Shaun Marple

3/27/2020

Abstract

A term project tutorial on some of the many facets an analyst can use in making sense of data, using the Spark platform and the Databricks framework

Term Project: Working With Databricks

CS779 Advanced Database Management

Contents

[Objective: 2](#_Toc36222696)

[Introduction 2](#_Toc36222697)

[What is Spark? 2](#_Toc36222698)

[And why use the Spark platform rather than Hadoop, for example? 2](#_Toc36222699)

[What is Databricks? 3](#_Toc36222700)

[Why Databricks? 3](#_Toc36222701)

[Learning by Example 3](#_Toc36222702)

[Extracting Data 3](#_Toc36222703)

[Read in CSV with PySpark 4](#_Toc36222704)

[Read in CSV with R 5](#_Toc36222705)

[Example 1 6](#_Toc36222706)

[Example 2 7](#_Toc36222707)

[Word Cloud Embedding 9](#_Toc36222708)

[Back to Spark Data Frame 11](#_Toc36222709)

[Aggregating Data 12](#_Toc36222710)

[Using SQL to Graph on Databricks 13](#_Toc36222711)

[Dashboarding 14](#_Toc36222712)

[Appendix 15](file:///E:\Boston_University\Grad_School\CS779\Project\TermProject_ShaunMarple.docx#_Toc36222713)

[References 16](#_Toc36222714)

# Objective:

To learn how to import and transform data so that meaningful analyses can take place in the Databricks environment. This paper will be will portray a firsthand look at Spark and will explain the advantages and disadvantages that Databricks brings to the data scientist’s table. I will also look to integrate various software of the versatile Spark platform in order to leverage some of their strengths in working with the data. Whether I am importing with Scala or PySpark, aggregating with SQL, or transforming in Python, I am looking to demonstrate how someone would transform and consume semi structured data and be able to report some simple analyses.

# Introduction

What is Databricks? What is Apache Spark? I will give a brief background in both and discuss what we need to know about them. I will explain that there are pluses and minuses to developing in the Databricks cloud and briefly list those as well. Since I am a novice, I will also discuss that the purpose of the paper has to do with Spark at a first glance, from an analytical developer’s point of view. I will also demonstrate how to extract, transform and create visualizations using the Databricks framework, using Amazon review data to show how productive an analyst can be while leveraging various strengths of each language on the Spark platform.

## What is Spark?

Spark is a distributed computational framework of interconnected platforms for Big Data projects.3

### And why use the Spark platform rather than Hadoop, for example?

There are many reasons why I would use the Spark platform. For one, Spark is versatile3. Rather than working in the MapReduce environment that you would in Hadoop, which, in my opinion is much more difficult to code, Spark works with familiar languages such as Python, Java, R, SQL, and Scala all run native on the Spark platform.

In addition, because data is processed in memory, it is considered to be 100 times faster than MapReduce. Spark even works quickly on smaller datasets, as opposed to Hadoop. Because Hadoop first, has to distribute the data over all of the nodes, it takes a while to get going. Once it has begun, however, is when the processing is fast. While data processing speed over large data sets works great in Hadoop, running smaller datasets is very slow. Slower than native Python in fact. Because Spark runs on in-memory processing, Spark can be just as fast on both small and large amounts of data.

Spark is also used for machine learning and supports its own graphics. Although I did not work with machine learning in this project, I can say that running neural networks, especially reoccurring neural networks, the amount of processing power need to run these algorithms is substantial and may be near impossible to run them on even the most “souped up” desktop computer. So, the fact that Spark can support these algorithms is a nice advantage.

Spark can also be integrated on just about any file system including: Hadoop HDFS, MongoDB, and Amazon S3. Not only is that useful, but I have found the Spark platform to be much easier to work with, especially in pulling data than HDFS, for example.

HDFS works on command line imports, which is something I’m not super comfortable with, and would much rather pull data in through a scripting interface. In my brief introduction on the HDFS platform, I’ve had to use such software like ssh and Putty. In my opinion, editing scripts is just done much more easily when you are working within an editor of some sort: whether we are talking about Databricks, Jupyter, Google Colaboratory, Spyder, or any other useful editor. Editing in Putty is a pain in my opinion.

## What is Databricks?

“Azure Databricks is an Apache Spark-based Analytics Platform.”1 Databricks is basically an interactive workspace that enables collaboration between data scientists, data engineers, and business intelligence developers. While it leverages all of the power and advantages of Spark, it also lets coworkers share the application, which could save lots of communication trouble during development.

Shared applications on a platform like this would be quite useful. At work, I am responsible with designing a data warehouse in Oracle, as well as developing the ETL tool in Python. Not only do I have to have multiple software open at the same time (i.e. SQL Developer, and Python), but I also have to communicate on a weekly basis, with the data modelers in anther team to let them know where I am in the process, what the data structure looks like, and when they can expect my part to be done. By using Databricks sharing, all of this extra communication, that pulls us off of being productive, would reduce a lot. Probably drop to monthly communication, rather than weekly.

### Why Databricks?

While the Spark platform is very useful by itself, Databricks adds another layer of versatility and usability. For one, it extracts data easily, whether the data in in JSON, CSV, SQL servers, MongoDB, or any other form of data, Databricks’ documentation support is available for anyone looking to extract this data. It also integrates well with the Microsoft stack which includes Azure SQL Database, IoT Hub, Stream Analytics, Machine Learning, Data Lake Power BI, and many other software.

# Learning by Example

For this demonstration, I will be pulling data from an Amazon review CSV file. Though not considered to be a big data job, it will serve as a proof of concept for the limited timeframe allotted.

Here, I will demonstrate the following:

* extracting CSV data into Databricks, using different technologies
* some of the methods I used in transforming the Amazon review data
* how the data can be cleaned and what technologies may be best in these areas
* some analytics that we can do, even on such simple data
* how to create a simple dashboard

## Extracting Data

Since I am familiar with Python and R, I wanted to see what needed to be done in extracting data with PySpark and SparkR. This might be useful to learn more than one way since much of the unstructured data would likely be pulled in and transformed with Python. So, rather than pulling it in with Scala and having to convert back to Python, it may just be easier to pull the data in with Python.

In addition, Statisticians who are using R for advanced computational purposes may want to know if SparkR works differently than the original R in pulling in data onto the Spark platform. So I have elected to show both Python and R here.

### Read in CSV with PySpark

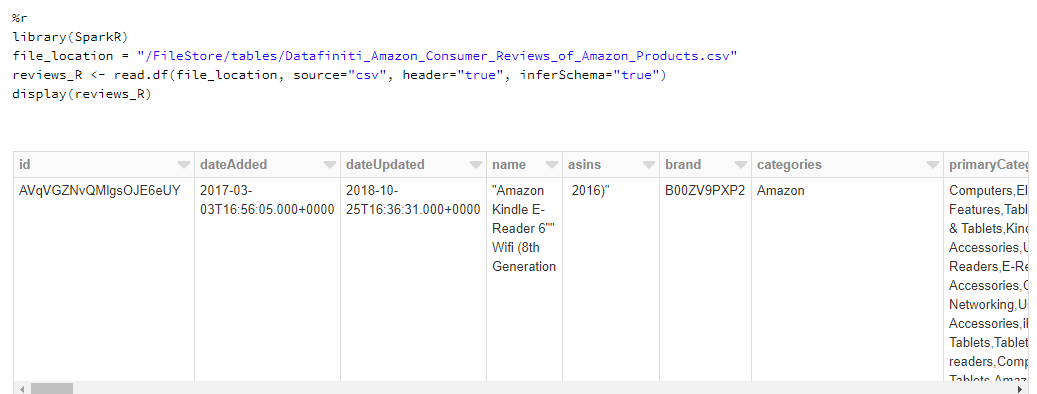


In the above script, I’ve set:

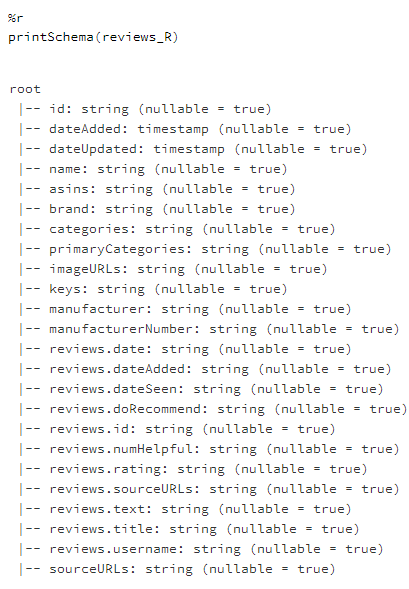
* my file location to the path to the csv file
* infer\_schema = ‘True’
* first\_row\_is\_header = ‘True’ to let Spark know that the headers are contained within the data
* delimiter = ‘,’ to let Spark know that it is comma separated values. Just because the file\_type = ‘csv’, doesn’t mean that it necessarily is separated by commas. It can also be separated by tabs or spaces. So unless specifying the delimiter, Spark may not be able to infer this information

By applying the method spark.read.format(), we are able to import the data successfully into a Spark dataframe.

### Read in CSV with R



As we can see from the example above, just about everything looks the same in this example as the one above in in PySpark. If we were looking to learn more about the data we can print the schema like:



## Example 1

For this first demonstration, I’d like to show how I found the misalignment in the data columns.

Recall the PySpark code above, where I set our data frame = [df](#_Read_in_CSV). Now we must convert to a Pandas Python data frame.



Next, rename all column headings. Notice my column heading, ‘nada’. I set that as a place holder when I realized that the columns were offset.

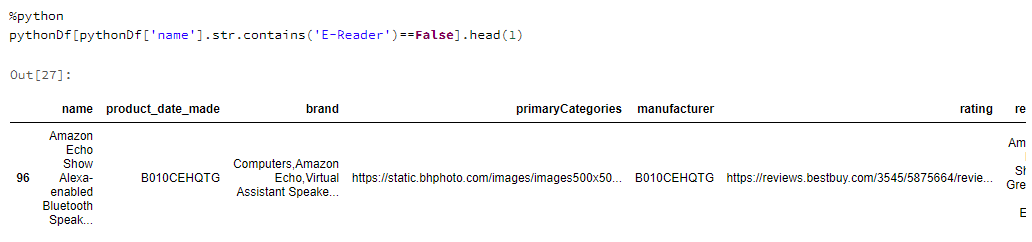


Next, I decided that I should clean up the data to see what I might be able to learn from it.



I removed added spacing, commas, quotes, and subset the data for the columns that I expected to use in my analysis.

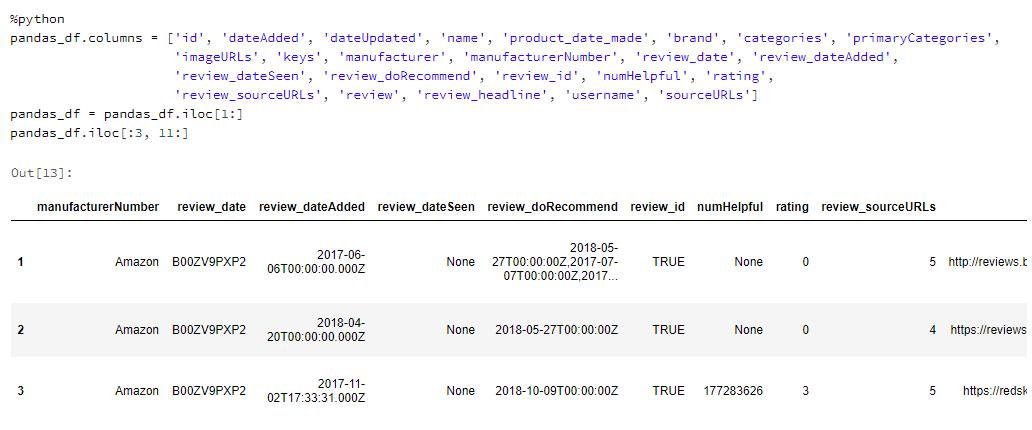
Then, after a few queries of this data, I decided to shrink the dataset and search through the data, looking for anomalies.



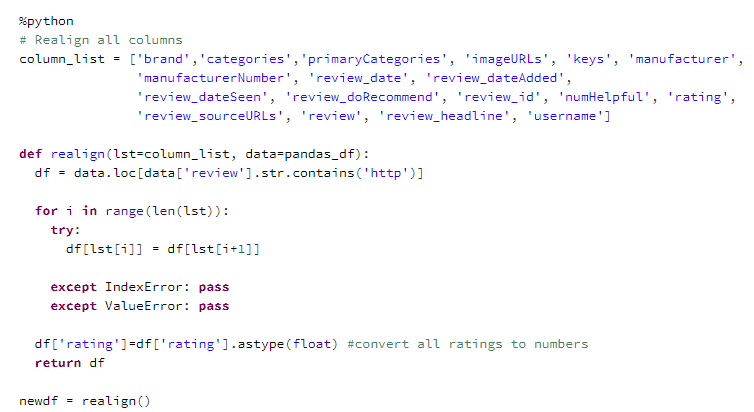
The above code searches the name column for something that doesn’t contain “E-Reader”. From there, I was able to see some of the rows that contained data, that wasn’t in the right columns.

## Example 2

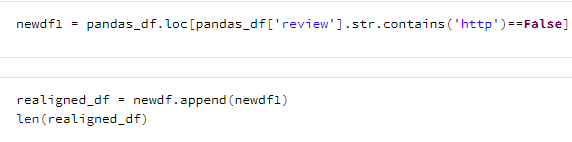
In this example, I went back to the pandas dataframe and designed a function to correct where data went wrong.



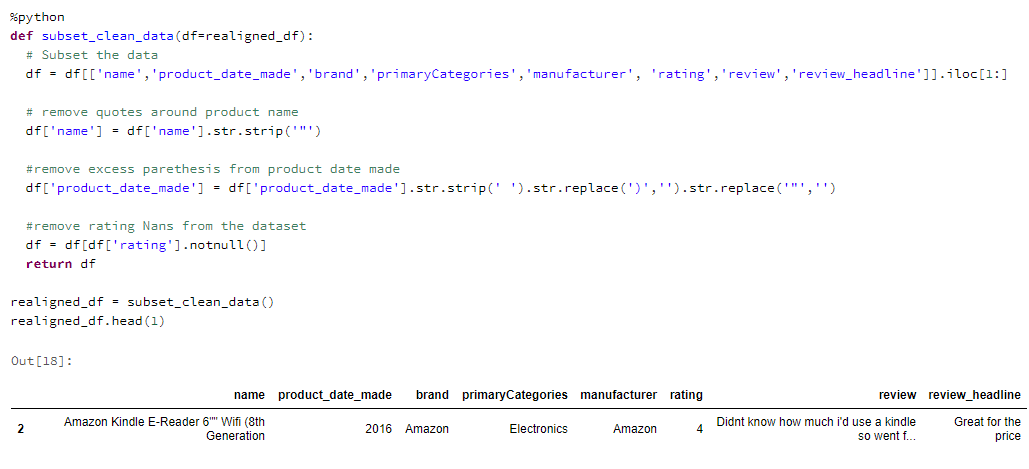
As we can see, none of these columns are lined up correctly. By looking at the code above, or counting the columns, we can see that the table is only showing the first 3 rows of columns 11 on. So, I was able to find that when the reviews included urls, this was most likely a mistake, so I anchored my function from there.



The function above searches the data frame for anytime a review contains “http”. If it does, all of the columns shift to the right one column. The next two lines of code split out the data to newdf1, which is where no shift is required and newdf, which is where a shift is implemented. Realigned\_df just appends both back together again.



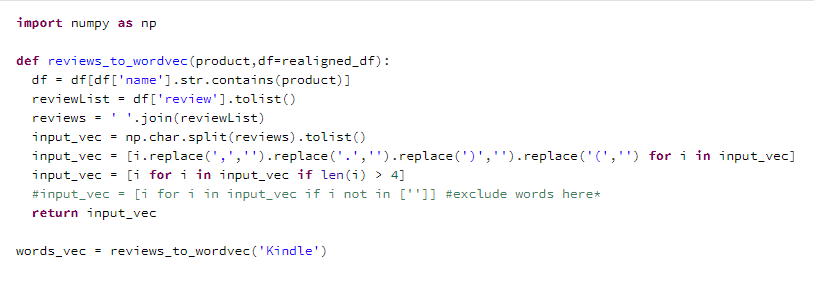
Next, I cleaned the data of apostrophes, commas, spaces, removed reviews without ratings and printed the top row:



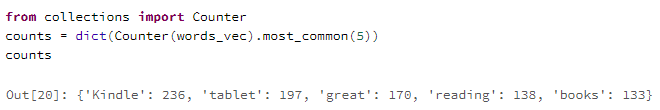
## Word Cloud Embedding

Since now the data is cleaned up enough to analyze, I decided that now was the time where I wanted to get something graphical for the project. I also wanted to apply a word vector embedding onto the reviews to get an idea on which words were used most often on these Kindle E-Readers. So, I decided to implement the word cloud via Python.

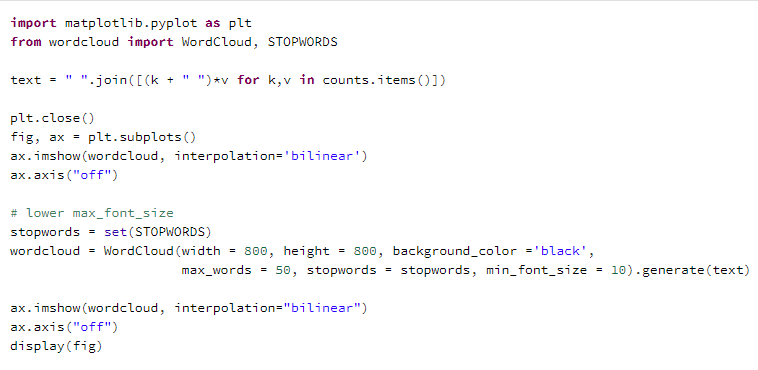
In order to create one of these things, I had to reduce all of the reviews to lists of lists of strings. Since my question that I wanted answered dealt with Kindles, I decided to write a python function to allow me to enter anything I wanted and it will subset based on my input.



Next, each word must be counted and I’ve printed the 5 most common words.



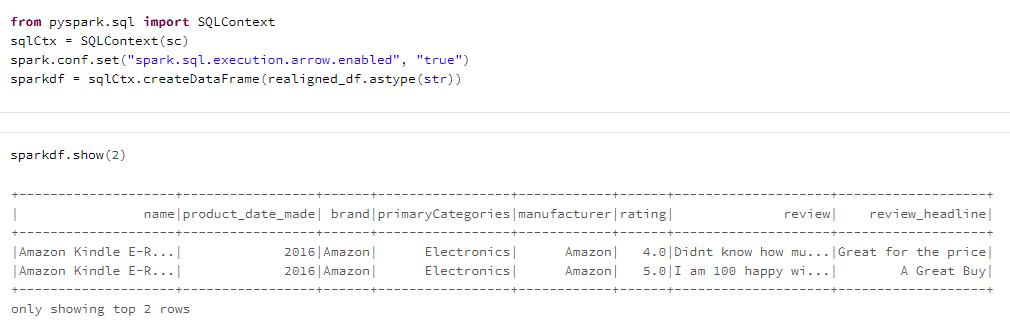
Finally, by using matplotlib.pyplot, a Python library, I was able to create a word cloud image.



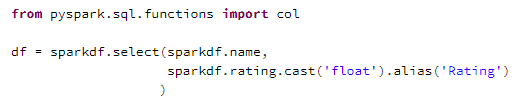


## Back to Spark Data Frame

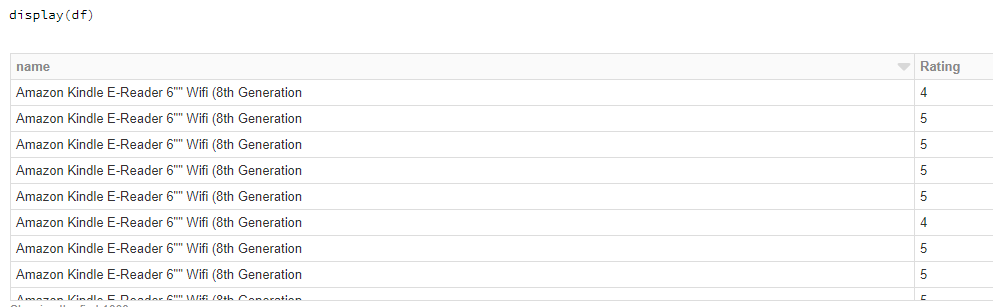
Now that I was able to get a decent graphic from a python library, I wanted to see about using Databricks inherent graphics to show me a little bit about the data distribution. So, in order to do that, I would first have to convert the pandas data frame back to a spark data frame.



As we can see in the table above, the data looks much better and should be that much easier to analyze. However, notice the final line of code above: spark = sqlCtx.createDataFrame(realigned\_df.astype(str)). In order for me to get the code to move the pandas data frame to a spark data frame, I first had to convert all columns to strings. Apparently, Spark has some issues in dealing with object oriented languages. So, since we still need the ratings to be numeric, in order to get the proper aggregated information, we must cast it as float data type.

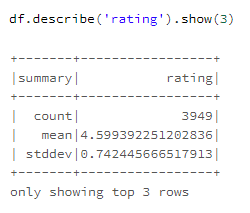


By looking at the syntax in the above code, we can see that I’ve casted rating as float, changed the heading name to Rating, and also selected name as part of the query. Here is the result:



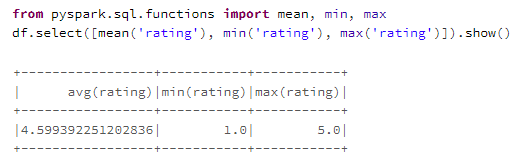
## Aggregating Data

From here, we can learn a little about the summary statistics. We can use describe:



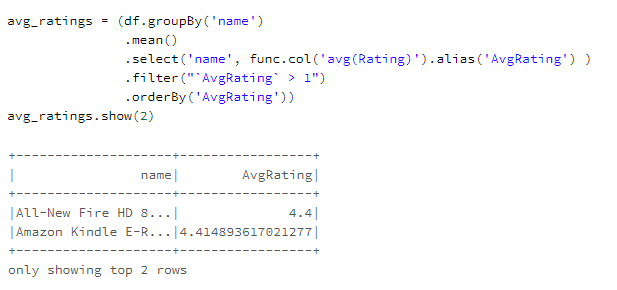
I only chose to show the top 3 rows since the last 2 rows are min() and max(). Two stats that are not too interesting on their own, especially if you have an average of about 4.6 stars and a standard deviation of less than 1. That information tells me that the distribution is likely to be considerably tight.

However, if we wanted to get the min and max information, we can also use the following alternative syntax:



Not only is this syntax using a library, but as we can see, the data is transposed.

In this next code, we will run a group by query, that takes the average ratings by product and ignore all data with no ratings (is null).

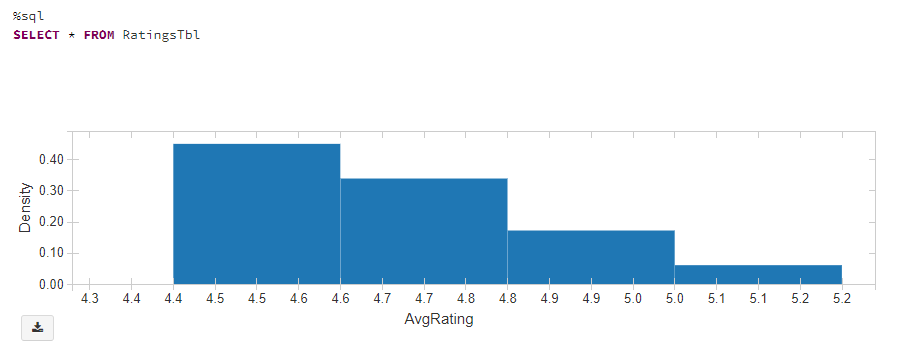


## Using SQL to Graph on Databricks

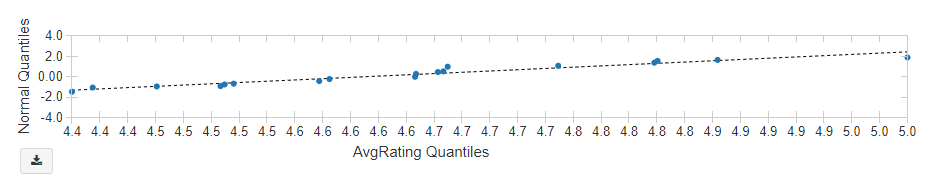
First, we must convert our ratings table to a SQL table.



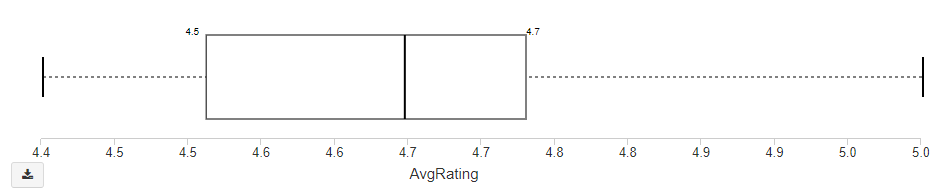
Now, I want to see what the distribution of the ratings data looks like, so I will first select from the table and then use the Databricks interface to display a histogram.



Next, I’d like to see a QQ (Quantile) plot:

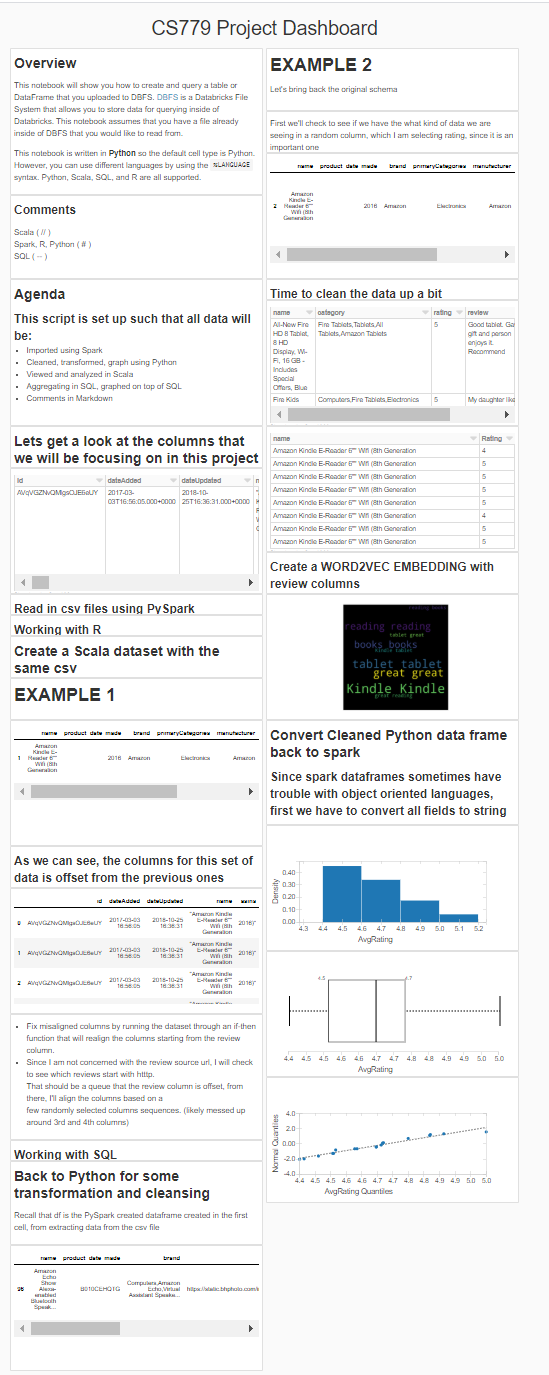


Followed by a box and whiskers’ plot to get a full understanding of the distribution of ratings:



## Dashboarding

In order to put it altogether, I used Databricks to put together a dashboard based on the Databricks’ notebook. I then removed redundant information from the dashboard, rearranged the images and headlines and save it as a dashboard, just to solidify just how useful this framework has proved to be. [Please see Appendix A.](#_Appendix)



Appendix A

# Appendix

# References

[1] Microsoft Azure (2019). “What is Azure Databricks?”. *Microsoft*. Retrieved from: <https://docs.microsoft.com/en-us/azure/azure-databricks/what-is-azure-databricks>

[2] Marr, Bernard (2019). “What Is Spark in Big Data?” *Bernard Marr & Co. Intelligent Business Performance*. Retrieved from: <https://www.bernardmarr.com/default.asp?contentID=1079>

[3] Polnar, Jack (2020). “Big Data & NoSQL DBs”. *Boston University*. CS779.

[4] Nanua, Roshan (2019). “6 Reasons to Use Azure Databricks Today”. *Hitachi Solutions*. Retrieved from: <https://us.hitachi-solutions.com/knowledge-center/6-reasons-to-use-azure-databricks-today/>

[5] IntelliPatt (2020). “What is Apache Spark” *IntelliPatt*. Retrieved from: <https://intellipaat.com/blog/what-is-apache-spark/>