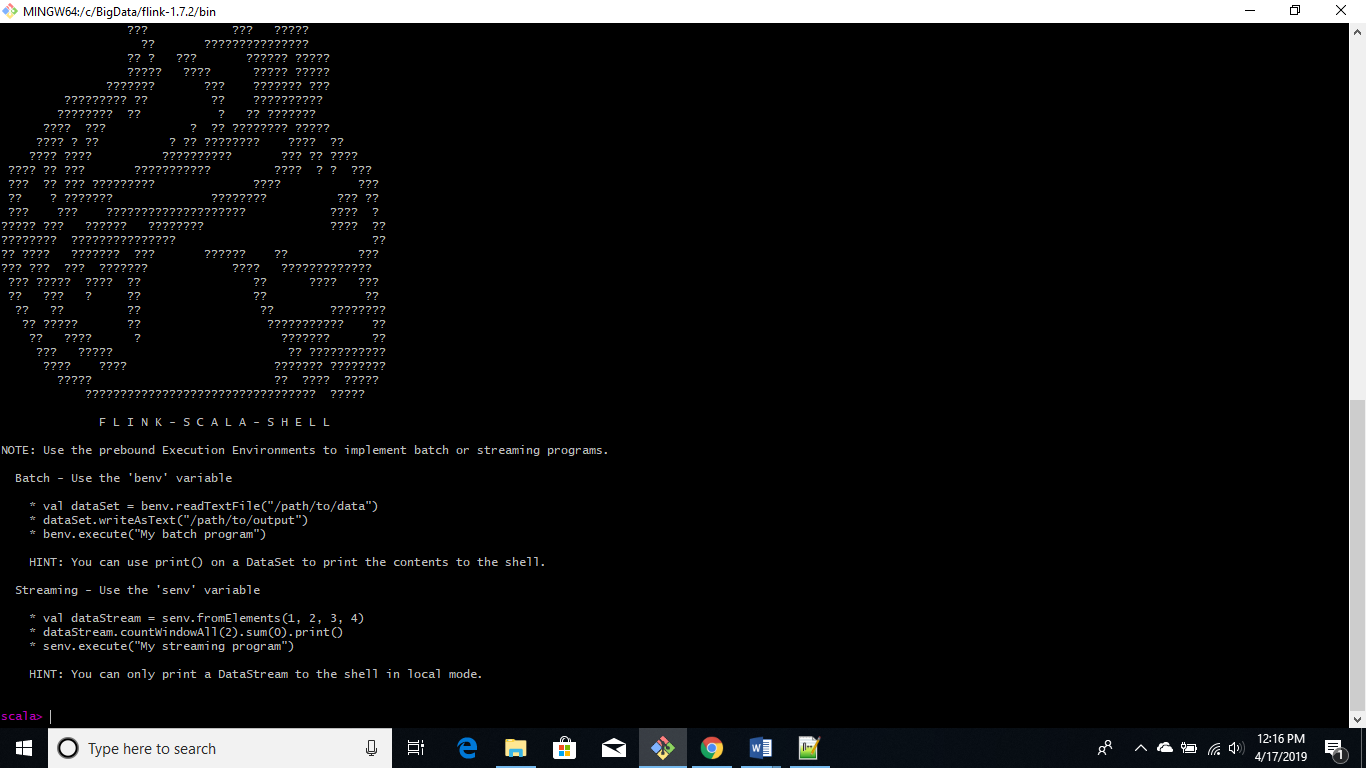
1. **Flink**

Flink is capable of processing real-time streaming data.

To start Flink, we navigate to the Flink bin directory and initiate the cluster by running the ./start-cluster.sh command

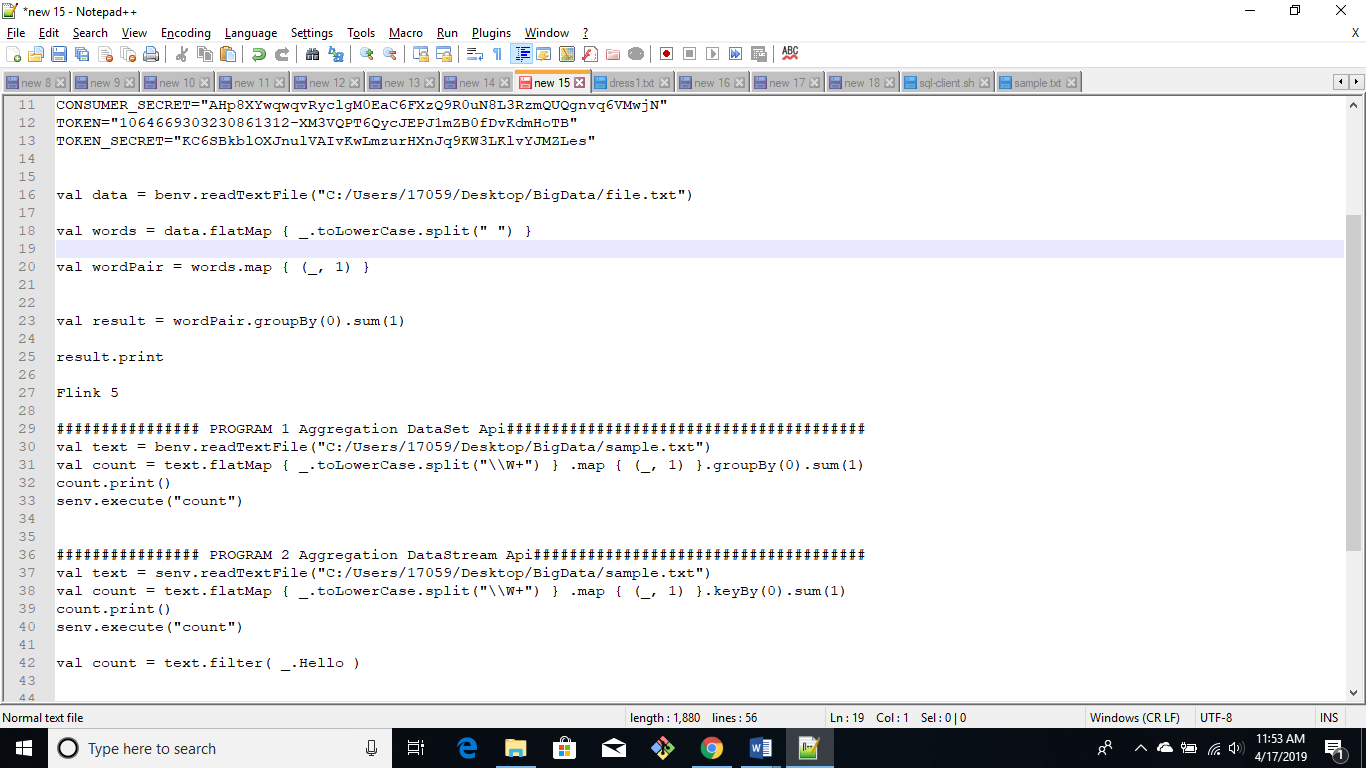


Then, we start Scala-Shell by running the command ./start-scala-shell.sh local

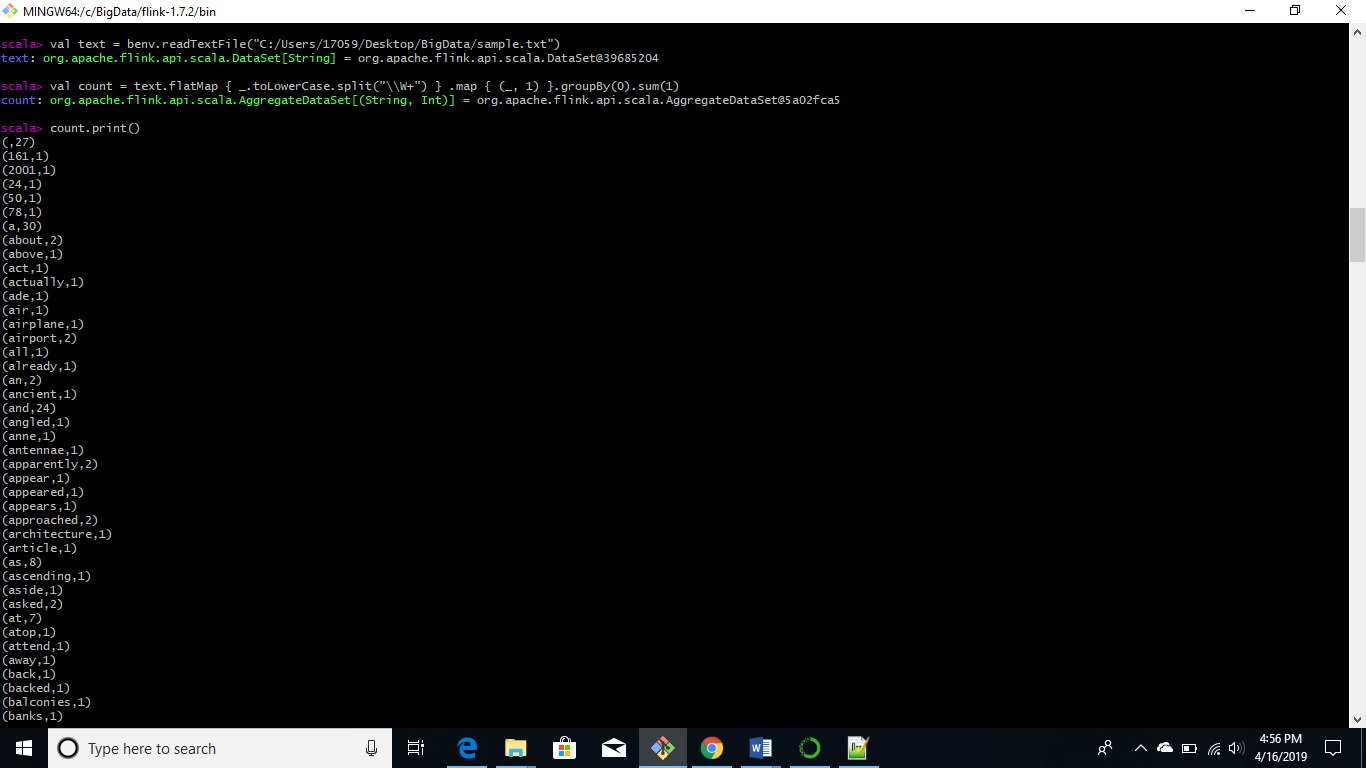


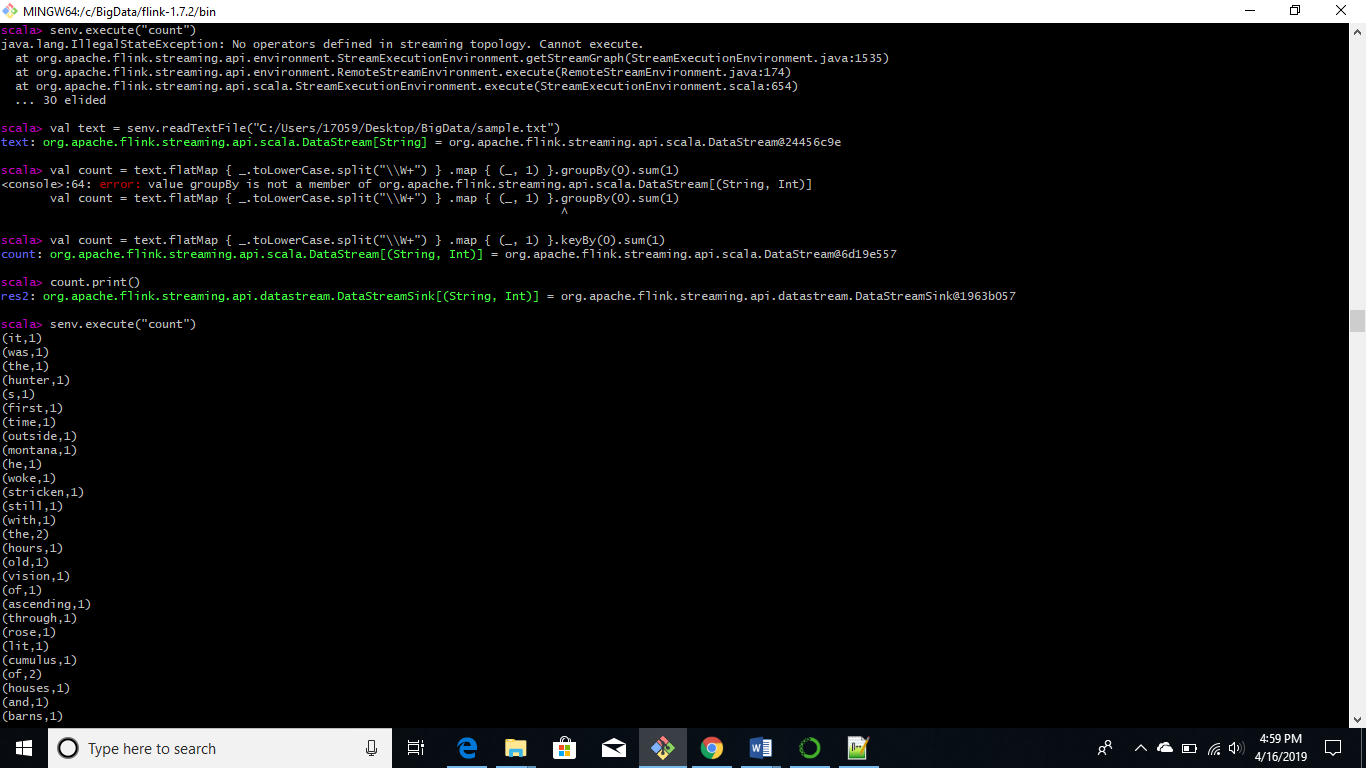
Below programs can be run in the shell line by line

In the first command read and save the sample.txt file to a variable text, benv means that we have set the execution environment to batch, similarly senv means that it is a streaming environment. In the next step, we map each word, group them and then sum them this will give us the number of occurrences of each word in the text file

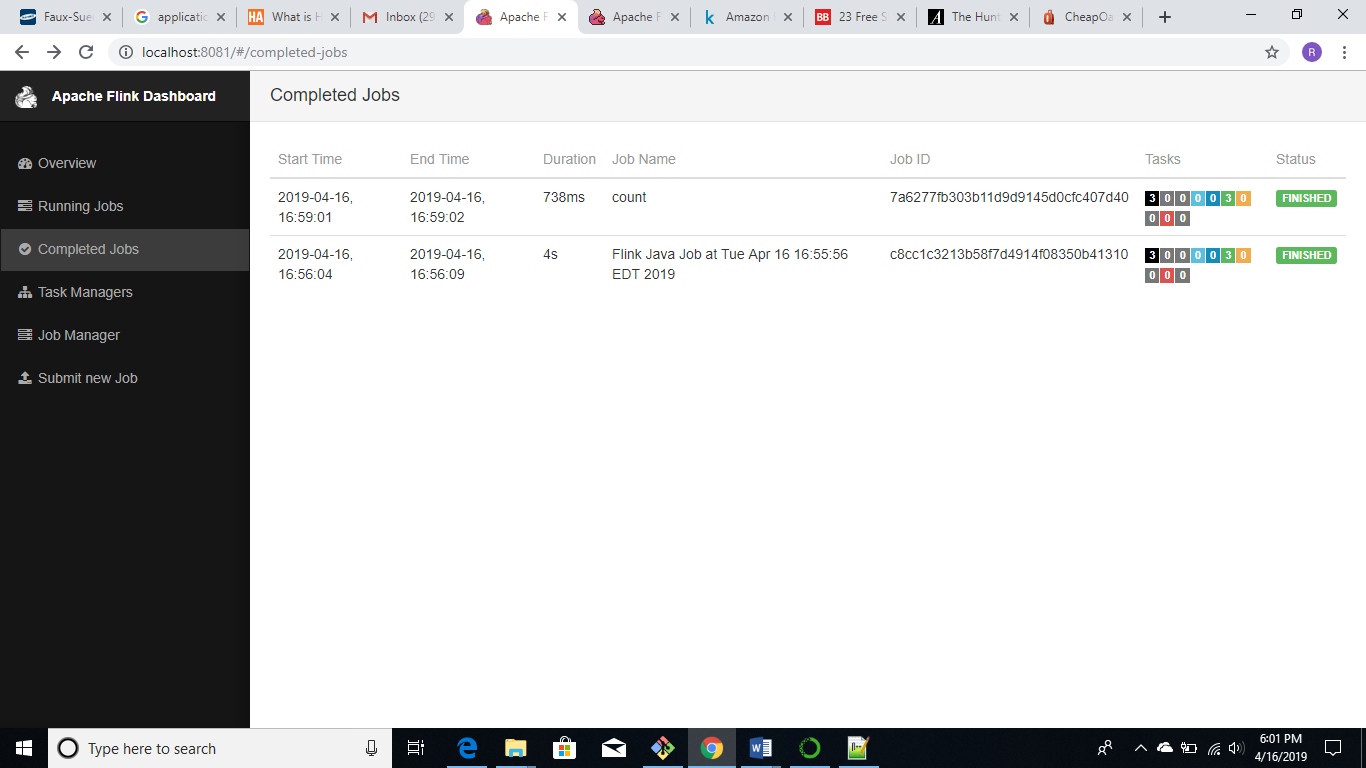


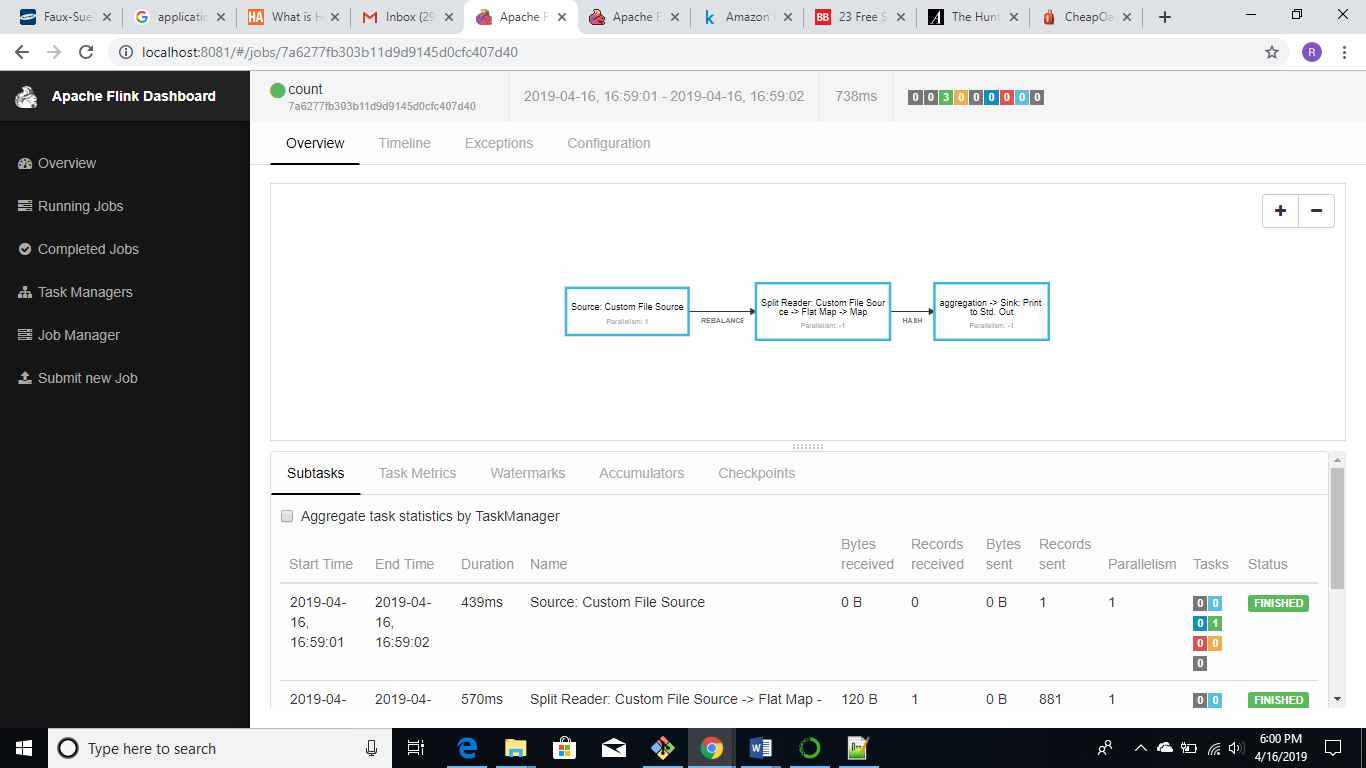
Below are the screenshots for the output

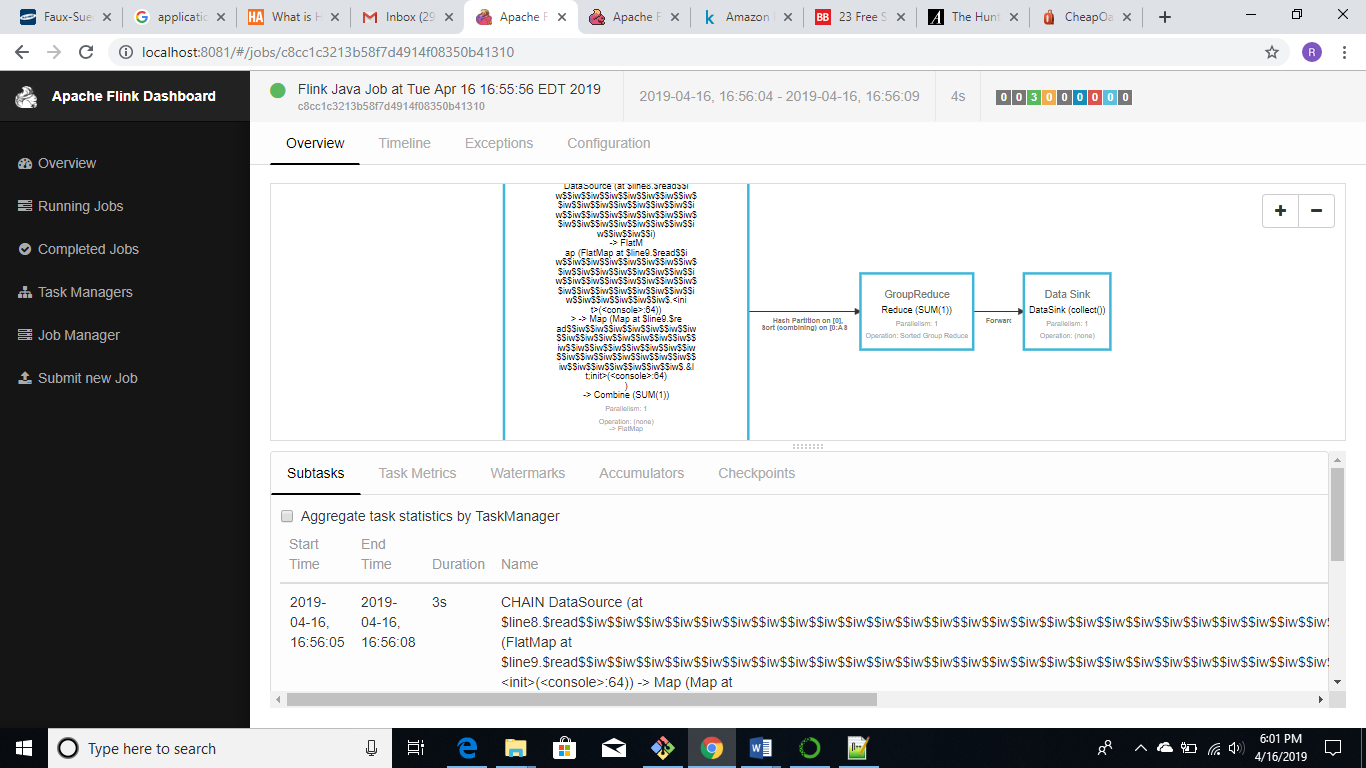




We can check the output of the jobs on the Flink dashboard to see if they have completed successfully, from the screenshots below we can see that both the jobs completed successfully.







**References:**

[1] <https://www.kaggle.com/ronitf/heart-disease-uci>

[2] <https://www.kaggle.com/kumarajarshi/life-expectancy-who>