**Project 2 Report**

**Analysis on Amazon Ratings**



**CS/MSA 6500 Big Data Analytics**



**Group 10**

**Submitted by:**

Cheng Chen

Ravali Bongoni

**Project Overview**

In this project, we are analyzing Amazon item rating data. We downloaded the data from the given link in the project document. We also tried to import data from docker and found both data are the same. The data is in the CSV text file format. The size of the file is 304 megabyte(mb). It contains four columns with information: userID, itemID, rating and timestamp. There are three tasks assigned in the project. Task 0 and 1 are given, while we selected three separate parts for Task 3 after the approval from the professor.

We used a terminal, Jupyter Notebook, and Pyspark Notebook to run spark, for the given task. We also used a panda library, matplotlib and word cloud to do this project. We also used the R language for generating graphs. We are four members in the group. So, we divided the task and everybody contributed to complete the assigned task. This is an interesting project to use Spark effectively and get the meaningful result. We did virtual meet ups for discussion and communication with group members.

We followed the lecture slides, read the given cheat sheet on spark on canvas and went through the activity task to do this project. When we encountered errors, we used the Stack Overflow website, YouTube, to see recommended error fixing methods and also we discussed with team members for improving errors.

**Task 0- Filtering/CleanUp**

In this task, we need to filter the dataset. In order to do that, we accessed the dataset from the given link mentioned in the project documents.

http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/ratings\_Electronics.csv

We have to keep the newest rating from each user and get rid of older ratings and duplicates. After doing that, we have to do readable timestamps conversion. Since, we have two parts on task 0. The first part of task 0 was done by terminal. First, we wanted to filter out duplicates, which means users who rated a specific item more than once. We used the following line of code to perform the filtration task. The importing original dataset step was quite smooth, but it took us 2 days to finish this task.

*# hdfs dfs -mkdir /project2*

*# hdfs dfs -put /root/ratings\_Elec=ronics.csv /project2/ratings\_Electronics.csv # pyspark*

*>>> input = sc.textFile('hdfs://localhost:8020/project2/ratings\_Electronics.csv') >>> input.first()*

*20/04/07 23:02:52 WARN DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.*

*u'AKM1MP6P0OYPR,0132793040,5.0,1365811200'*

*>>> from pyspark.sql import Row*

*>>> rows = input.map(lambda x: x.split(',')).map(lambda x: Row(userId = x[0], itemId = x[1], rating = float(x[2]), timestamp = int(x[3])))*

*>>> rows.first()*

*Row(itemId=u'0132793040', rating=5.0, timestamp=1365811200,*

*userId=u'AKM1MP6P0OYPR') >>> amazon = rows.map(lambda row:*

*list(row)).toDF(["itemId","rating","timestamp","userId"])*

*>>> amazon.first()*

*Row(itemId=u'0132793040', rating=5.0, timestamp=1365811200, userId=u'AKM1MP6P0OYPR')*

In this part, we selected the userId column and itemId column, then we combined the two columns together to create a new dataset which was named amazon. We checked the number of rows of the original dataset, and the result was 7824482 rows.

*>>> amazon.select(amazon.columns[:1]).count()*

*7824482*

Next, we also checked the distinct number of the new amazon dataset, and the result is 7824482 either.

*>>> from pyspark.sql.functions import concat,col,lit*

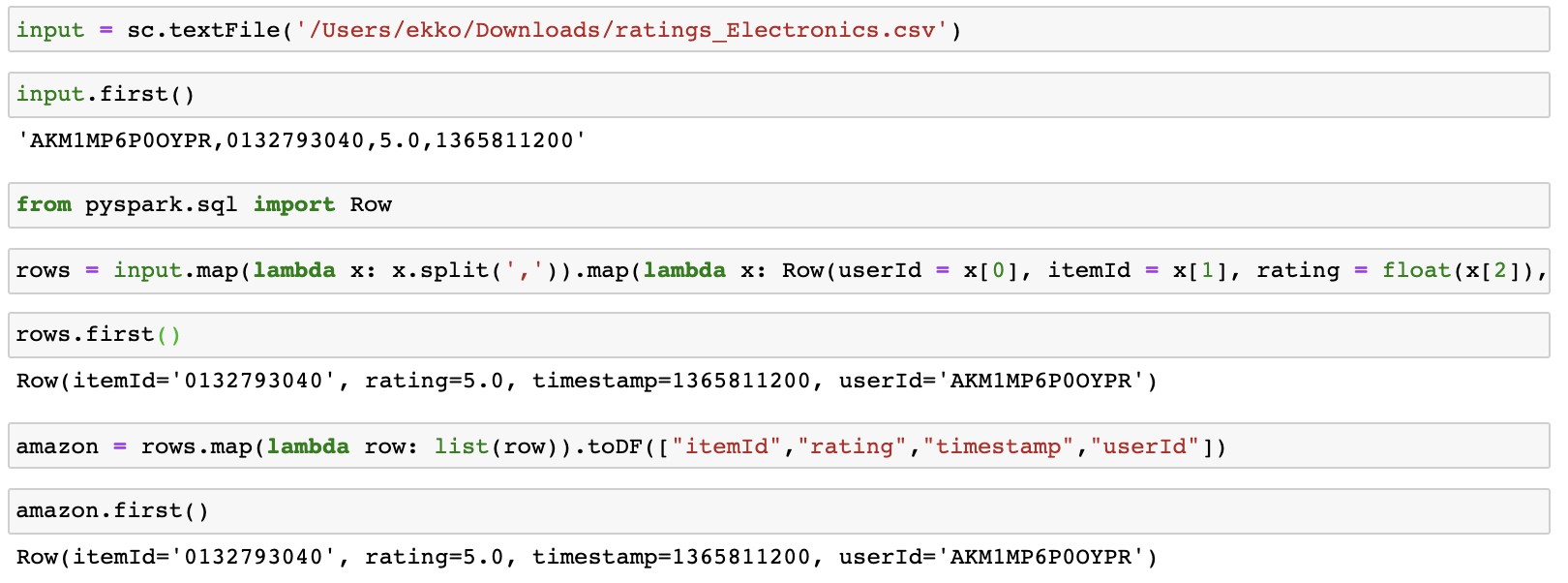
*>>> userId\_itemId = amazon.select(concat(col("userId"),lit(','),col("itemId"))) >>> userId\_itemId.distinct().count()*

*7824482*

We concluded that the original data have no duplicates. No users who rated a particular item more than once.

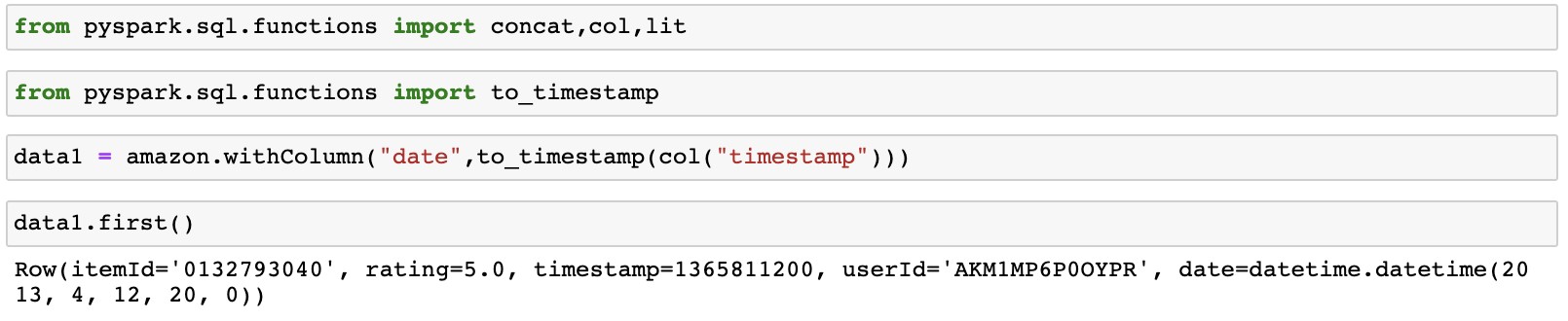
Second, we wanted to convert the timestamps into a human readable form. Because the terminal is not stable and hard to save codes, so we did this part on jupyter. We build the spark environment on the jupyter notebook in order to run spark codes.

We read the csv file again, and named all the columns.

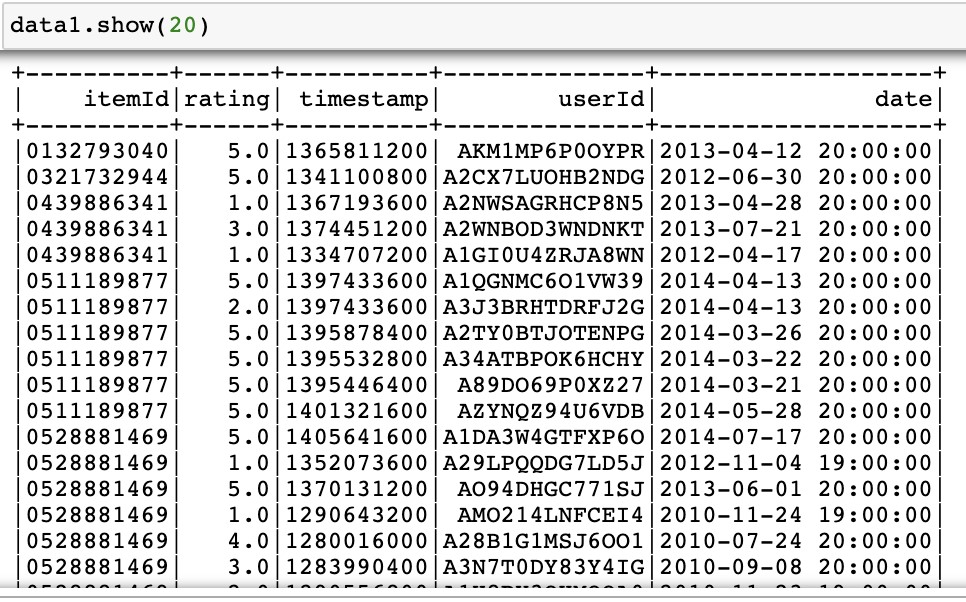
 After we named these columns, the order of these columns changed automatically.

We had to check the order in order to avoid mistakes.

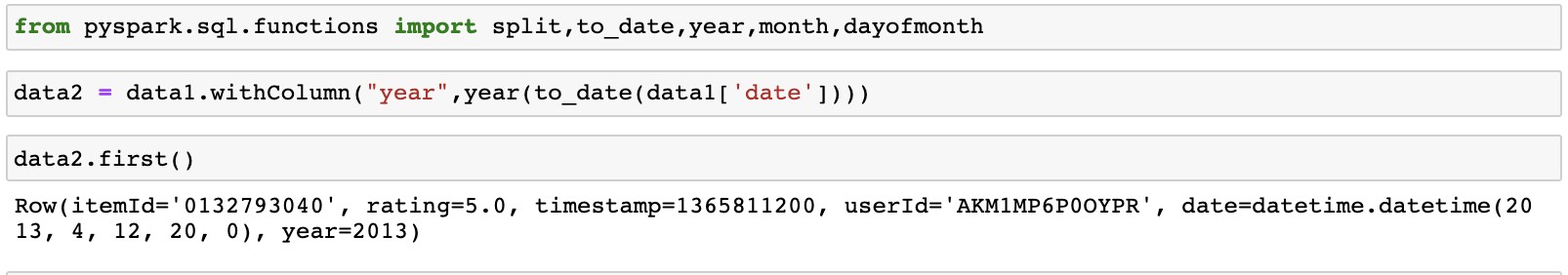
Then we imported spark functions, concat, col, lit and to\_timestamp.



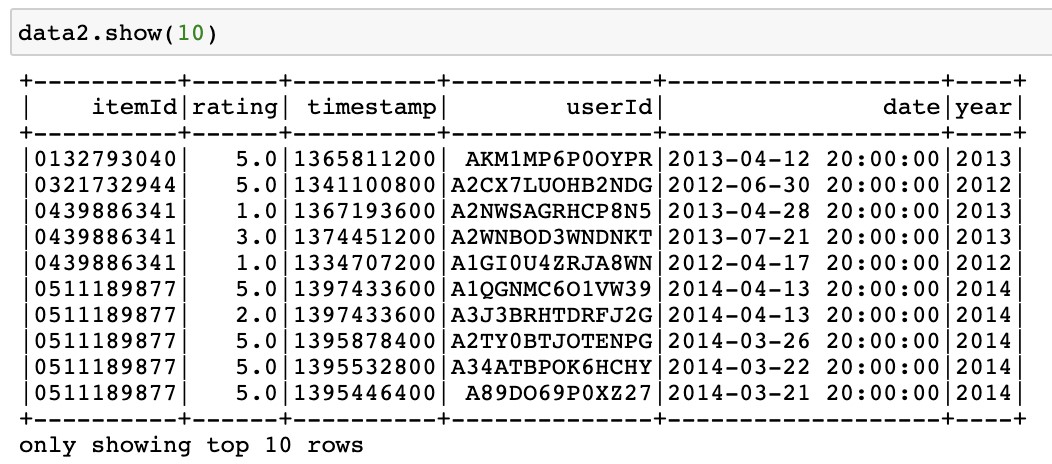
We add one more column which is a human readable form date, and the new dataset is data1.



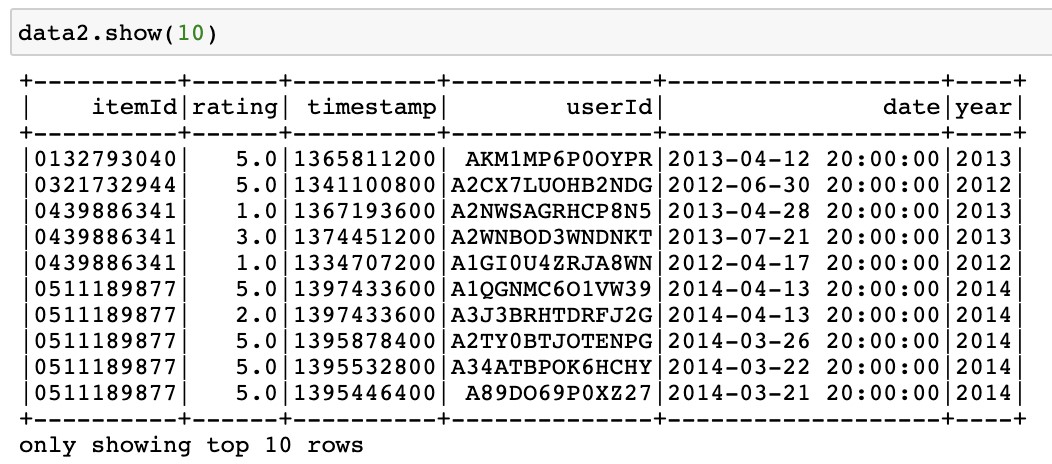
The table was clear and readable, but it might be different to analyze because the readable form data were still complex. We splitted the data into year, month, and day. These steps need new spark functions, so we imported them.



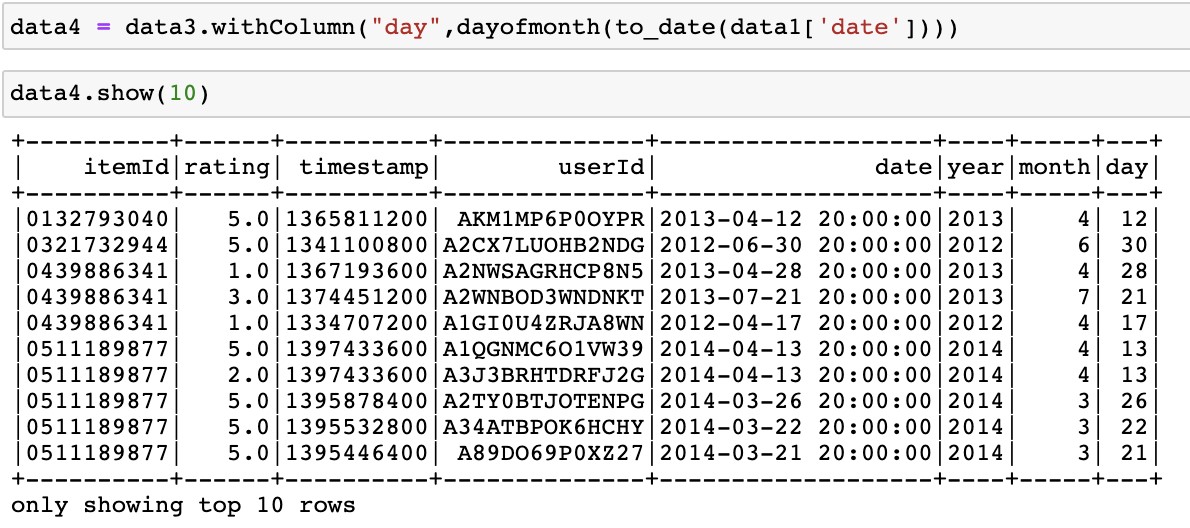
We added the year column first.



Next, we add month columns in the table.



And then, we add the last column which is the day information.



We finished filtering our dataset and downloaded it. The storage of the filtered dataset is 638.1MB

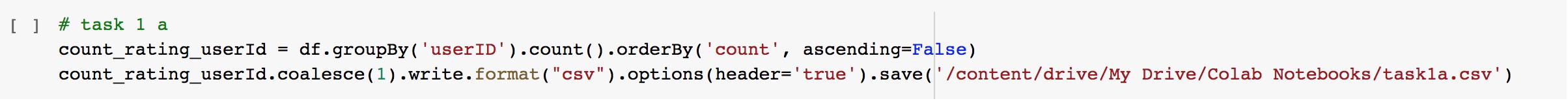
**Task 1 - Exploratory Analysis Tasks**

In this task 1, there are seven parts. We worked on each part below, have shown code, displayed output screenshots and individual graphs to do analysis. We used google colabaratory notebook to perform each task. Google colabaratory is very much easy in running, viewing and saving results. We found it as a great platform to run multiple libraries at one place with fast processing speed. We put the main csv file in the google drive. We linked its drive path into the google colab. notebook and then also saved the result in the csv format in the drive. This tool also generates errors if something goes wrong, and we used the Stack Overflow website to check error solutions when we encountered errors.

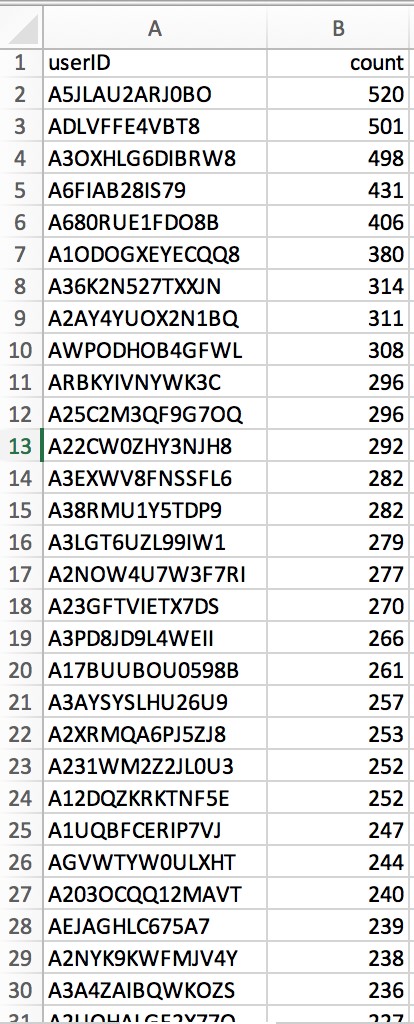
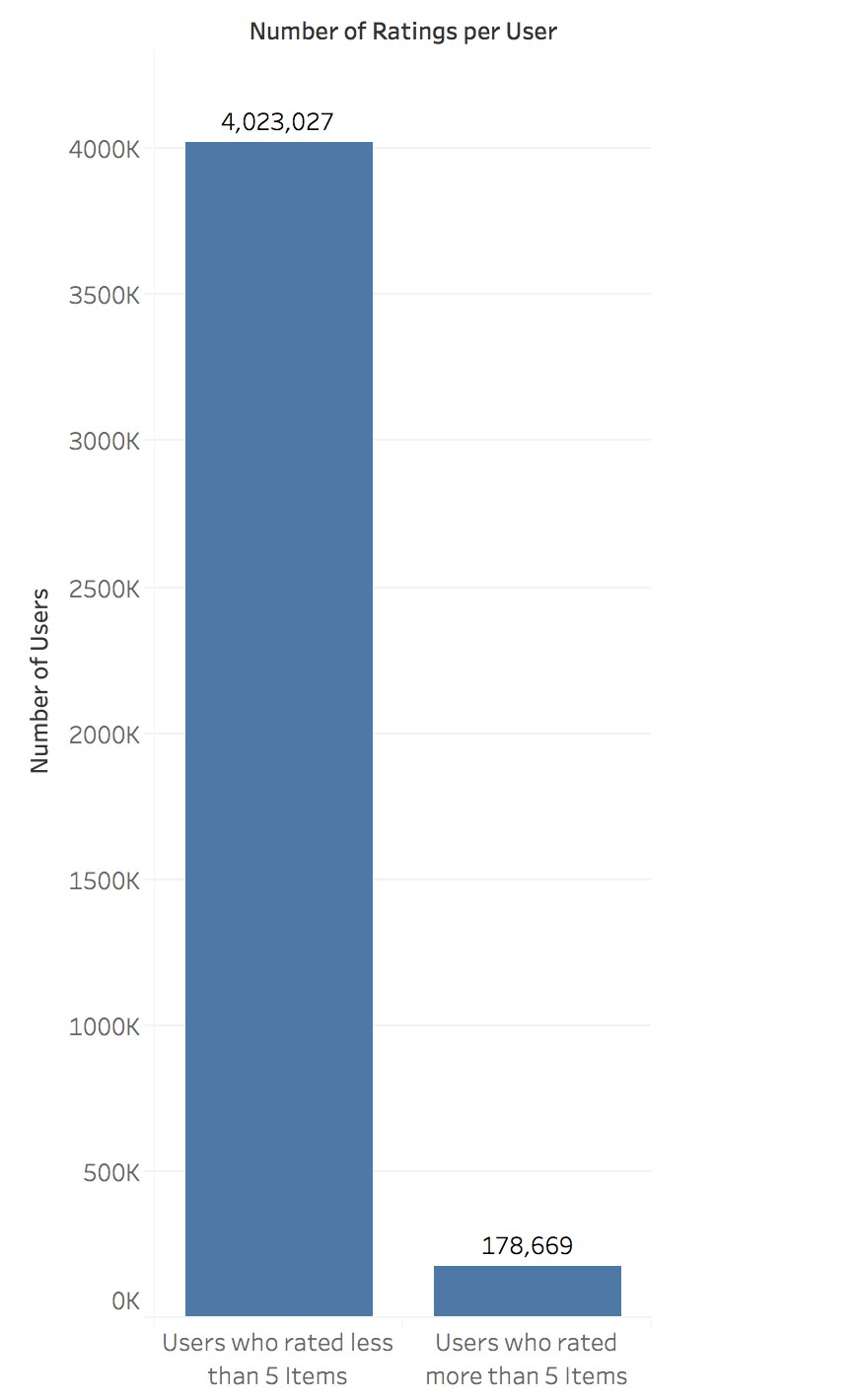
It took around 15 hours for data importing and the analysis of all the parts b,c,d,e and f. We used graphs to demonstrate the analysis done. We got the graphs from Tableau software.

**Part 1 a.** Users vs number of ratings per user: do most users have a few or a lot of ratings?

-The first parts involve finding the relationship between user and number of ratings. In order to do this task, we tried to group userID from the csv file and then we counted their ratings using the following code.



The code runs successfully and it generates the following output in the google drive’s location. The output file shows the userId and the number of reviews given by each user. Spark makes it easier to analyze the given part of the task. It just took 2-3 seconds to run the code, and storing the output file automatically in google drive takes around 10-15 seconds. The output size of file for task 1 a is 70 mb in csv format.

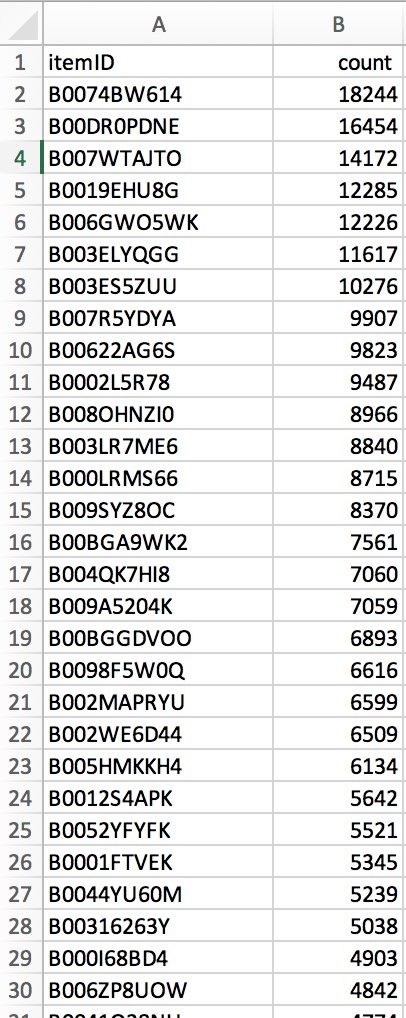
 

**Fig:** Output Data **Fig:** Data Representation in bar diagram

We chose the cutoff to be “5” to group the users if they have less or more number of ratings. On analysing the graph, we found that 178,669 users have given more than 5 ratings while 4,023,027 users have given less than 5 ratings. That means maximum people preferred giving less rating. This task shows behavior of users on giving ratings.

**Task 1 b.** Items vs number of ratings per item: do most items have a few or a lot of ratings?

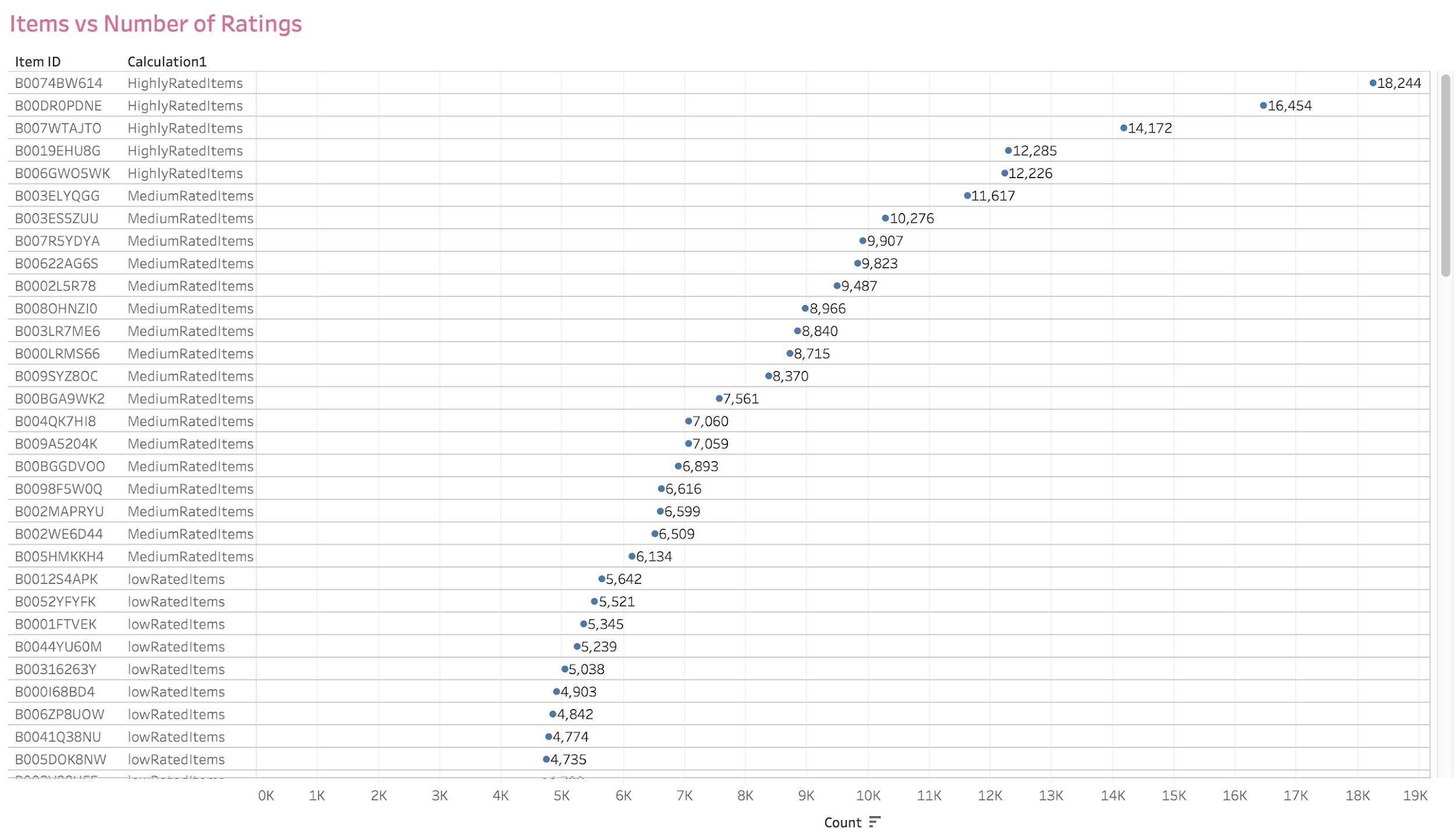
This is another interesting task. In this task, we need to understand the relationship between ratings and the number of items. This task is also a good way to figure out the popularity of the item. For this task also, we used google colabaratory notebook to perform the task.



**Fig:** Output Data in Excel Format

It took 2 seconds to run the code, 8 seconds for output generation and 6 mb output file size. The following is code used to perform Task 1 b.





**Fig:** Prediction for Item Level of Rating

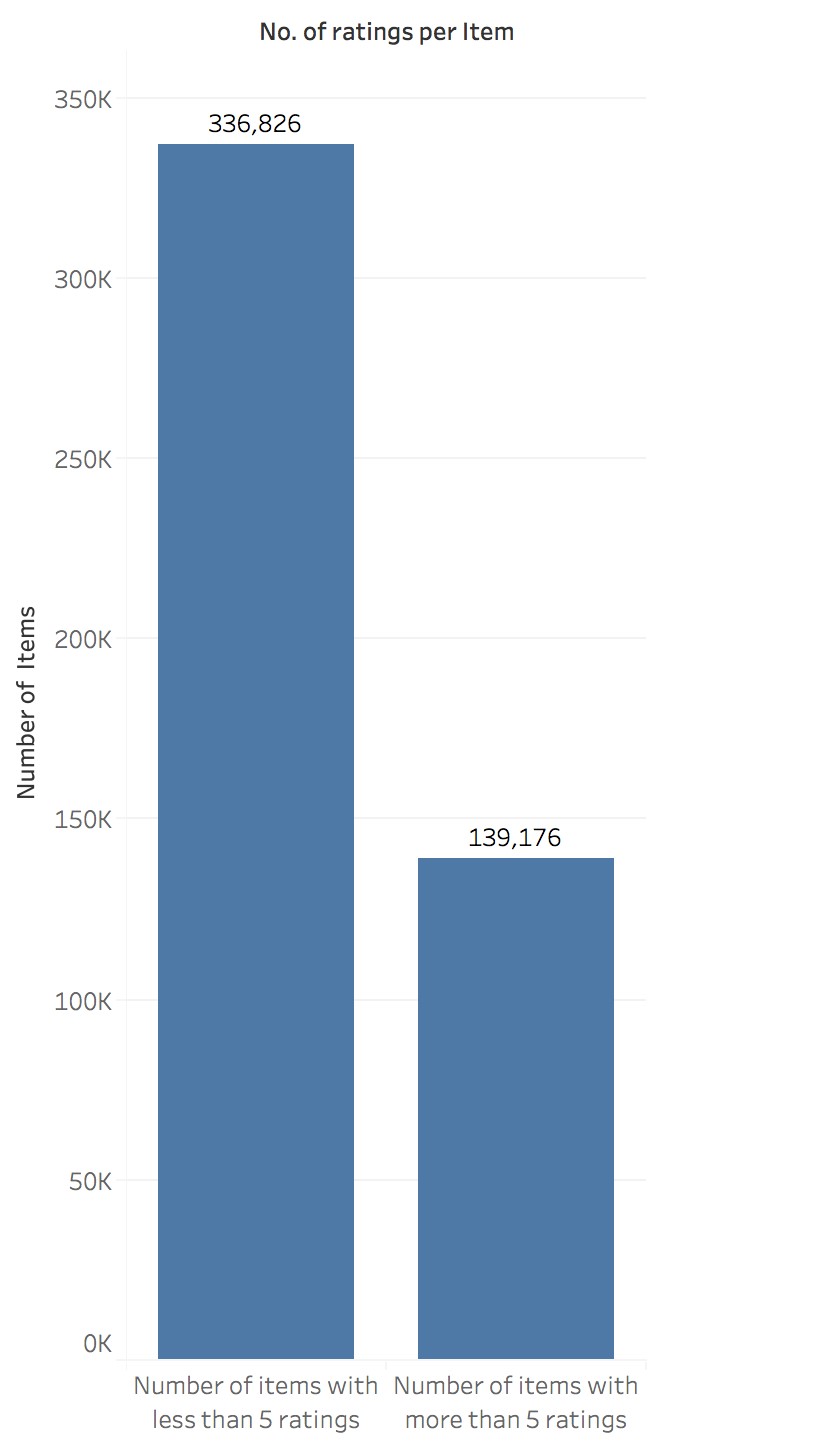
We divided the data into 3 groups

1. **Highly Rated Items**- The number of ratings per item range between 12k and 19k. There are around 5 products grouped as Highly Rated Items.
2. **Medium Rated Items**- The number of ratings per item range between 6k and

12k. These products are in considerable amounts compared to high rated items.

1. **Low Rated Items**- The number of ratings per item range between 0k and 6k. The remaining products other than high rated and medium rated comes under Low rated items.

There should be a specific cutoff to group products or items for most of the items that have a low or more number of ratings. Hence we chose the cutoff to be 5 ratings.



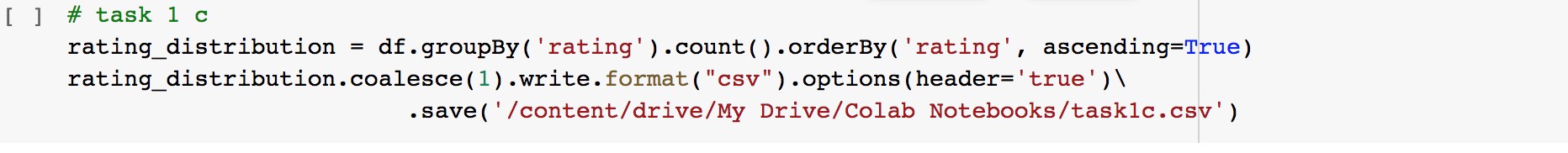
**Fig:** Number of Items Vs Ratings Less or More than 5

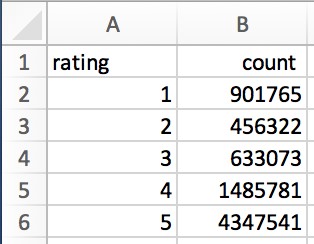
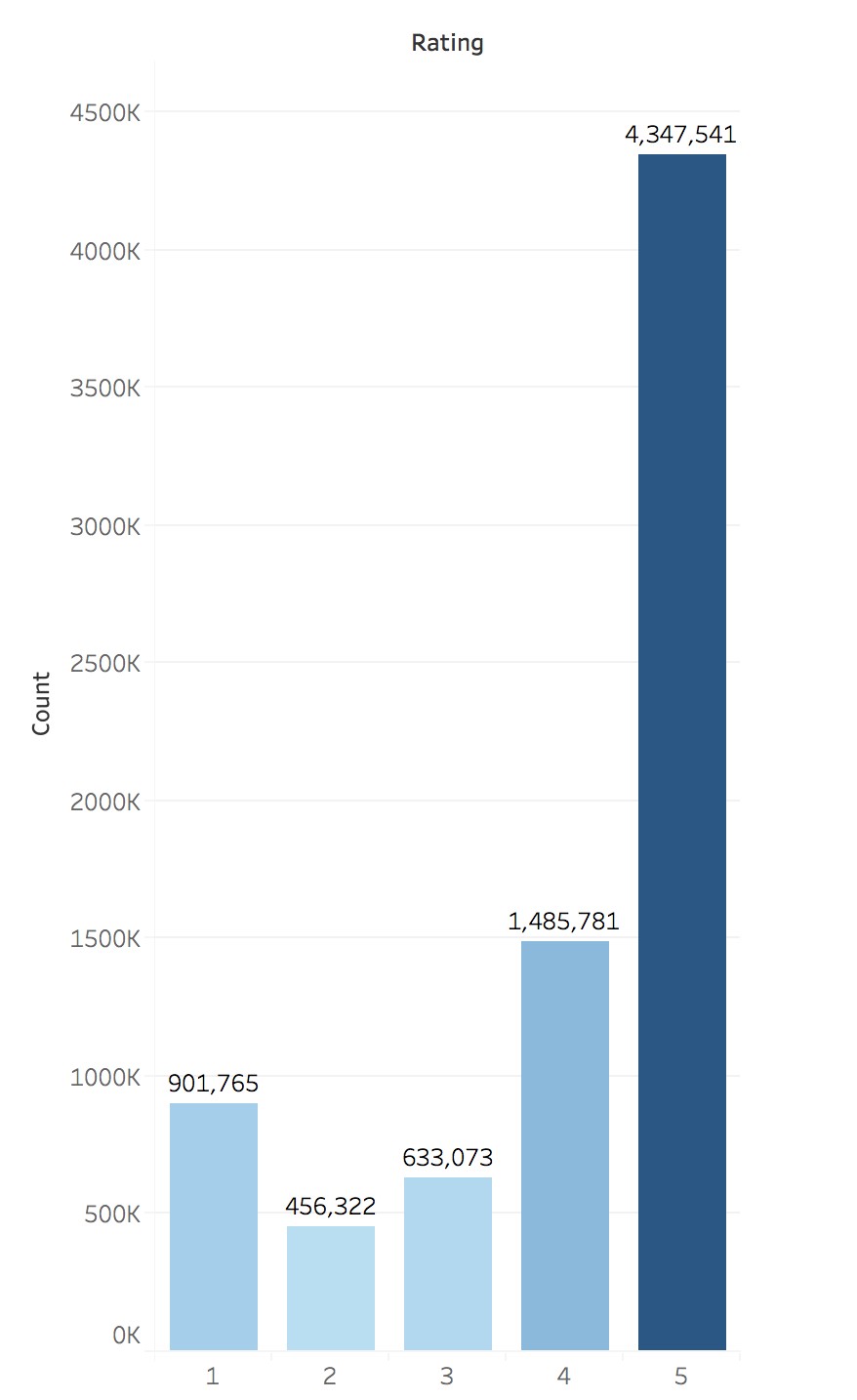
In the given bar diagram above, it shows that 336,826 number of items have ratings less than 5 which are in the range of low rating while 139,176 items have ratings more than 5. This shows the popularity of the item. There were so many items sold on amazon but only 139,176 items got more reviews from the user. These 139,176 items got more attention from the users.

**Task 1c.** ratings distribution: how many times does each rating appear?

-This is another task to find out the number of items which got a rating from 1 to 5.

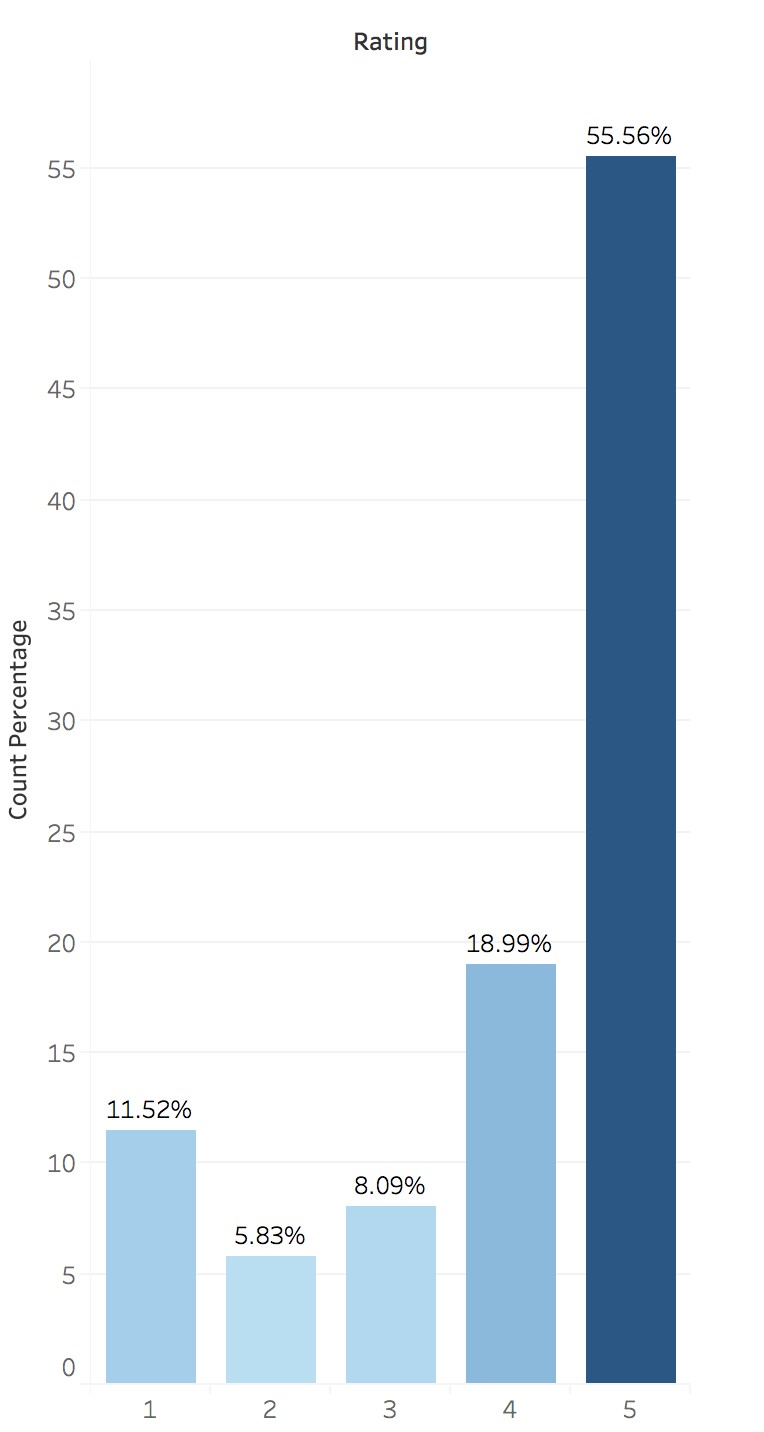
This task let us know which items are the most and least likeable by the users. It took 1.5 second to run, 3 second to generate output in google drive and the size of the output file is 70 bytes.



**Fig:** Output File Screenshot **Fig:** Rating Vs Count Bar Graph

We can see that all the items of this amazon review file have got the reviews. 901,765 items have a review of 1 which is the least reviews given by the user. That means 901,765 items are the least liked by the users. 4,347,541 items have the 5 rated star reviews. That means 4,347,541 items out of 7824482 are the most popular items. Very few people have given the rating 2 while many people have given 5 ratings.

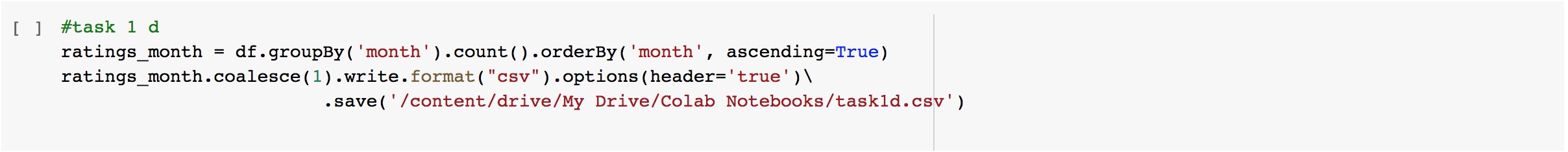


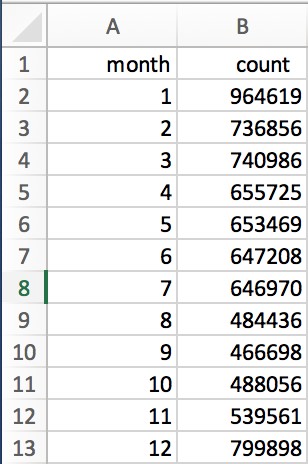
**Fig:** Percentage distribution of Ratings

Considering the percentage distribution of ratings, we could see 55.56% for Rating5 which has a high percentage relatively. The lowest distribution is for Rating-2 of 5.83%.

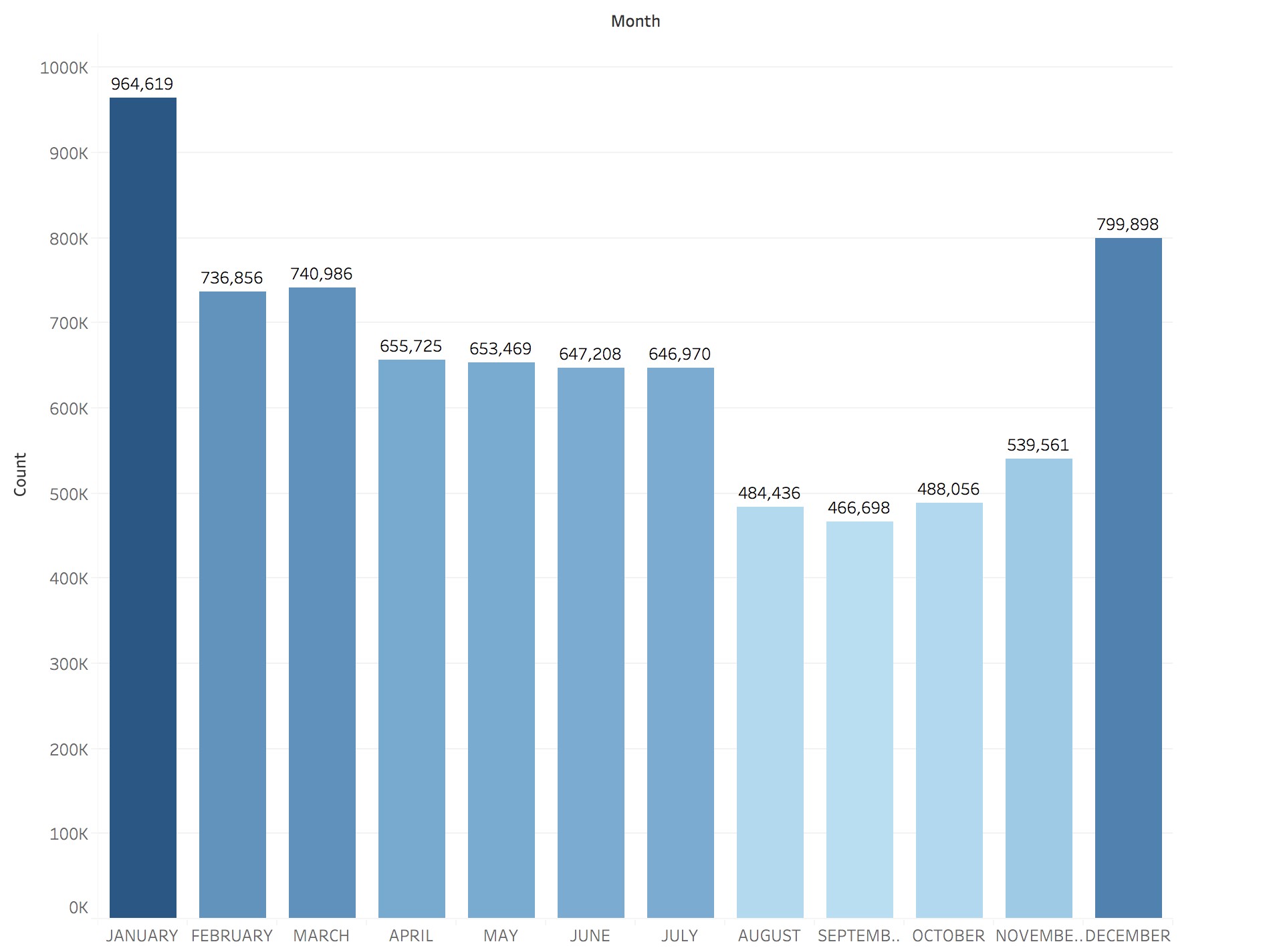
**Task 1d.** time vs ratings: do certain months have more ratings?

-For this task, we took months as a time factor for analysis. We counted the number of ratings given in each month. 1 to 12 represents January to December month. It took 2 seconds to run, 4 seconds to store output in google drive and the size of the output file is 123 bytes. We used the following code to run task 1 d.





**Fig:** Output Excel File Showing Month and Number of rating

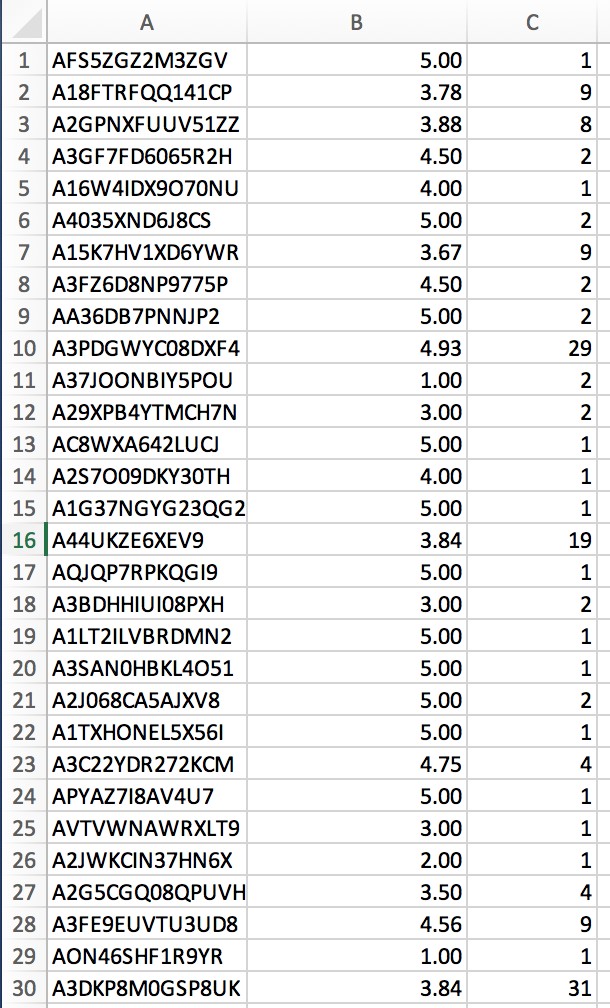


**Fig**: Bar Diagram for Month Vs Number of Rating

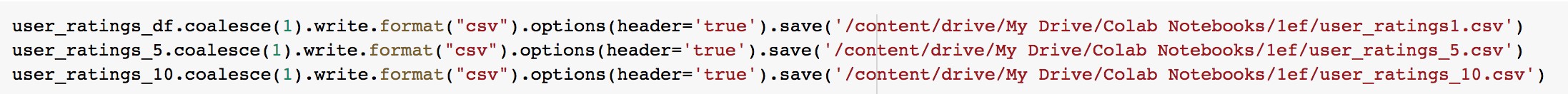
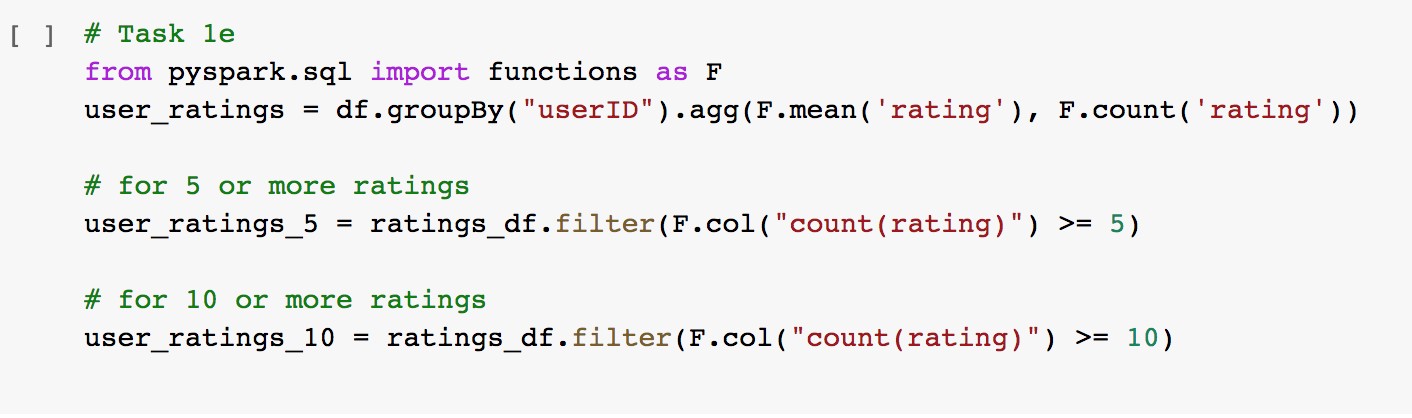
We can see the pattern for the ratings from the given bar diagram. In the month of January, there were the most ratings given, while in February and March, have the most consistent ratings. April, May, June and July have another consistent number of ratings given. September month has the lowest rating given.

**Task 1e**. distribution of average ratings for each user - does it change if you only consider users with 5 or more ratings? 10 or more ratings?

- In this task, we worked to find the relationship between average ratings given by the user and the number of ratings provided. It took 3 seconds to run, 8 seconds to generate an output file in google drive and the size of the output file is 29 mb.



**Fig:** Output Excel file for 1 (e)



Above is the code used to run for task 1 e.

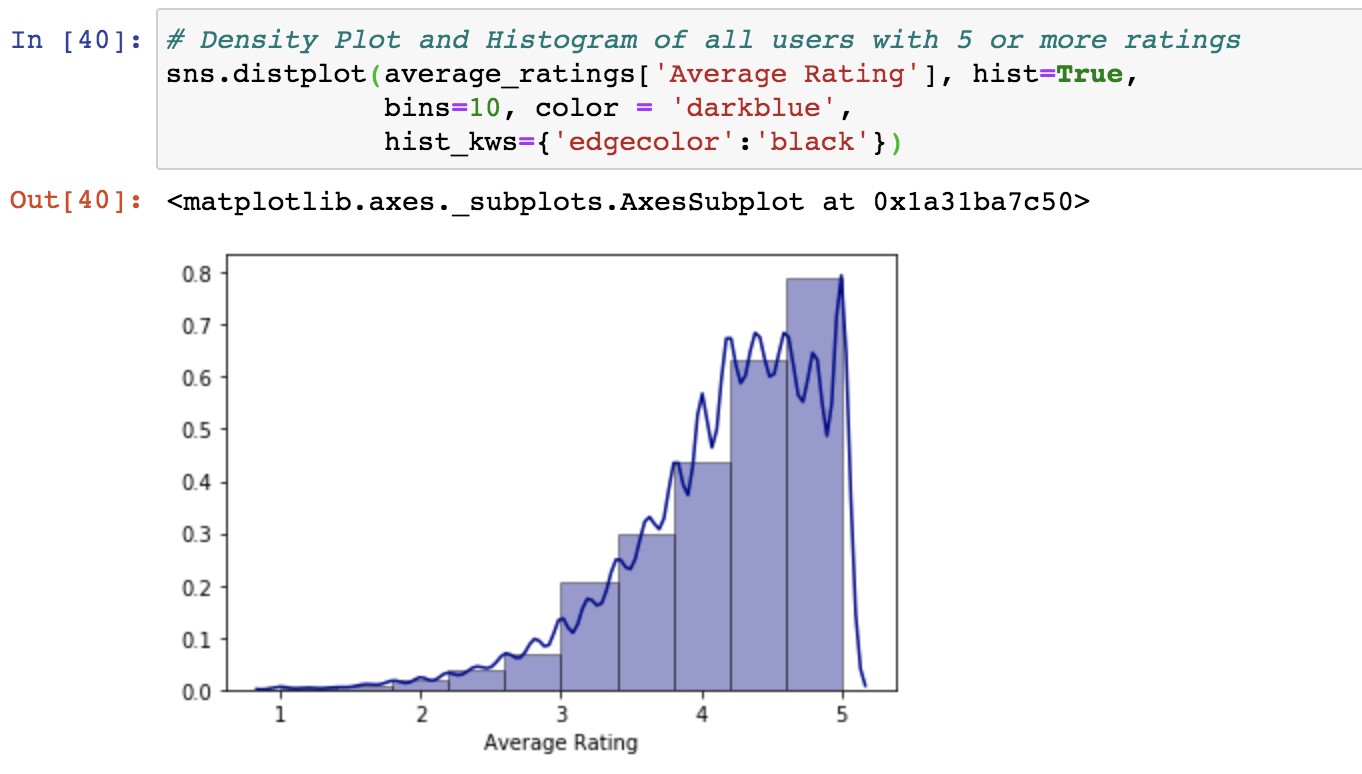
Distribution for average ratings:



**Fig e1**.Distribution of average ratings of users.

From the Fig e1. we could observe that the most of the users are rating the products to be “Excellent” or a 5-star rating. Next highest distribution is for rating-4 where the ratings have reduced 1.5 times by the 5-rating. Lowest Rating given is “2”.

Rating-1 has considerable count here. People rather than giving 2 or 3 ratings, they prefer to give a 1 rating which states “very bad” experience with the product.



**Fig e2**. Distribution of Average rating for users with rating count 5 or more

From the above plot (Fig e2), this is the graph of users which rated 5 or more products. The distribution plot tends to be left skewed as the number of users with 5 or more ratings are with the ratings 4 and 5. The similarity we can observe compared to Fig e1 is that most of the users chose the products to be good or excellent i.e., 4 and 5 Rating. But the distribution is gradually increasing from Rating 1-5 in Fig e. We could see different patterns of distribution at Ratings 1, 2 and 3.

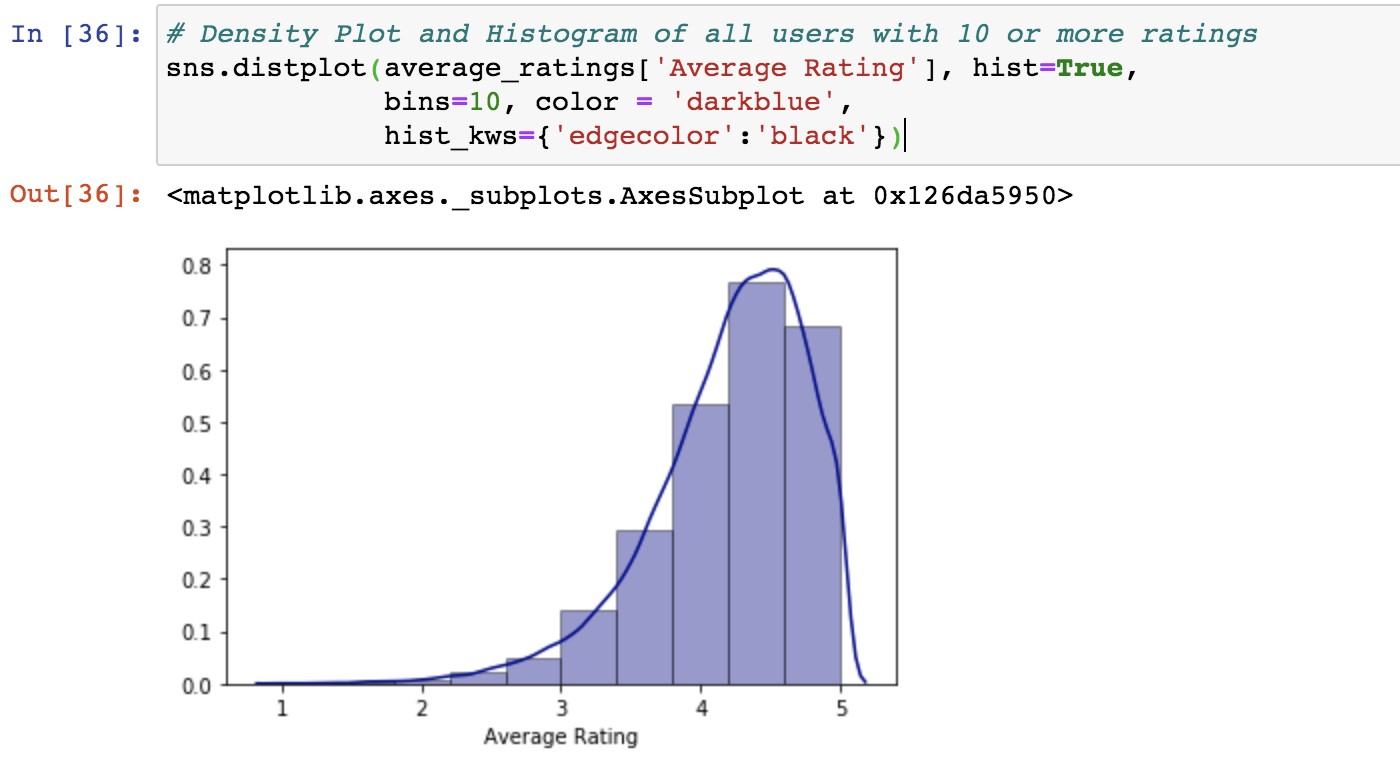
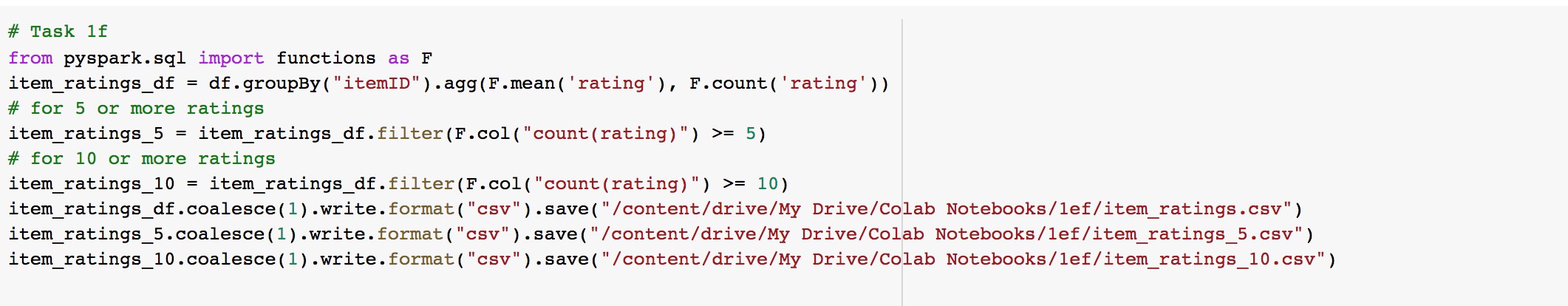


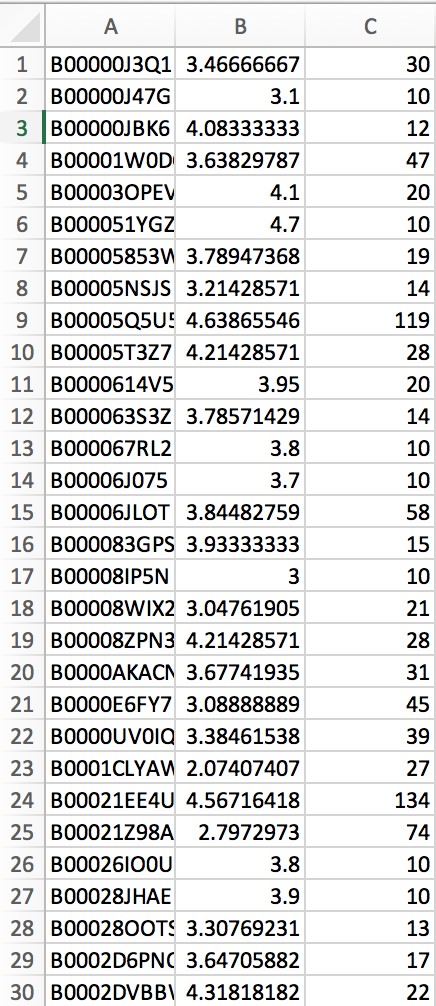
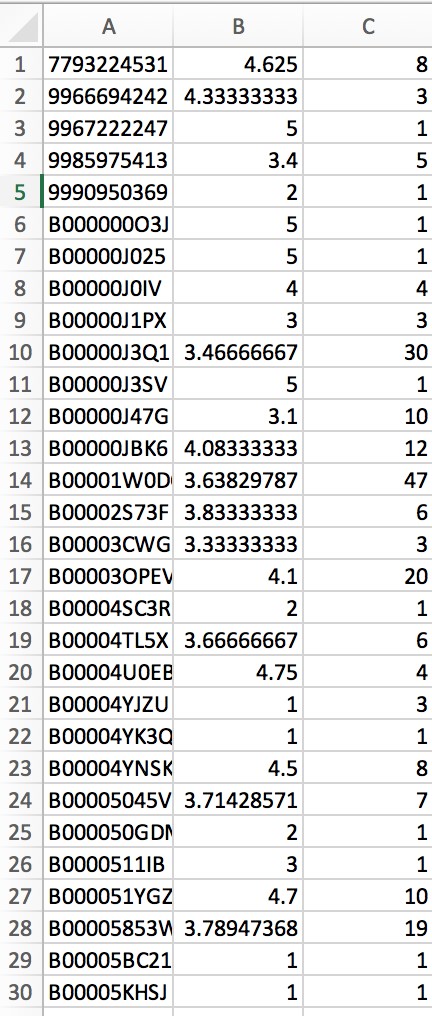
Fig e2. Distribution of Average rating for users with rating count 10 or more

From the third plot, considering users with more than 10 ratings, we can see the most average rating for users with 10 more ratings is around between 4 and 5 Rating. The graph tends to be left skewed as in the case of users with 5 or more ratings. Compared to Fig e1., we could see the similarity that people mostly experience the products to be good or excellent.

**Task 1f.** Distribution of average ratings for each item - does it change if you only consider items with 5 or more ratings? 10 or more ratings?

-In this part, we group the data by item to explore the average ratings. Also, we worked on items with five or more ratings and ten or more ratings and created separate datasets to display the distribution. It took 3 seconds to run and the size of the output file is 17.2 Mb. We worked to find the relation between the item and its average rating to understand the item's popularity.



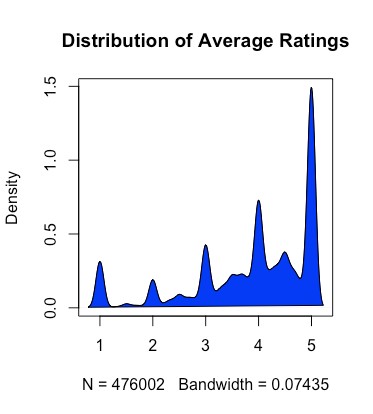


**Fig:** item\_ratings **Fig:**Item\_ratings\_5 **Fig:** item\_ratings\_10

> distribution <- density(average$rating)

> plot(distribution, main="Distribution of Average Ratings")

> polygon(distribution, col="blue", border="black")

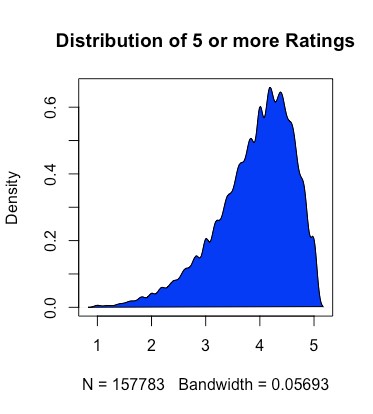


**Fig**：Distribution for Average Ratings

> d1 <- density(more\_than\_5$rating)

> plot(d1, main="Distribution of 5 or more Ratings")

> polygon(d1, col="blue", border="black")

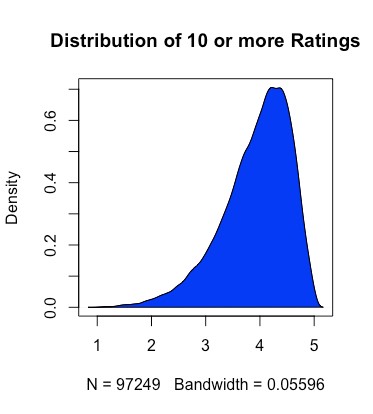


**Fig**：Distribution for 5 or more Ratings

> d2 <- density(more\_than\_10$rating)

> plot(d2, main="Distribution of 10 or more Ratings")

> polygon(d2, col="blue", border="black")



**Fig**：Distribution for 10 or more Ratings

Based on the first plot, we can see that most average ratings for each item is around 5, which is about two times than items with an average rating of 4. Some items have the average rating of 3 or 4. Only a few items have an average rating of 1 and 2.

From the second plot, it shows the distribution of items with 5 or more ratings. It’ s clear that most average ratings for items are mostly between 4 and 5.

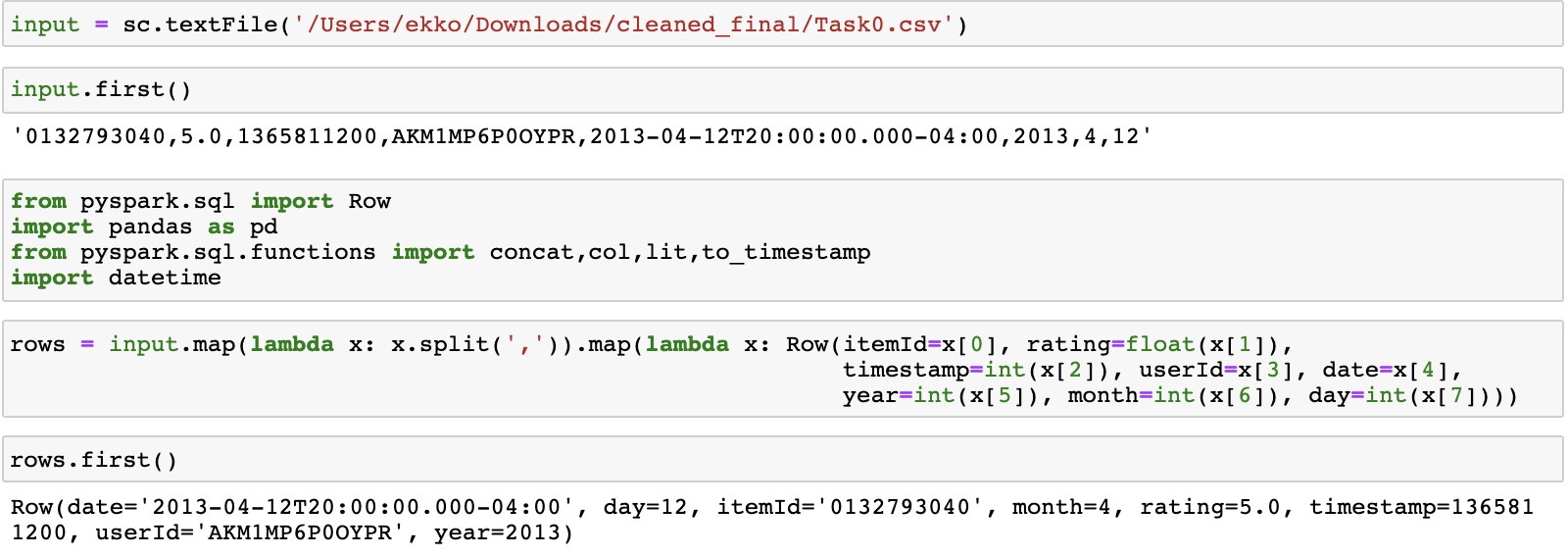
The last plot displays the distribution of items with 10 or more ratings. We can see the most average ratings for items are around between 4 and 5. This distribution plot, as shown, is similar to the distribution of items with 5 or more ratings. Moreover, it’s more skewed towards the high level of ratings.

Through the comparison of these three distributions, we found that the average rating of each item shows an increasing trend with the more times an item is rated. Therefore, we can get the conclusion that users tend to buy the items with a high rating and then they are more likely to give high ratings after use.

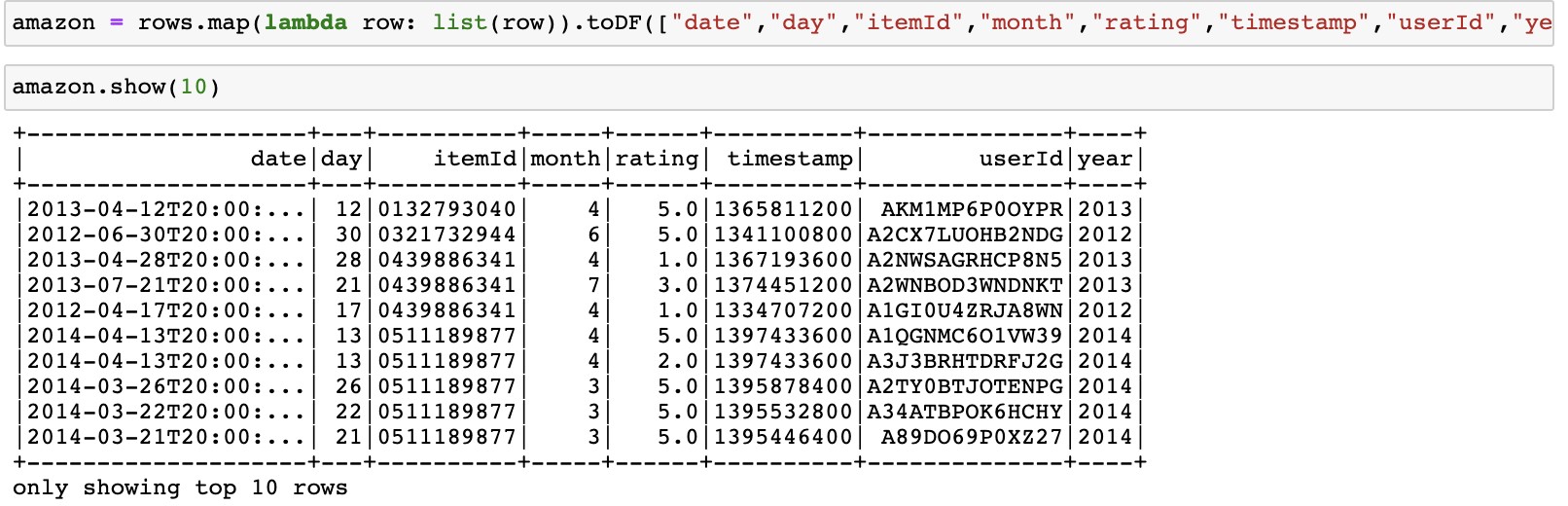
**Task 2 - Analysis Tasks Part 1:**

In this analysis, we want to know if the date will affect people’s rating. It’s possible to get a good rating for an item on the weekend because people will be happier to receive products on the unbusy time. The importing dataset step was going well, and this analysis took one afternoon to finish it.

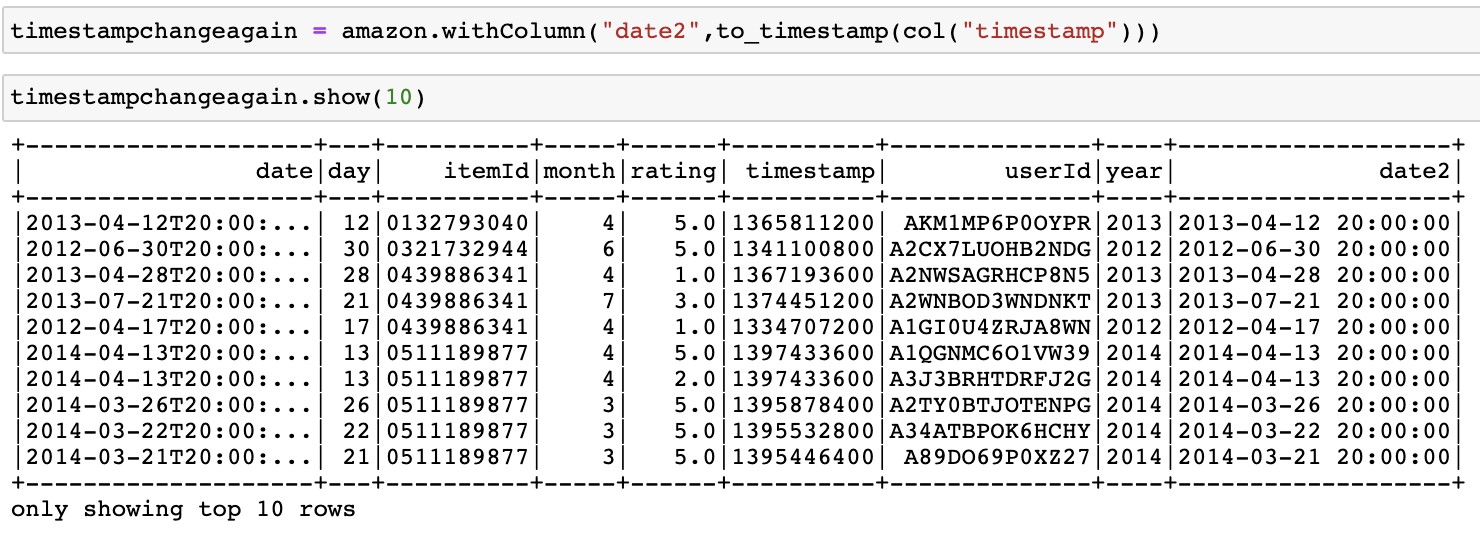
We first read the filtered dataset and named columns.



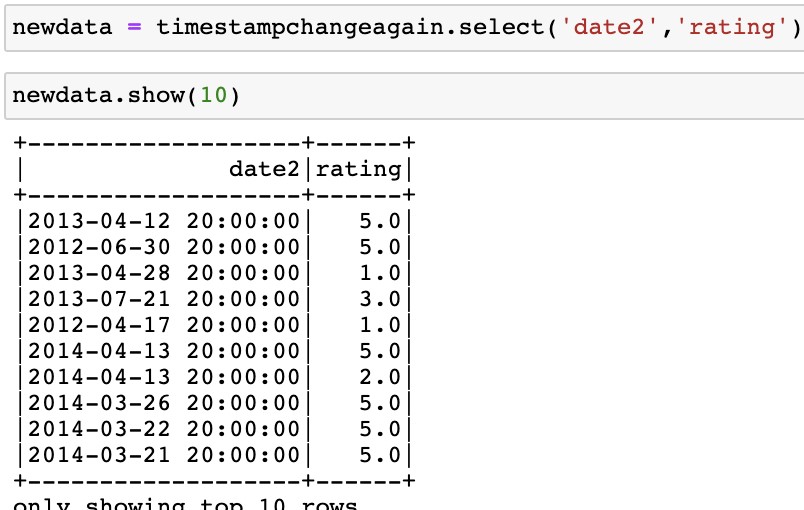
Then, we transformed the dataset to dataframe.



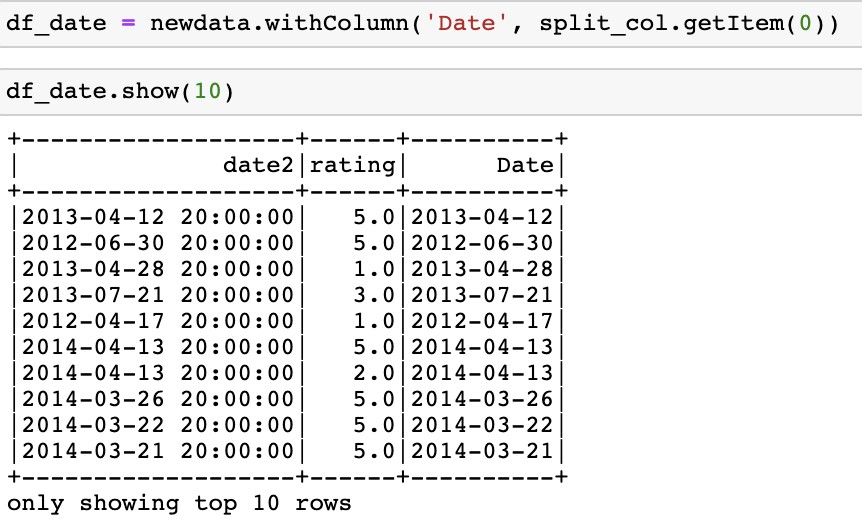
Although we have a clear month column, a day column, and a year column, it’s weird to see “T” between date information and time information in the date column.



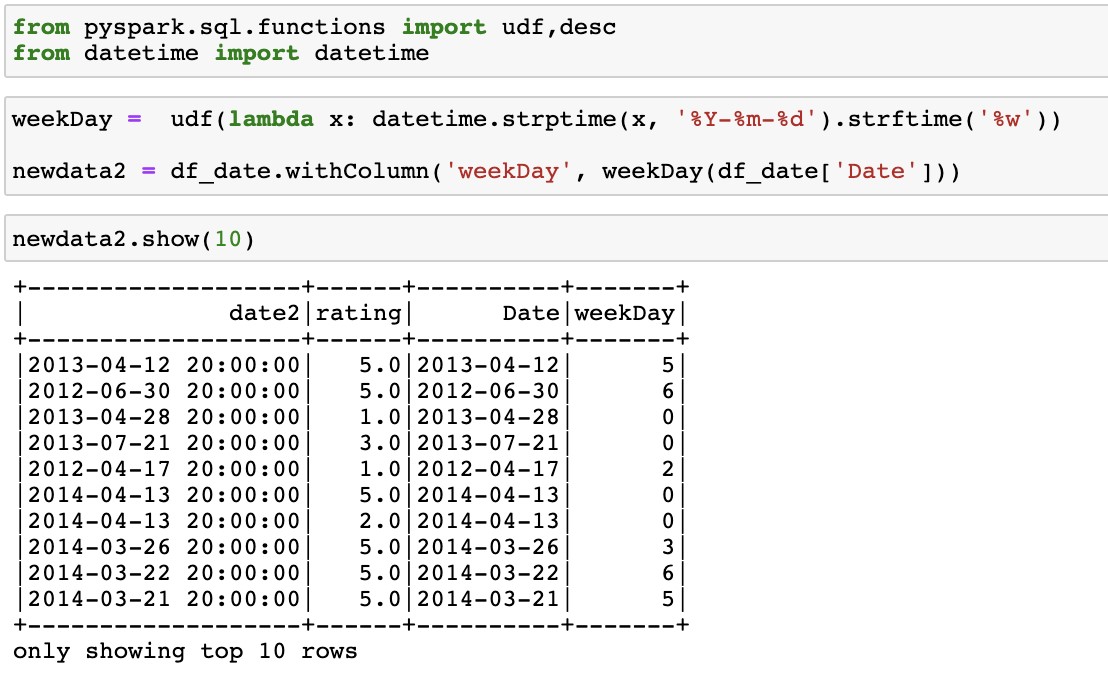
We converted the timestamps again in order to get a new dataframe which only contained data information and rating information.



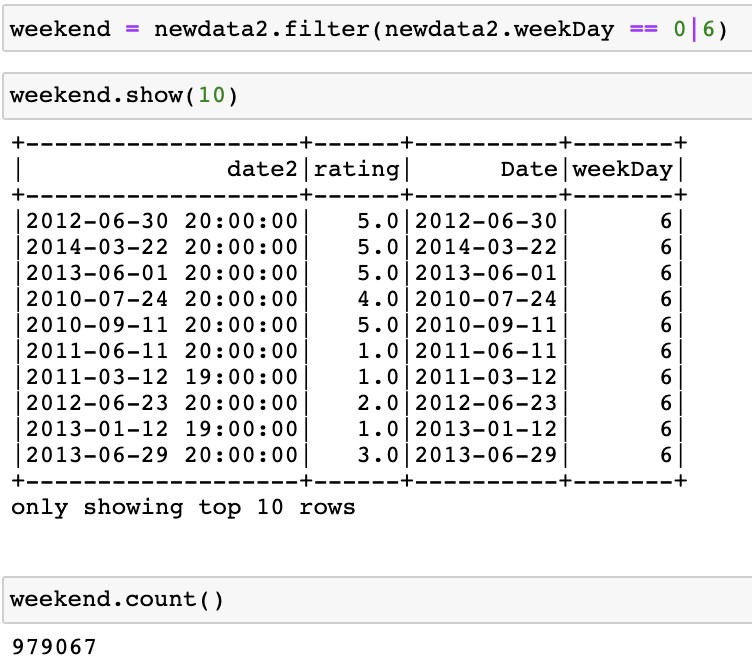
We splitted the data2 column in order to get the column which only contained date information.



Added a new column which contained the weekday information. In this case, 0 stands for Sunday, 1 stands for Monday, 2 stands for Tuesday, 3 stands for Wednesday, 4 stands for Thursday, 5 stands for Friday, and 6 stands for Saturday.



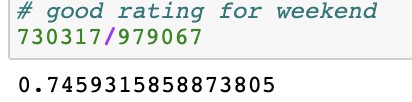
Then we filtered the dataset. We wanted these rows which only contained the weekends information, and counted the number of rows.



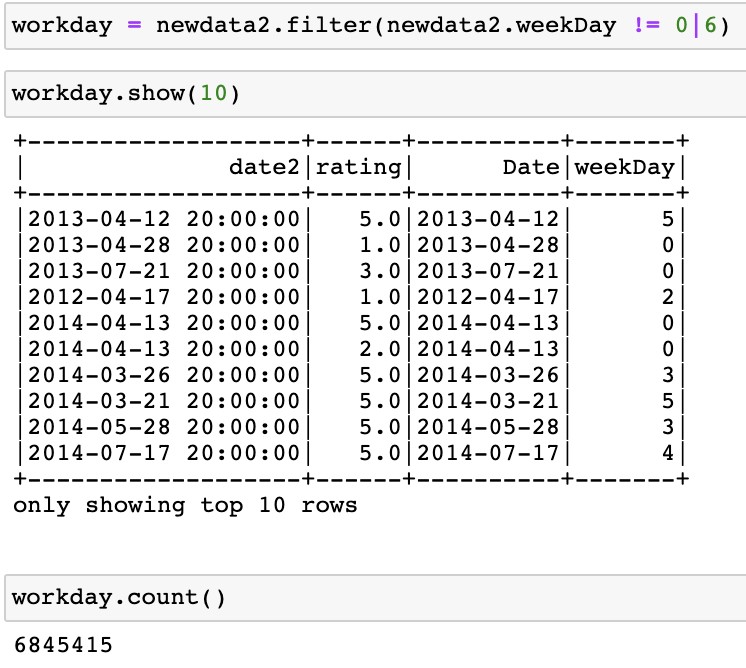
Next, we filtered the rating column. We wanted the rating column to only contain the rating which was larger than 3, and count the number of rows.

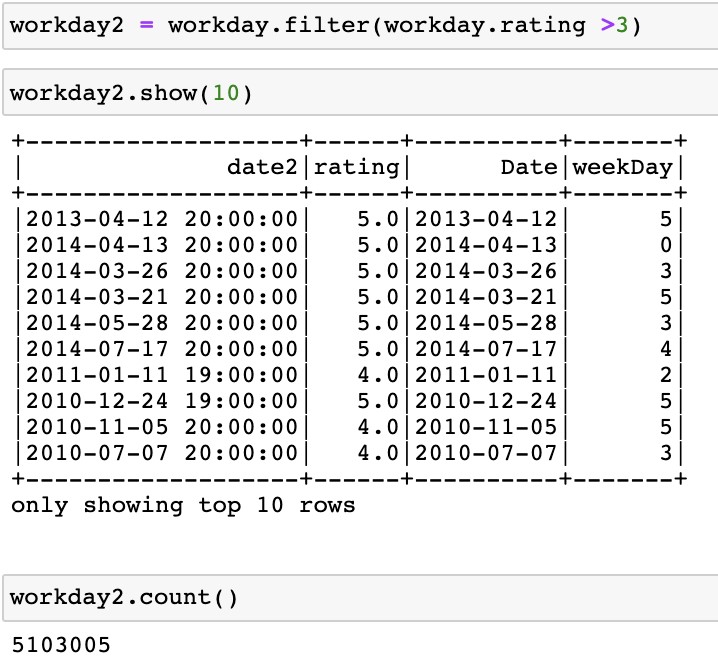


The rate of good rating on weekends is 74.593%



We did a similar step to get the rate of good rating on workdays.





The rate of good rating on work days is 74.546%

Conclusion

The percentages of good ratings for weekends and work days are just slightly different. People may tend to give a good rating on weekends. Thus, we do not have enough evidence to support that item will be rated good on weekends.

# Part 2: Descriptive analysis and Poisson Regression

For this task, we decided to explore the relationship between rating and month. We used the same file as Task 0 and we want to know whether the season affects the item's rating. We used the filtered data set from task 0 and we mainly focused on the month and rating. We first imported the data from the local csv file and then we used the reshape package in R to rename the columns. It took about one and a half minutes to load the data due to the large file size , and it took 25 seconds to rename the columns.

> data <- read.csv("/Users/ccheng/Desktop/Task 0.csv",sep = ",", header=F) > head(data)

> library(reshape)

> rating <- rename(Task.0, c(V1="itemID",V2="rating",V3="timestamp", V4="userID",

V5="date", V6="year", V7="month",V8="day"))



Then we did descriptive statistical analysis of rating in twelve months. Also, we defined the seasons here because we did descriptive statistical analysis of ratings for different seasons in the next step. It took about 12 seconds to run and 1 seconds to get the output.

|  |
| --- |
| > season<- rep(4,length(data$rating))  > season[data$month<=3]<-1  > season[data$month>=4&data$month<=6]<-2  > season[data$month>=7&data$month<=9]<-3  > install.packages("psych")  > library(psych)  > describe1<- describeBy(data$rating,list(data$Month)) > describe1    Descriptive statistics by group  : 1  vars n mean sd median trimmed mad min max range skew kurtosis se  X1 1 962423 4.05 1.36 5 4.31 0 1 5 4 -1.26 0.21 0  --------------------------------------------------------------------------------- |

: 2

vars n mean sd median trimmed mad min max range skew kurtosis se

X1 1 733160 4.04 1.37 5 4.29 0 1 5 4 -1.24 0.14 0

---------------------------------------------------------------------------------

: 3

vars n mean sd median trimmed mad min max range skew kurtosis se

X1 1 740585 4.03 1.37 5 4.29 0 1 5 4 -1.24 0.14 0

---------------------------------------------------------------------------------

: 4

vars n mean sd median trimmed mad min max range skew kurtosis se

X1 1 653589 4.03 1.37 5 4.28 0 1 5 4 -1.22 0.1 0

---------------------------------------------------------------------------------

: 5

vars n mean sd median trimmed mad min max range skew kurtosis se

X1 1 653162 4.01 1.38 5 4.27 0 1 5 4 -1.2 0.04 0

---------------------------------------------------------------------------------

: 6

vars n mean sd median trimmed mad min max range skew kurtosis se

X1 1 650693 4 1.39 5 4.26 0 1 5 4 -1.19 -0.01 0

---------------------------------------------------------------------------------

: 7

vars n mean sd median trimmed mad min max range skew kurtosis se

X1 1 639205 4.02 1.38 5 4.27 0 1 5 4 -1.21 0.06 0

---------------------------------------------------------------------------------

: 8

vars n mean sd median trimmed mad min max range skew kurtosis se

X1 1 482047 3.98 1.39 5 4.23 0 1 5 4 -1.16 -0.08 0

---------------------------------------------------------------------------------

: 9

vars n mean sd median trimmed mad min max range skew kurtosis se

X1 1 465927 3.98 1.4 5 4.22 0 1 5 4 -1.16 -0.08 0

---------------------------------------------------------------------------------

: 10

vars n mean sd median trimmed mad min max range skew kurtosis se

X1 1 491356 3.99 1.39 5 4.24 0 1 5 4 -1.17 -0.05 0

---------------------------------------------------------------------------------

: 11

vars n mean sd median trimmed mad min max range skew kurtosis se

X1 1 543742 3.98 1.4 5 4.22 0 1 5 4 -1.15 -0.12 0

---------------------------------------------------------------------------------

: 12

vars n mean sd median trimmed mad min max range skew kurtosis se X1 1 808593 4 1.39 5 4.25 0 1 5 4 -1.18 -0.02 0

According to the result, we focused on the average rating of each month. It’s clear that January has the highest average rating and then it shows a decreasing trend. We can see it remains in relatively low rating points from August to November. However, the average rating displays an upward trend in December. Thus, we may think the score is relatively high from December to May, and relatively low from June to November. Overall, there is no significant difference in different months.

Then ,we did descriptive statistical analysis of ratings in different seasons to find if there any seasonal effect on the item's rating. It took 7 seconds to run and 1 seconds to get the output.

>describe2<-describeBy(data$rating,list(season))

>describe2

Descriptive statistics by group

: 1

vars n mean sd median trimmed mad min max range skew kurtosis se

X1 1 2436168 4.01 1.38 5 4.27 0 1 5 4 -1.21 0.04 0

---------------------------------------------------------------------------------

: 2

vars n mean sd median trimmed mad min max range skew kurtosis se

X1 1 1957444 4.02 1.38 5 4.27 0 1 5 4 -1.21 0.05 0

---------------------------------------------------------------------------------

: 3

vars n mean sd median trimmed mad min max range skew kurtosis se

X1 1 1587179 4.01 1.38 5 4.26 0 1 5 4 -1.2 0.03 0

---------------------------------------------------------------------------------

: 4

vars n mean sd median trimmed mad min max range skew kurtosis se X1 1 1843691 4.01 1.38 5 4.26 0 1 5 4 -1.2 0.03 0

Based on the result, we can see the average ratings are the same in spring, autumn and winter. It is higher in summer, but the kurtosis in summer is also a little high. It means that the increase of mean is caused by the extreme outliers. Combining the result of rating in 12 months, we regard there is no seasonal effect on the item’s rating.

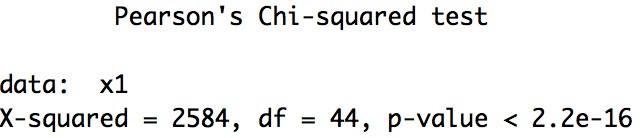
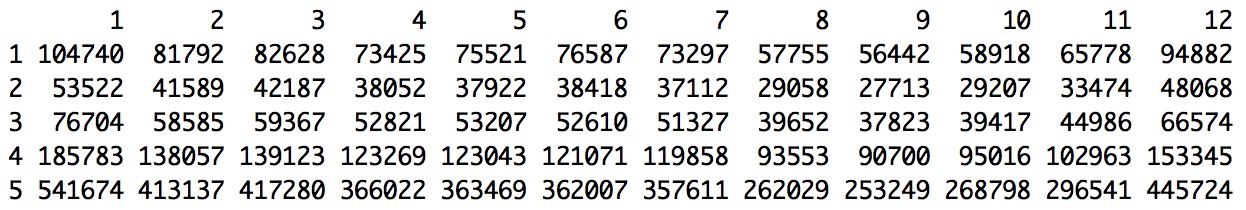
The first part is to use the descriptive analysis to find the correlation. In this step, we used the test to check the correlation. We analyzed the 12-month rating in the contingency table and did the Chi square test. It took 5 seconds to run the contingency table and 1 seconds to get the result. The Chi square test was much faster, it only took 1 seconds to run.

> x1<-table(data$rating,data$month)

> x1

> result\_chi1<-chisq.test(x1)

> result\_chi1



Here we can see the p value of Chi square test is less than 0.05, so we believe there is no significant relationship between rating and month. In order to prove our thought, we also analyzed the four season’s rating in the contingency table and did the Chi square test.

|  |
| --- |
| > x2<-table(data$rating,season)  > x2 season  1 2 3 4   1. 269160 225533 187494 219578 2. 137298 114392 93883 110749 3. 194656 158638 128802 150977 4. 462963 367383 304111 351324 5. 1372091 1091498 872889 1011063     > result\_chi2<-chisq.test(x2)  > result\_chi2    Pearson's Chi-squared test    data: x2  X-squared = 1772.8, df = 12, p-value < 2.2e-16 |

From the contingency table, there is no obvious difference between four seasons. We can see users with a maximum rating of 5 during the all seasons. The overall performance of the item's rating within the high level. From the Chi-squared test, the p value is less than 0.05, so there is not enough evidence to say there is seasonal effect on the item’s rating.

Then we started to build the Poisson regression model to validate our conclusion. The first model used to find the relationship between rating and different seasons. It took 35 seconds to run and 3 seconds to get the output.

|  |
| --- |
| > model <- glm(formula = data$rating~season, family =poisson(link="identity") , data = data)  > summary(model)  Call: glm(formula = data$rating ~ season, family = poisson(link = "identity"), data = data)    Deviance Residuals:  Min 1Q Median 3Q Max  -1.8113 -0.5159 0.4627 0.4715 0.4892    Coefficients:  Estimate Std. Error z value Pr(>|z|)  (Intercept) 4.0525990 0.0016375 2474.87 <2e-16 \*\*\* season -0.0170403 0.0006218 -27.41 <2e-16 \*\*\*  ---  Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1    (Dispersion parameter for poisson family taken to be 1)    Null deviance: 4649743 on 7824481 degrees of freedom  Residual deviance: 4648992 on 7824480 degrees of freedom AIC: 29523537    Number of Fisher Scoring iterations: 3 |

Based on the output, it’s able to see the estimated model is rating = -0.017season +4.053. We can see the coefficient of the month is negative, which means the ratings will decrease from spring to winter. However, it's really small and the p-value of this model is less than 0.05, so there is no significant evidence to say different seasons have an impact on the item’s rating.

The second model used to find the relationship between rating and different months. It took 35 seconds to run and 3.5 seconds to get the output.

|  |
| --- |
| > model2<- glm(formula = data$rating~data$month, family = poisson(link="identity"), data = data)  > summary(model2)    Call:  glm(formula = data$rating ~ data$month, family = poisson(link = "identity"), data = data)    Deviance Residuals:  Min 1Q Median 3Q Max  -1.8135 -0.5134 0.4599 0.4744 0.4920    Coefficients:  Estimate Std. Error z value Pr(>|z|)  (Intercept) 4.0464577 0.0013997 2891.02 <2e-16 \*\*\* data$month -0.0056139 0.0001973 -28.46 <2e-16 \*\*\*  ---  Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1    (Dispersion parameter for poisson family taken to be 1)    Null deviance: 4649743 on 7824481 degrees of freedom  Residual deviance: 4648933 on 7824480 degrees of freedom  AIC: 29523478    Number of Fisher Scoring iterations: 3 |

Based on the output, it’s able to see the estimated model is rating = -0.006month +4.046. The p-value of this model is less than 0.05, so there is no significant evidence to say different months have an impact on the item’s rating.

The third model used to explore the influence by combining the seasonal effect and month effect. It took 36 seconds to run and 5 seconds to get the summary. Actually, it got the same result as the first two models.

> model3<- glm(formula = data$rating~season+data$month, family = poisson(link="identity"),

data = data)

> summary(model3)

Call:

|  |
| --- |
| glm(formula = data$rating ~ season + data$month, family = poisson(link = "identity"), data = data)    Deviance Residuals:  Min 1Q Median 3Q Max  -1.8138 -0.5130 0.4595 0.4751 0.4923    Coefficients:  Estimate Std. Error z value Pr(>|z|)  (Intercept) 4.044682 0.001926 2100.277 < 2e-16 \*\*\* season 0.003659 0.002726 1.342 < 2e-16 \*\*\* data$month -0.006744 0.000865 -7.797 6.35e-15 \*\*\*  ---  Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1    (Dispersion parameter for poisson family taken to be 1)    Null deviance: 4649743 on 7824481 degrees of freedom  Residual deviance: 4648931 on 7824479 degrees of freedom AIC: 29523478    Number of Fisher Scoring iterations: 3 |

Conclusion:

Based on these three methods, we found that there is no obvious relationship between rating and months. Also, there is no significant seasonal effect on the item’s rating. Therefore, we can say that sales might be affected by time, but it doesn’t have much influence on the item’s rating .

**Part 3 : Recommending products based on User ratings using Cluster**

**Analysis**

In this open-ended analysis, we are concerned with recommending products to a user based on the purchase history and similarity of ratings provided by other users. For this, we used a python Jupyter notebook to analyze the data. we downloaded the data provided in the project pdf with a link given below.

“http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/ratings\_Electronics.” It just took 10 min in importing the data from the link. After importing, the dataset size was about 326mb. As it is a .csv file, we could only use only 1048576 records out of around 7800000 records for analysis in our local system. This is the problem we faced when importing the dataset. This took around 5 hours for the analysis.

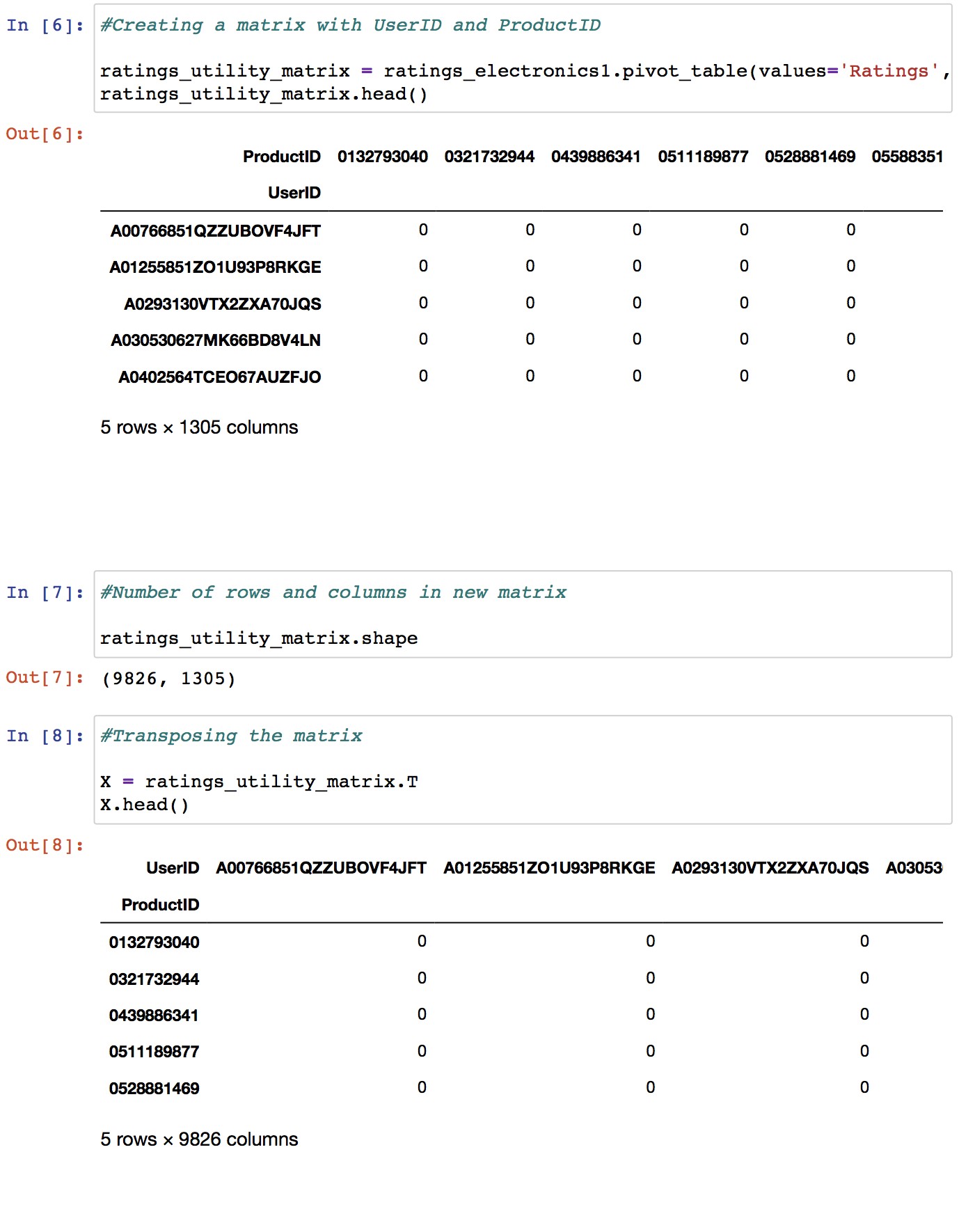
Companies like Amazon, help users discover relevant and new items and give recommendations to the customers to increase user experience while driving incremental revenue. This system helps users by suggesting the list of items that suits the interest of the user from which they can select the right one. Recommender systems personalize customer experience by understanding their usage of the system and recommending items they would find useful.

The dataset has 4 attributes with “UserID”, “ProductID”,”Ratings” and “Timestamp”. For recommender system analysis, we need 3 columns except “Timestamp” since we are dealing with recommendation of products based on user ratings. There are around 7824482 users with respective ratings of different products.

First, we are importing a few python packages like NumPy, pandas, matplotlib and sklearn. Considering the top 10,000 observations of the data and based on the preferences or ratings from multiple customer data, predicting products for a specific user by designing the pattern of analysis.



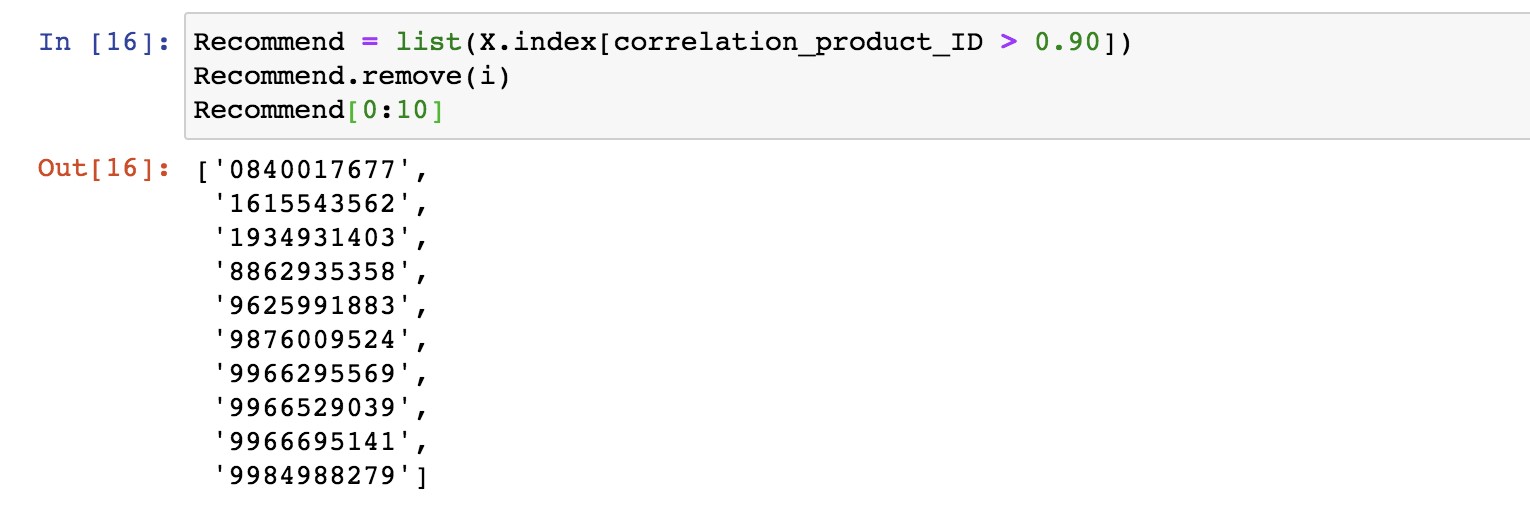
We are creating the utility matrix with user and corresponding products where the matrix gives the ratings provided. Every user may go rate every product available. Hence most of the values are null as appear in the output 6 below. We transpose the matrix after that and the shape of the matrix is known. This returns 1305 and 9826 records.



Considering 10 products to be recommended for a product chosen. We apply reduction in dimension to 10 records.



We assumed that a user purchases a product ID starting with “134048160x” with an index 50. Correlation matrix is observed here which can give us the correlation coefficient between the variables like user id and products id in our case study.



**Conclusion:**

The output above shows the top 10 recommended products with reference to their productID’s. The main idea in the recommendation system with multiple users is considering the correlation. If the correlation between different products is greater than 0.90, we consider those products to be recommended for a particular user.

# Summary Unanticipated Problem Faced

## Task 0

The first unanticipated problem was on the terminal. We first tried to do the task on the terminal, but it was not stable. We restarted the terminal several times, it still had problems with connecting to the java server. We gave up running codes on the terminal because he could not import to\_timestamp spark function on the terminal. It took several hours to install java, spark, Jupyter notebook. The second unanticipated problem occurred when I tried to download the filtered data. He tried the “. write.csv” function to download the filtered data into csv. However, it came out with 10 csv files. we got a solution from google. Function “coalesce (1)” helped him to solve the problem. Secondly, we tried to use his mac terminal to do it, but our laptop used to get spacing problems. He switched to a windows laptop to try again to do Task 0. We needed to download all the required installs. Also, the windows was taking time to update it software. It took us 6 hours to just set up the running environment. We later tried Google Collaboratory Notebook to run, looking on canvas and following its tutorial online. We were able to do part 0 in this Google colab. Notebook. It took us around three hours to do Part 0 there.

## Task 1

We did coding part for part 1 a, b, c, d, e and f. We used google colab Notebook to do. We encountered a couple of errors like Py4JJavaError, Analysis Exception, <ipython\_input-11-0984 error, ModuleNotFoundError, and Name Error maximum time. One needs to be very careful running in a google colabaratory environment. It needs authorization code submitted. For that, one needs to sign in using google account, allow permission to access and then copy the code from the newly opened page and then need to submit it on google colab. page for verification. The other code only runs once it gets validated. Otherwise, though your code for part 1 a to f right, but they show error in running until it gets verified. It took us a few hours checking the errors solution and configuring it to run each part successfully. It took us around 8 hours to write code and finish the task 1 coding part. Though codes take a couple of seconds to run successfully or generate the error message, one needs to define all the functions and variables to move further. We analyzed the task 1 a, b, c, d, e and f parts with graphs. We used Tableau, Jupyter python and RStudio for analysis.

**Task 2-1**

This task was pretty smooth, and no unanticipated problems occurred.

## Task 2-2

At first, we tried to do the time series analysis for this part and draw the time series plot to present the change and trends. However, when we created the time series, we found that the data is not continuous and there is duplicate data at the same time period. Then we tried to aggregate monthly ratings from 1998 to 2004 and use it to analyze the correlation. But we found there is still a lack of data due to the discontinuous month and it may not be accurate though time series analysis. Therefore, we decided to do descriptive statistical analysis and build the Poisson regression to explore the research question. Secondly, we found variable error, name not defined errors and exception error while writing and running the code.

## Task 2-3

For the cluster analysis part, we used python code for analysis. This is the first time we tried with python. We anticipated that We would get stuck in between with the code syntax. we took help from google for proper syntax of the code. There were no unanticipated problems that we faced while doing analysis.

# 