SCHOOL OF SCIENCE AND TECHNOLOGY

ASSIGNMENT FOR THE
BSC (HONS) IS (DATA ANALYTICS); YEAR 2

ACADEMIC SESSION APRIL 2021
IST3134: BIG DATA ANALTICS IN THE CLOUD

DEADLINE: 14 July 2021

STUDENT NAME:	**SHAMALAN RAJESVARAN**	**STUDENT ID: 18042945**
STUDENT NAME:	**NG WEI XIANG**	**STUDENT ID: 18033167**

INSTRUCTIONS TO CANDIDATES

- This assignment will contribute 20% to your final grade.

IMPORTANT

The University requires students to adhere to submission deadlines for any form of assessment. Penalties are applied in relation to unauthorized late submission of work.

- Coursework submitted after the deadline but within 1 week will be accepted for a maximum mark of 40%.
- Work handed in following the extension of 1 week after the original deadline will be regarded as a non-submission and marked zero.

Lecturer's Remark (Use additional sheet if required)

We …... (Name) ………..…………………........std. ID received the assignment and read the comments ....……………………...……… (Signature/date)

Academic Honesty Acknowledgement

SHAMALAN RAJESVARAN	**18042945**
NG WEI XIANG	**18033167**

"We ……….. (student name) verify that this paper contains entirely our own work. We have not consulted with any outside person or materials other than what was specified (an interviewee, for example) in the assignment or the syllabus requirements. Further, We have not copied or inadvertently copied ideas, sentences, or paragraphs from another student. We realize the penalties *(refer to page 16, 5.5, Appendix 2, page 44 of the student handbook diploma and undergraduate programme)* for any kind of copying or collaboration on any assignment."

[illegible] Xiang **[12 JULY 2021]**

…………………………………….... (Student's signature / Date) …………………………….....

INTRODUCTION TO THE PROBLEM

When it comes to big data, we would want to ascertain as to the powers of big data analytics in the cloud. Our method of analysis is to go about understanding the exact difference between running analysis virtually on the cloud compared to locally. We were told and taught constantly about the power of running on the cloud because virtual machines are much more powerful than our own machines. However, this was something that we had very little knowledge and experience in. Hence, this analysis proved useful as we were able to learn more into the powers of big data analytics in the cloud.

In order to identify the problem that we would want to solve, we first conducted extensive research into the various datasets that we can analyze.

We had ultimately chosen to go with the Amazon Product Data particularly with regards to the Review on the books.

Upon choosing the dataset that we wanted, we begun work so as to identify the main Research Questions that we would want answered with regards to the dataset. The purpose of these Research Questions is to allow us to identify the problem that we would want to solve and analyse. Our way around it is to conduct some form of analysis on the dataset. We then could study the output and provide our justifications. Despite the dataset being large in entries (rows), it is insufficient in the features (columns).

The **Research Topics** that we have identified are as follows:

- **Research Topic 1**: To compare the number of books in each rating group.
- **Research Topic 2**: To identify the number of reviews done per user.

Our approach to analyzing this dataset would be to analyze the dataset on different systems and machines. We varied our approach to analyze the dataset. We decided to utilize MapReduce on the AWS Console. This would allow us access to a myriad of facilities and features within the AWS platform. We have also chosen to utilize MapReduce as the dataset that we have chosen is large and is suitable for analysis on MapReduce. Furthermore, we also decided to run the analysis on our personal computer.

Our **analysis methods** are as follows:

- Analysis using Hive Query language on the Hadoop system (Running on the Virtual Machine through Amazon Web Services)
- Analysis using the Python language for data queries (Running on Windows 10 local personal computer)

With these research topics present, we can finally move on and really continue solving our main problem. Our method of analysis would be to look into the time taken for both the separate systems to run the queries that we have requested based on **Research Question 1** and **Research Question 2**. This would help us in best understanding the inner workings of the big data analytics in the cloud as well as on a local machine.

INTRODUCTION TO THE DATASET

The dataset that we have chosen was an Amazon Product dataset. This dataset was containing the product reviews and metadata from Amazon which includes 142.8 million reviews spanning from May 1996 to July 2014. We had elected to utilize the “Small” Subset for experimentation as there was no metadata or reviews but only the “User”, “Item”, “Rating” and “Timestamp” tuples. The Amazon Product data site can be accessed here. This site contains the ratings left by different users on Amazon about different categories of products. As part of this assignment, we have chosen to analyze the **Amazon Books Review Dataset** which can be obtained here. We have chosen this dataset as it was the largest dataset.

The Amazon Books Review Dataset that we have chosen consist of 22, 507, 155 ratings. This dataset fulfills the requirement for it is to be sufficiently large to be processed in the cloud with a minimum of 2GB large with an actual size of this file being 2GB.

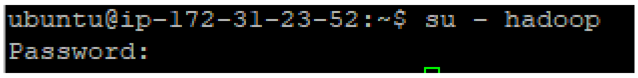
As part of our research when searching for a suitable dataset based on our requirement, we also wanted it to be interesting to research onto. There are 4 columns, namingly "Item ID", "Book ID", "Rating" and "Timestamp". The rating is a categorical variable ranging from 1 to 5 only.

This dataset contains the ratings left by different users on products in the Amazon platform. The dataset was 4.4GB large and consists of 51 million entries in total. With our interest on the books dataset only, the final dataset size will be scaled down to 2gb only.

EXPLANATION OF THE MAPREDUCE APPROACH

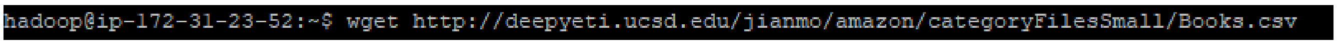
1. Analysis using Hive Query language on the Hadoop system
(Running on the Virtual Machine through Amazon Web Services)

Firstly, we have to initiate Hadoop. This process is done by inputting *"su -hadoop"* in the terminal. We then had to input the password which is" MapReduce*"*. We would have known that Hadoop has been initiated based on the start of the line.



```
ubuntu@ip-172-31-23-52:~$ su - hadoop
Password:
```

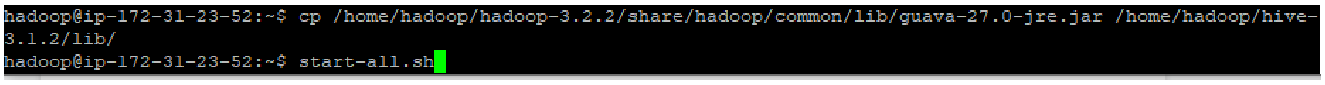
Once, we have confirmed that Hadoop has been initiated (Line starts with Hadoop@ip-xxx), we then had to proceed to download the csv file. Since, we are running Hadoop on a remote machine, we are unable to upload or direct the dataset from our home directory to the remote machine (due to disk usage quota). Hence, our way around it is utilize the *"wget"* command which is a non-interactive network downloader. "wget" is utilized to download files from the server when the user has not logged on to the system. It can work in the background without hindering the current process.



```
hadoop@ip-172-31-23-52:~$ wget http://deepyeti.ucsd.edu/jianmo/amazon/categoryFilesSmall/Books.csv
```

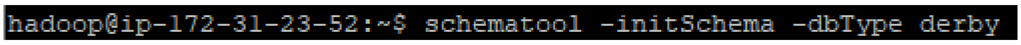
Hence, utilizing the *"wget"* and the link to the dataset. Once the command has run, the dataset has been loaded.

Next, we started the services utilizing the command *"start-all.sh"*. This command starts all Hadoop in a single node cluster. This command will startup a namenode, datanode, jobtracker and a tasktracker in the machine.



```
hadoop@ip-172-31-23-52:~$ cp /home/hadoop/hadoop-3.2.2/share/hadoop/common/lib/guava-27.0-jre.jar /home/hadoop/hive-
3.1.2/lib/
hadoop@ip-172-31-23-52:~$ start-all.sh
```

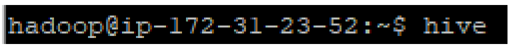
This is followed by initializing the Hive Metastore database schema. The command to go about this is by utitilizing the command *"schematool -initSchema -dbType derby"*. This initialization to a current scheme is done using the schematool provided by Hive *(The database type within the Hive Metastore database schema can be* ***derby, mysql, oracle, mssql, or postgres****)*. In this case, we would be utilizing the *"derby"* database type.



```
hadoop@ip-172-31-23-52:~$ schematool -initSchema -dbType derby
```

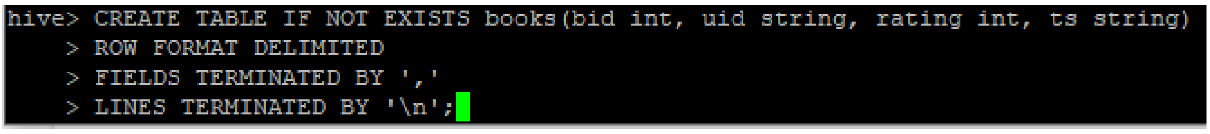
This process would ultimately initiate the schema.

We then started the Hive setup by inputting “hive” in the command line.



```
hadoop@ip-172-31-23-52:~$ hive
```

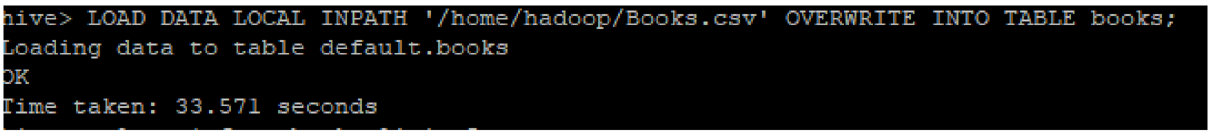
We then create the table which would best represent and suit the Amazon Books Review Dataset. As mentioned earlier, the dataset chosen has 4 variables, “User”, “Item”, “Rating” and “Timestamp”. Hence, this would allow us to have the data inputted into the table based on its variables.



```
hive> CREATE TABLE IF NOT EXISTS books(bid int, uid string, rating int, ts string)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > LINES TERMINATED BY '\n';
```

The line *“ROW FORMAT DELIMITED”* indicated that whenever a new line is encountered the new record entry will start. The *“ FIELDS TERMINATED BY ‘,’ ”* line shows that we are utilizing the *“,”* delimiter to separate each column. Once, this query has been completed, the table books has been created.

The next process would be to load the csv file into the created table “books”.



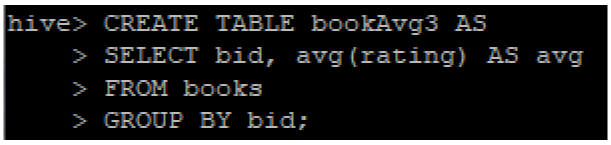
```
hive> LOAD DATA LOCAL INPATH '/home/hadoop/Books.csv' OVERWRITE INTO TABLE books;
Loading data to table default.books
OK
Time taken: 33.571 seconds
```

This process utilizes the syntax of Hive’s *“LOAD DATA”* command. The *“LOCAL”* command indicates that the file is in the server where the beeline is running. Once this process is completed. The dataset that we have initially downloaded using the *“wget”* command would have been loaded to the new table *“books”* that we have created.

We will now proceed with the analysis on the 2 Research Questions.

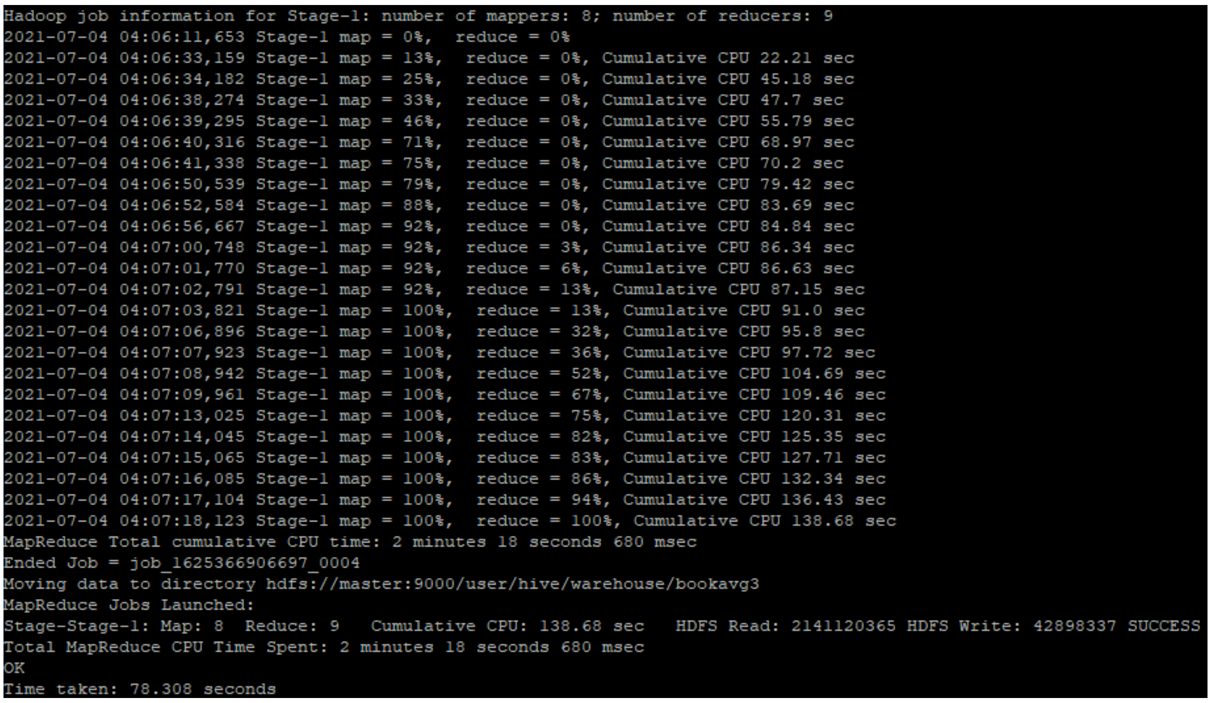
Research Question 1: To compare the number of books in each rating group.

We first ran the query to create a separate table “bookAvg3” so that we can run our queries to satisfy Research Question 1.



```
hive> CREATE TABLE bookAvg3 AS
    > SELECT bid, avg(rating) AS avg
    > FROM books
    > GROUP BY bid;
```

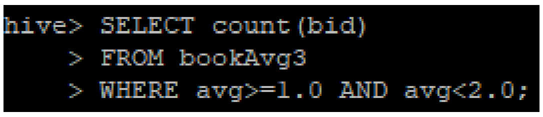
The new table created “bookAvg3” would tally the average of the rating of each specific book. This is done because each book is reviewed many times. Hence, by using this new table, we are able to count the **distinct** number of groups in each rating group.



```
Hadoop job information for Stage-1: number of mappers: 8; number of reducers: 9
2021-07-04 04:06:11,653 Stage-1 map = 0%,  reduce = 0%
2021-07-04 04:06:33,159 Stage-1 map = 13%,  reduce = 0%, Cumulative CPU 22.21 sec
2021-07-04 04:06:34,182 Stage-1 map = 25%,  reduce = 0%, Cumulative CPU 45.18 sec
2021-07-04 04:06:38,274 Stage-1 map = 33%,  reduce = 0%, Cumulative CPU 47.7 sec
2021-07-04 04:06:39,295 Stage-1 map = 46%,  reduce = 0%, Cumulative CPU 55.79 sec
2021-07-04 04:06:40,316 Stage-1 map = 71%,  reduce = 0%, Cumulative CPU 68.97 sec
2021-07-04 04:06:41,338 Stage-1 map = 75%,  reduce = 0%, Cumulative CPU 70.2 sec
2021-07-04 04:06:50,539 Stage-1 map = 79%,  reduce = 0%, Cumulative CPU 79.42 sec
2021-07-04 04:06:52,584 Stage-1 map = 88%,  reduce = 0%, Cumulative CPU 83.69 sec
2021-07-04 04:06:56,667 Stage-1 map = 92%,  reduce = 0%, Cumulative CPU 84.84 sec
2021-07-04 04:07:00,748 Stage-1 map = 92%,  reduce = 3%, Cumulative CPU 86.34 sec
2021-07-04 04:07:01,770 Stage-1 map = 92%,  reduce = 6%, Cumulative CPU 86.63 sec
2021-07-04 04:07:02,791 Stage-1 map = 92%,  reduce = 13%, Cumulative CPU 87.15 sec
2021-07-04 04:07:03,821 Stage-1 map = 100%,  reduce = 13%, Cumulative CPU 91.0 sec
2021-07-04 04:07:06,896 Stage-1 map = 100%,  reduce = 32%, Cumulative CPU 95.8 sec
2021-07-04 04:07:07,923 Stage-1 map = 100%,  reduce = 36%, Cumulative CPU 97.72 sec
2021-07-04 04:07:08,942 Stage-1 map = 100%,  reduce = 52%, Cumulative CPU 104.69 sec
2021-07-04 04:07:09,961 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 109.46 sec
2021-07-04 04:07:13,025 Stage-1 map = 100%,  reduce = 75%, Cumulative CPU 120.31 sec
2021-07-04 04:07:14,045 Stage-1 map = 100%,  reduce = 82%, Cumulative CPU 125.35 sec
2021-07-04 04:07:15,065 Stage-1 map = 100%,  reduce = 83%, Cumulative CPU 127.71 sec
2021-07-04 04:07:16,085 Stage-1 map = 100%,  reduce = 86%, Cumulative CPU 132.34 sec
2021-07-04 04:07:17,104 Stage-1 map = 100%,  reduce = 94%, Cumulative CPU 136.43 sec
2021-07-04 04:07:18,123 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 138.68 sec
MapReduce Total cumulative CPU time: 2 minutes 18 seconds 680 msec
Ended Job = job_1625366906697_0004
Moving data to directory hdfs://master:9000/user/hive/warehouse/bookavg3
MapReduce Jobs Launched:
Stage-Stage-1: Map: 8  Reduce: 9   Cumulative CPU: 138.68 sec   HDFS Read: 2141120365 HDFS Write: 42898337 SUCCESS
Total MapReduce CPU Time Spent: 2 minutes 18 seconds 680 msec
OK
Time taken: 78.308 seconds
```

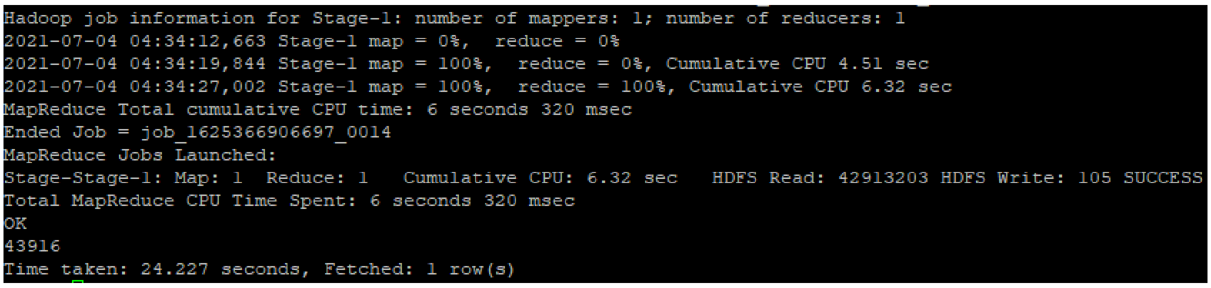
This process takes **78.308 seconds**.

a. **Books with an average rating of 1**



```
hive> SELECT count(bid)
    > FROM bookAvg3
    > WHERE avg>=1.0 AND avg<2.0;
```

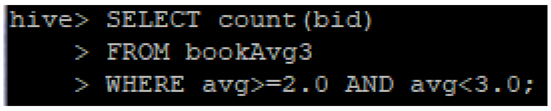
Count: 43916



```
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2021-07-04 04:34:12,663 Stage-1 map = 0%,  reduce = 0%
2021-07-04 04:34:19,844 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.51 sec
2021-07-04 04:34:27,002 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.32 sec
MapReduce Total cumulative CPU time: 6 seconds 320 msec
Ended Job = job_1625366906697_0014
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.32 sec   HDFS Read: 42913203 HDFS Write: 105 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 320 msec
OK
43916
Time taken: 24.227 seconds, Fetched: 1 row(s)
```

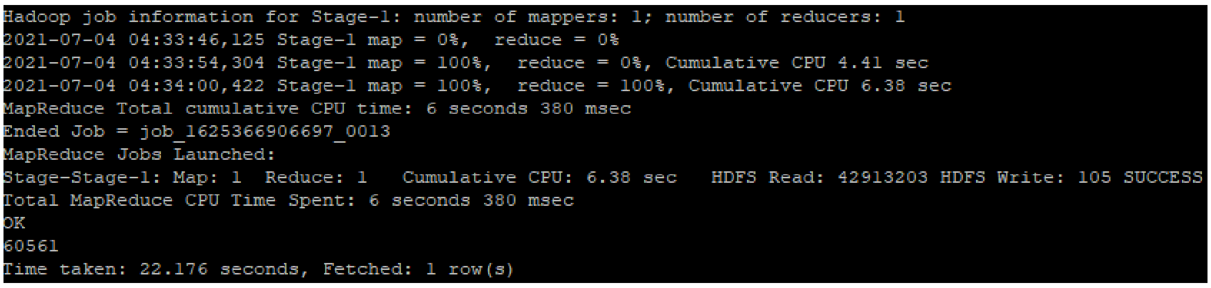
The time taken for this query is **24.227 seconds.**

b. **Books with an average rating of 2**



```
hive> SELECT count(bid)
    > FROM bookAvg3
    > WHERE avg>=2.0 AND avg<3.0;
```

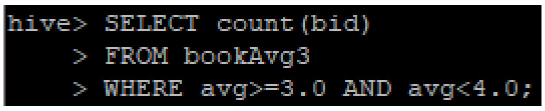
Count: 60561



```
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2021-07-04 04:33:46,125 Stage-1 map = 0%,  reduce = 0%
2021-07-04 04:33:54,304 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.41 sec
2021-07-04 04:34:00,422 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.38 sec
MapReduce Total cumulative CPU time: 6 seconds 380 msec
Ended Job = job_1625366906697_0013
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.38 sec   HDFS Read: 42913203 HDFS Write: 105 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 380 msec
OK
60561
Time taken: 22.176 seconds, Fetched: 1 row(s)
```

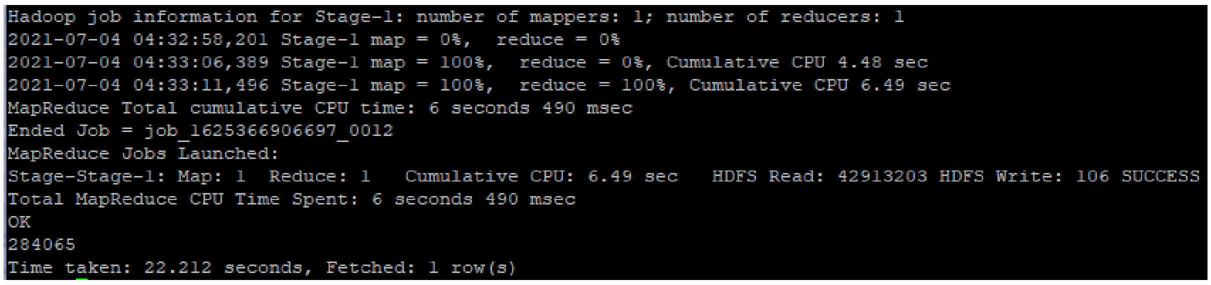
The time taken for this query is **22.176 seconds.**

c. **Books with an average rating of 3**



```
hive> SELECT count(bid)
    > FROM bookAvg3
    > WHERE avg>=3.0 AND avg<4.0;
```

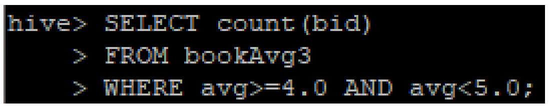
Count: 284065



```
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2021-07-04 04:32:58,201 Stage-1 map = 0%,  reduce = 0%
2021-07-04 04:33:06,389 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.48 sec
2021-07-04 04:33:11,496 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.49 sec
MapReduce Total cumulative CPU time: 6 seconds 490 msec
Ended Job = job_1625366906697_0012
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.49 sec   HDFS Read: 42913203 HDFS Write: 106 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 490 msec
OK
284065
Time taken: 22.212 seconds, Fetched: 1 row(s)
```

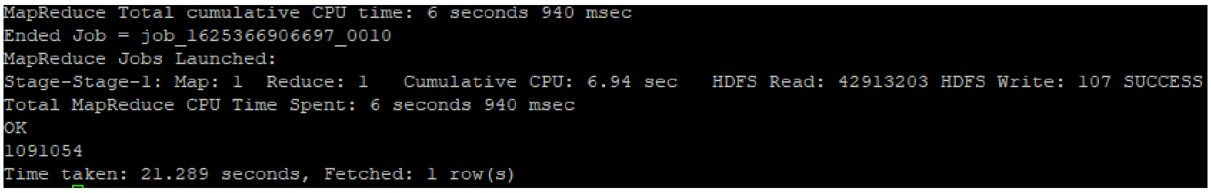
The time taken for this query is **22.212 seconds.**

d. **Books with an average rating of 4**



```
hive> SELECT count(bid)
    > FROM bookAvg3
    > WHERE avg>=4.0 AND avg<5.0;
```

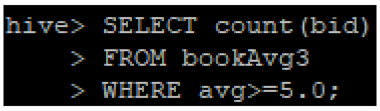
Count: 1, 091, 054



```
MapReduce Total cumulative CPU time: 6 seconds 940 msec
Ended Job = job_1625366906697_0010
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.94 sec   HDFS Read: 42913203 HDFS Write: 107 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 940 msec
OK
1091054
Time taken: 21.289 seconds, Fetched: 1 row(s)
```

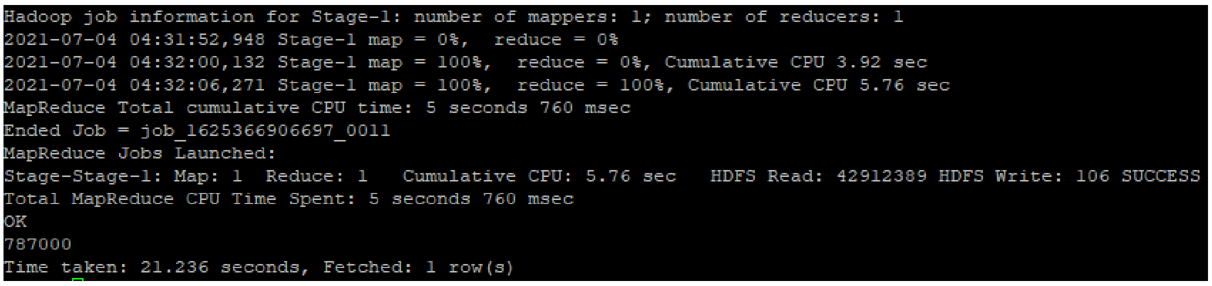
The time taken for this query is **21.289 seconds.**

e. **Books with an average rating of 5**



```
hive> SELECT count(bid)
    > FROM bookAvg3
    > WHERE avg>=5.0;
```

Count: 787000

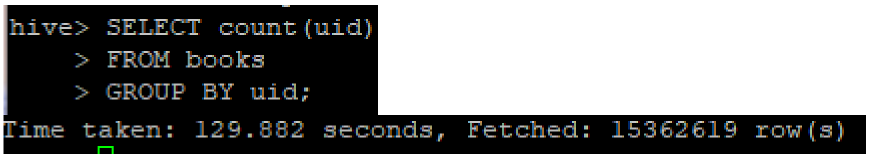


```
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2021-07-04 04:31:52,948 Stage-1 map = 0%,  reduce = 0%
2021-07-04 04:32:00,132 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.92 sec
2021-07-04 04:32:06,271 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.76 sec
MapReduce Total cumulative CPU time: 5 seconds 760 msec
Ended Job = job_1625366906697_0011
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.76 sec   HDFS Read: 42912389 HDFS Write: 106 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 760 msec
OK
787000
Time taken: 21.236 seconds, Fetched: 1 row(s)
```

The time taken for this query is **21.236 seconds.**

Research Topic 2: To identify the number of reviews done per user.

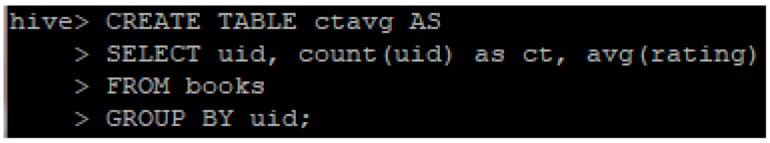
In order to satisfy Research Topic 2, we ran a count function to calculate the number of "UserID" from the organic "books" table that we had created in the beginning. We then followed it with a "GROUP BY" function with the "UserID". The "GROUP BY" function allows us to collapse a field into its distinct values. Hence, we would be able to count the number of reviews done by a distinct user.



```
hive> SELECT count(uid)
    > FROM books
    > GROUP BY uid;
Time taken: 129.882 seconds, Fetched: 15362619 row(s)
```

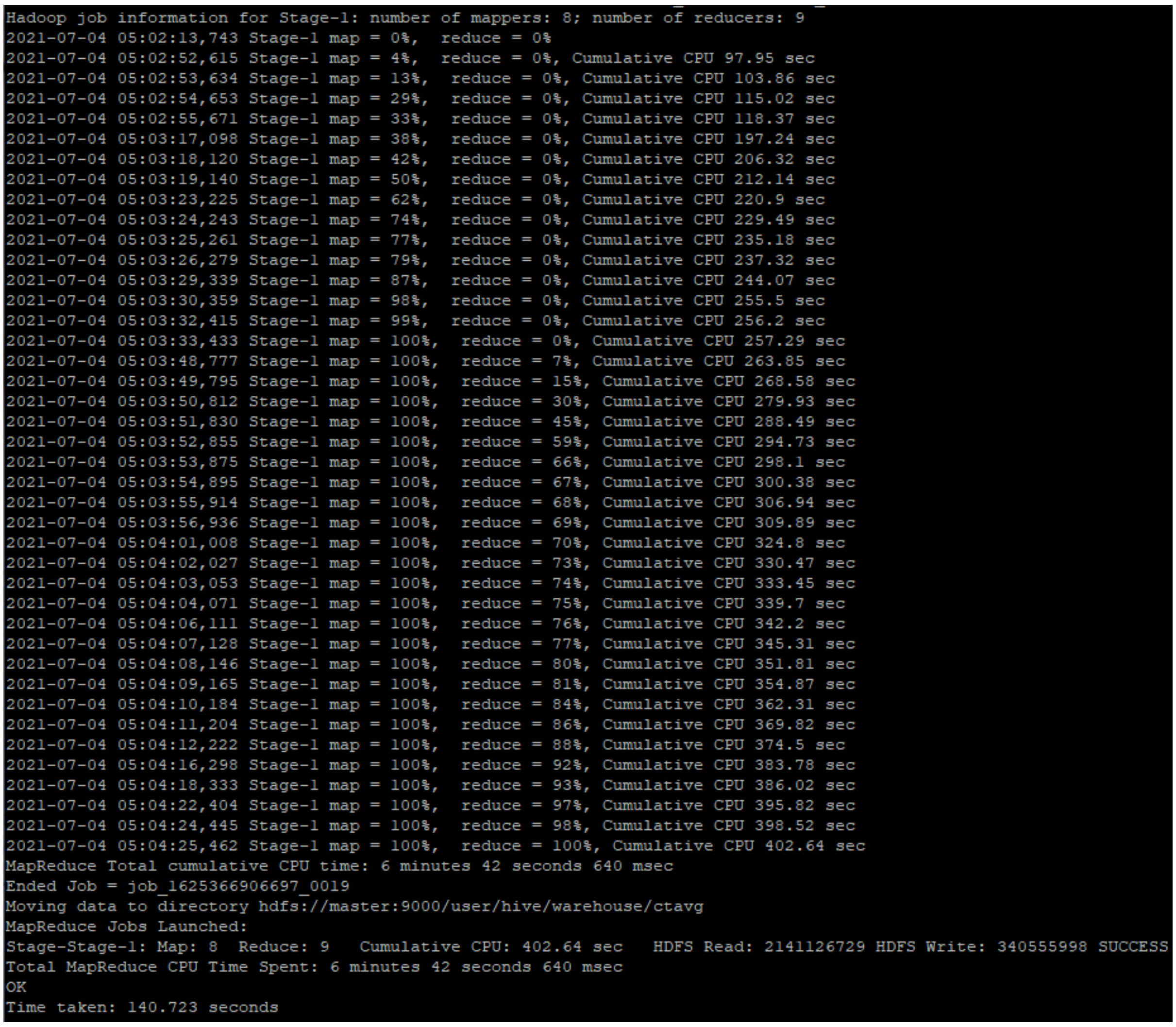
This process took **129.882 seconds.**

We also took the analysis a step further and indicated the number of reviews done per user along with the average rating.



```
hive> CREATE TABLE ctavg AS
    > SELECT uid, count(uid) as ct, avg(rating)
    > FROM books
    > GROUP BY uid;
```

The extension from the previous analysis was to add the command "Avg(rating)" which would then calculate the average rating given per user.

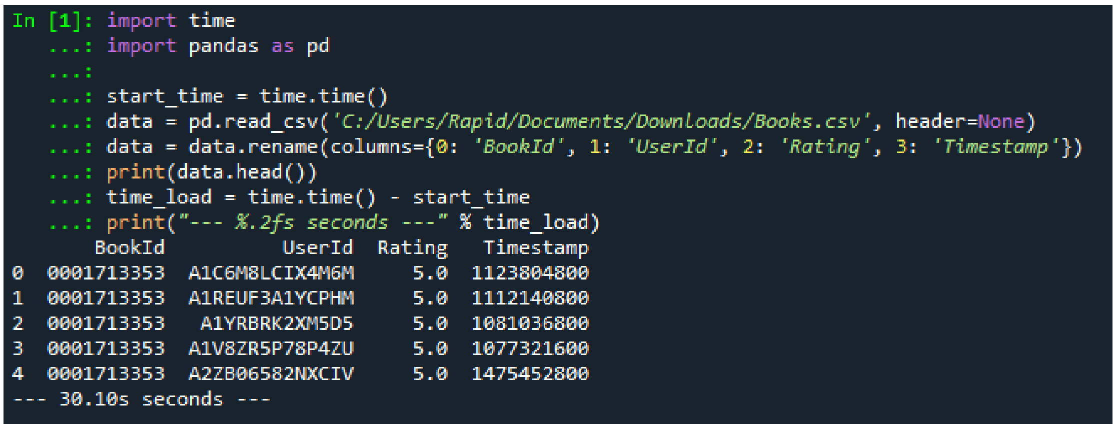


```
Hadoop job information for Stage-1: number of mappers: 8; number of reducers: 9
2021-07-04 05:02:13,743 Stage-1 map = 0%,  reduce = 0%
2021-07-04 05:02:52,615 Stage-1 map = 4%,  reduce = 0%, Cumulative CPU 97.95 sec
2021-07-04 05:02:53,634 Stage-1 map = 13%,  reduce = 0%, Cumulative CPU 103.86 sec
2021-07-04 05:02:54,653 Stage-1 map = 29%,  reduce = 0%, Cumulative CPU 115.02 sec
2021-07-04 05:02:55,671 Stage-1 map = 33%,  reduce = 0%, Cumulative CPU 118.37 sec
2021-07-04 05:03:17,098 Stage-1 map = 38%,  reduce = 0%, Cumulative CPU 197.24 sec
2021-07-04 05:03:18,120 Stage-1 map = 42%,  reduce = 0%, Cumulative CPU 206.32 sec
2021-07-04 05:03:19,140 Stage-1 map = 50%,  reduce = 0%, Cumulative CPU 212.14 sec
2021-07-04 05:03:23,225 Stage-1 map = 62%,  reduce = 0%, Cumulative CPU 220.9 sec
2021-07-04 05:03:24,243 Stage-1 map = 74%,  reduce = 0%, Cumulative CPU 229.49 sec
2021-07-04 05:03:25,261 Stage-1 map = 77%,  reduce = 0%, Cumulative CPU 235.18 sec
2021-07-04 05:03:26,279 Stage-1 map = 79%,  reduce = 0%, Cumulative CPU 237.32 sec
2021-07-04 05:03:29,339 Stage-1 map = 87%,  reduce = 0%, Cumulative CPU 244.07 sec
2021-07-04 05:03:30,359 Stage-1 map = 98%,  reduce = 0%, Cumulative CPU 255.5 sec
2021-07-04 05:03:32,415 Stage-1 map = 99%,  reduce = 0%, Cumulative CPU 256.2 sec
2021-07-04 05:03:33,433 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 257.29 sec
2021-07-04 05:03:48,777 Stage-1 map = 100%,  reduce = 7%, Cumulative CPU 263.85 sec
2021-07-04 05:03:49,795 Stage-1 map = 100%,  reduce = 15%, Cumulative CPU 268.58 sec
2021-07-04 05:03:50,812 Stage-1 map = 100%,  reduce = 30%, Cumulative CPU 279.93 sec
2021-07-04 05:03:51,830 Stage-1 map = 100%,  reduce = 45%, Cumulative CPU 288.49 sec
2021-07-04 05:03:52,855 Stage-1 map = 100%,  reduce = 59%, Cumulative CPU 294.73 sec
2021-07-04 05:03:53,875 Stage-1 map = 100%,  reduce = 66%, Cumulative CPU 298.1 sec
2021-07-04 05:03:54,895 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 300.38 sec
2021-07-04 05:03:55,914 Stage-1 map = 100%,  reduce = 68%, Cumulative CPU 306.94 sec
2021-07-04 05:03:56,936 Stage-1 map = 100%,  reduce = 69%, Cumulative CPU 309.89 sec
2021-07-04 05:04:01,008 Stage-1 map = 100%,  reduce = 70%, Cumulative CPU 324.8 sec
2021-07-04 05:04:02,027 Stage-1 map = 100%,  reduce = 73%, Cumulative CPU 330.47 sec
2021-07-04 05:04:03,053 Stage-1 map = 100%,  reduce = 74%, Cumulative CPU 333.45 sec
2021-07-04 05:04:04,071 Stage-1 map = 100%,  reduce = 75%, Cumulative CPU 339.7 sec
2021-07-04 05:04:06,111 Stage-1 map = 100%,  reduce = 76%, Cumulative CPU 342.2 sec
2021-07-04 05:04:07,128 Stage-1 map = 100%,  reduce = 77%, Cumulative CPU 345.31 sec
2021-07-04 05:04:08,146 Stage-1 map = 100%,  reduce = 80%, Cumulative CPU 351.81 sec
2021-07-04 05:04:09,165 Stage-1 map = 100%,  reduce = 81%, Cumulative CPU 354.87 sec
2021-07-04 05:04:10,184 Stage-1 map = 100%,  reduce = 84%, Cumulative CPU 362.31 sec
2021-07-04 05:04:11,204 Stage-1 map = 100%,  reduce = 86%, Cumulative CPU 369.82 sec
2021-07-04 05:04:12,222 Stage-1 map = 100%,  reduce = 88%, Cumulative CPU 374.5 sec
2021-07-04 05:04:16,298 Stage-1 map = 100%,  reduce = 92%, Cumulative CPU 383.78 sec
2021-07-04 05:04:18,333 Stage-1 map = 100%,  reduce = 93%, Cumulative CPU 386.02 sec
2021-07-04 05:04:22,404 Stage-1 map = 100%,  reduce = 97%, Cumulative CPU 395.82 sec
2021-07-04 05:04:24,445 Stage-1 map = 100%,  reduce = 98%, Cumulative CPU 398.52 sec
2021-07-04 05:04:25,462 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 402.64 sec
MapReduce Total cumulative CPU time: 6 minutes 42 seconds 640 msec
Ended Job = job_1625366906697_0019
Moving data to directory hdfs://master:9000/user/hive/warehouse/ctavg
MapReduce Jobs Launched:
Stage-Stage-1: Map: 8  Reduce: 9   Cumulative CPU: 402.64 sec   HDFS Read: 2141126729 HDFS Write: 340555998 SUCCESS
Total MapReduce CPU Time Spent: 6 minutes 42 seconds 640 msec
OK
Time taken: 140.723 seconds
```

The time taken for this process is **140.723 seconds.**

2. Analysis using the Python language for data queries *(Running on Windows 10 local personal computer)*

The first step we performed in python was to load the dataset into the python environment. Then, we renamed the columns according to their original name.

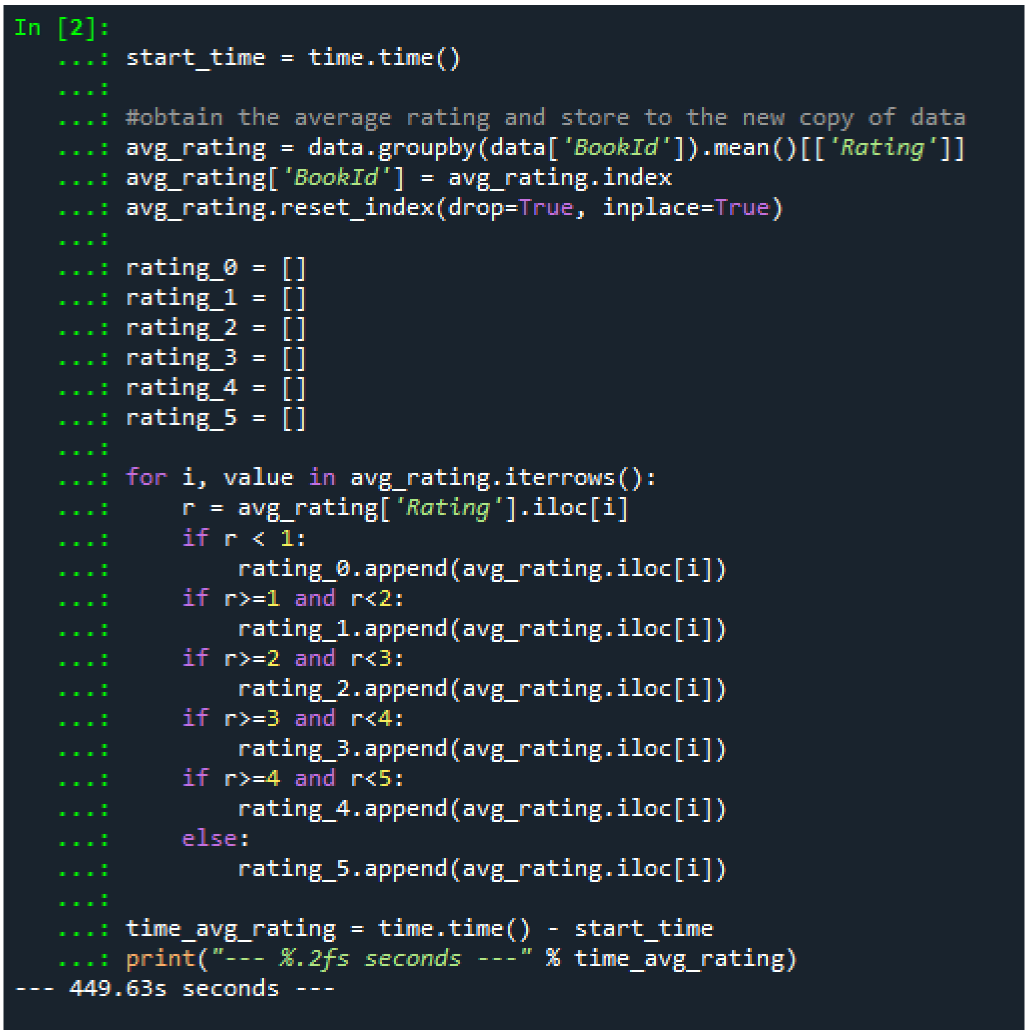


```
In [1]: import time
   ...: import pandas as pd
   ...:
   ...: start_time = time.time()
   ...: data = pd.read_csv('C:/Users/Rapid/Documents/Downloads/Books.csv', header=None)
   ...: data = data.rename(columns={0: 'BookId', 1: 'UserId', 2: 'Rating', 3: 'Timestamp'})
   ...: print(data.head())
   ...: time_load = time.time() - start_time
   ...: print("--- %.2fs seconds ---" % time_load)
       BookId          UserId  Rating   Timestamp
0  0001713353  A1C6M8LCIX4M6M     5.0  1123804800
1  0001713353  A1REUF3A1YCPHM     5.0  1112140800
2  0001713353   A1YRBRK2XM5D5     5.0  1081036800
3  0001713353  A1V8ZR5P78P4ZU     5.0  1077321600
4  0001713353  A2ZB06582NXCIV     5.0  1475452800
--- 30.10s seconds ---
```

This process took us **30.10 seconds**.

Research Topic 1: To compare the number of books in each rating group.

Then, we proceed to perform Research Topic 1. Firstly, we grouped the data by the bookID and calculated the mean of it. After each of the bookID has its mean rating, we grouped them into 6 different groups according to their mean ratings.

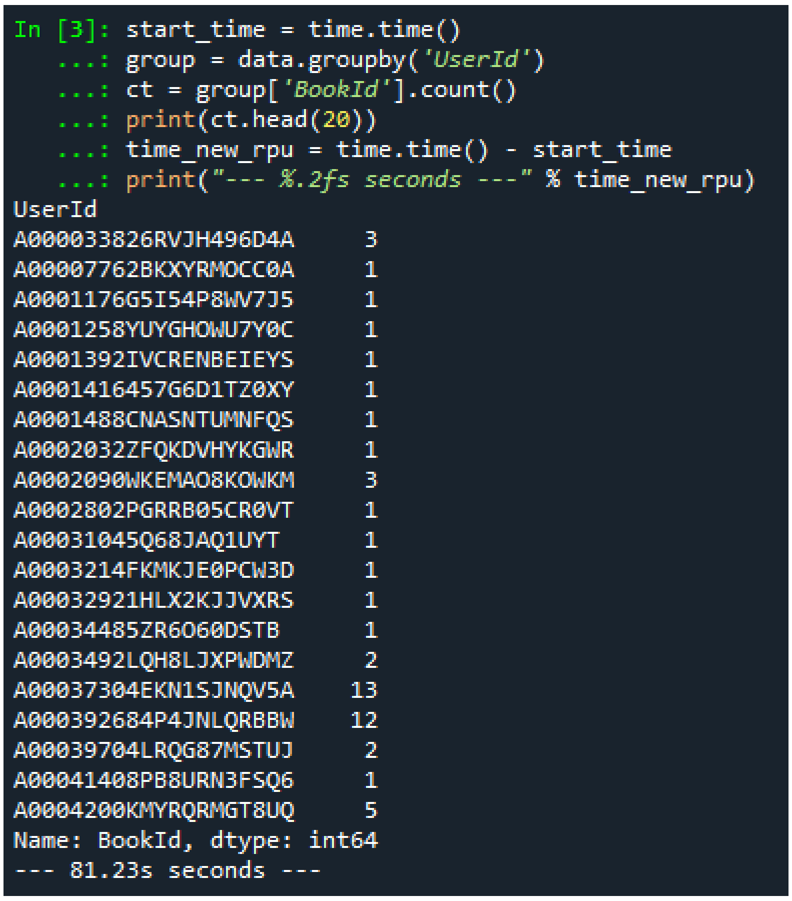


```
In [2]:
   ...: start_time = time.time()
   ...:
   ...: #obtain the average rating and store to the new copy of data
   ...: avg_rating = data.groupby(data['BookId']).mean()[['Rating']]
   ...: avg_rating['BookId'] = avg_rating.index
   ...: avg_rating.reset_index(drop=True, inplace=True)
   ...:
   ...: rating_0 = []
   ...: rating_1 = []
   ...: rating_2 = []
   ...: rating_3 = []
   ...: rating_4 = []
   ...: rating_5 = []
   ...:
   ...: for i, value in avg_rating.iterrows():
   ...:     r = avg_rating['Rating'].iloc[i]
   ...:     if r < 1:
   ...:         rating_0.append(avg_rating.iloc[i])
   ...:     if r>=1 and r<2:
   ...:         rating_1.append(avg_rating.iloc[i])
   ...:     if r>=2 and r<3:
   ...:         rating_2.append(avg_rating.iloc[i])
   ...:     if r>=3 and r<4:
   ...:         rating_3.append(avg_rating.iloc[i])
   ...:     if r>=4 and r<5:
   ...:         rating_4.append(avg_rating.iloc[i])
   ...:     else:
   ...:         rating_5.append(avg_rating.iloc[i])
   ...:
   ...: time_avg_rating = time.time() - start_time
   ...: print("--- %.2fs seconds ---" % time_avg_rating)
--- 449.63s seconds ---
```

For example, if the book has a mean rating of 1.6, it belongs to group 1. If the book has a perfect mean rating of 5, it belongs to group 5. This grouping and calculating process took 460 seconds which is also roughly 7.6 minutes to complete. In the end, this research topic took us 449.63 seconds or roughly 7.5 minutes to process

Research Topic 2: To identify the number of reviews done per user.

Moving on to our next analysis, which was done by grouping the data according to the userID, and then performing a count on them.



```
In [3]: start_time = time.time()
   ...: group = data.groupby('UserId')
   ...: ct = group['BookId'].count()
   ...: print(ct.head(20))
   ...: time_new_rpu = time.time() - start_time
   ...: print("--- %.2fs seconds ---" % time_new_rpu)
UserId
A000033826RVJH496D4A     3
A00007762BKXYRMOCC0A     1
A0001176G5I54P8WV7J5     1
A0001258YUYGHOWU7Y0C     1
A0001392IVCRENBEIEYS     1
A0001416457G6D1TZ0XY     1
A0001488CNASNTUMNFQS     1
A0002032ZFQKDVHYKGWR     1
A0002090WKEMAO8KOWKM     3
A0002802PGRRB05CR0VT     1
A00031045Q68JAQ1UYT      1
A0003214FKMKJE0PCW3D     1
A00032921HLX2KJJVXRS     1
A00034485ZR6O60DSTB      1
A0003492LQH8LJXPWDMZ     2
A00037304EKN1SJNQV5A    13
A000392684P4JNLQRBBW    12
A00039704LRQG87MSTUJ     2
A00041408PB8URN3FSQ6     1
A0004200KMYRQRMGT8UQ     5
Name: BookId, dtype: int64
--- 81.23s seconds ---
```

This time, it was taking 81 seconds or roughly 1.35 minutes to finish the task in research topic 2.

We believe that most of the time taken here was mostly for 2 parts, 1st being the part where the system needed to calculate the value of mean for each book and the 2nd one being the grouping of them according to its mean. The system needs to group them according to the same book id, then count their total ratings, and then divide by their counts. This process is a little complex that requires a lot of space on the RAM to run it faster. After having their mean ratings, the system needs to look into them one-by-one and group them accordingly based on the conditions.

The result of research topic 2 was much faster than research topic 1 definitely because of the complexity of the command. All it needs to do is group the user id accordingly and perform a count. There is no if else condition that the system needs to iterate one-by-one to check.

ANALYSIS OF THE OUTPUT

	Dataset loading	Research Topic 1	Research Topic 2
Local Machine 1	30.10 s	449.63 s	81.23 s
Local Machine 2	256.32 s	1106.13 s	174.00s

Cloud (AWS)	33.57 s	184 s	129 s

Specifications of **Local Machine 1**:
- 6 core, 12 thread CPU running at 3.6 GHz
- SSD Storage
- 16GB RAM running at 3600mhz

Specifications of **Local Machine 2**:
- 4 core, 8 thread CPU running at 2.4GHz
- SSD Storage
- 4GB RAM running at 2400mhz

Specifications of **AWS (Cloud)** – (Instance: t2.medium)
- 2 vCPU, 2 cores, 1 thread per core
- Intel AVX up to 3.3 Ghz Intel Scalable Processor
- EBS Storage
- 4GB RAM
- Low to Moderate Network Performance

Possible reasons for the difference in performance between the Local Machine and the AWS (Cloud):

1. **Storage Medium**

- The AWS EC2 uses the Amazon Elastic Block Store (EBS) as its storage medium. EBS is backed by network-connected block storage. EBS offers large storage capacity and reasonable throughput and latency.
- The local machines use the Solid State Drive (SSD) as its storage medium. SSD stores the data in an interconnected flash-memory chips that retain the data even when there's no power flowing through them.
- SSD are faster because there's no network latency, but it last for a short time and you can't detach it from an instance and attach it to another. As you can see, it is available to more powerful instances.
- EBS are more flexible, since you can attach and detach it from instances, but is a little bit slower, as more suitable for general purpose.

2. **Parallel Processing**

- The AWS EC2 infrastructure is designed for parallel processing. Parallel processing is a common computing approach to help solve large problems in a shorter period of time. By running the components of a problem concurrently, the total duration is greatly reduced.
- The local machine utilizes real time processing. Real time processing is whereby the system can input rapidly changing data and then provide output instantaneously so that the change over time can be seen very quickly.
- This would perhaps answer the question of ***"Why is the processing time of Research Question 2 longer in the AWS Cloud compared to the Local Machine?".*** This can be explained whereby the nature of the analysis and programming for Research Question 2 is a simple straight forward count function. It does not have any form or loops or recursion which would require enhanced processing that makes it behaves like a real-time process than batch in nature. Hence, the query does not benefit from the parallel processing by splitting the load to other nodes in the cluster because it does not need to do so. Hence, the processing time in the local machine is shorter than that of the AWS Cloud.
- Similarly, Research Question 1 utilizes loops within the programming. Hence, it would benefit from the parallel processing that is available within the AWS (Cloud). This explains why the processing is faster in the AWS (Cloud) compared to the local machine.

3. **CPU Capabilities**

- The AWS EC2 infrastructure in this case uses the t2.medium instance. This t2.medium instance utilizes 2 vCPU. It utilizes 2 cores and has 1 thread per core (AWS, n.d.).

- The local machine has 6 cores and 12 threads running in the CPU.
- Hence, this would explain how the processing time for Research Question 2 is much faster in the local machine compared to the AWS (Cloud).
- This is also taking into account that the parallel processing offered by the AWS (Cloud) does not have a significant impact on this analysis.

4. **RAM Capacity**
   - The AWS EC2 infrastructure utilizes 4GB RAM.
   - The local machine utilizes 16GB RAM.
   - This would also explain how the processing time for Research Topic 2 is faster in the local machine compared to the AWS (Cloud) when the parallel processing does not play a significant role when data are loaded onto RAM ready for processing.
5. **Network Connectivity**
   - The AWS EC2 infrastructure utilizing the t2.medium instance has a low to medium network connectivity.
   - The local machine meanwhile handles the processing locally on the machine itself.
   - The time to run the test across the network is making the whole test take a lot longer than running it local. This is because it involves it involves the way web requests and HTTP works, along with how TCP works, its 3 way handshake for establishing a connection, and how latency and throughput limitations affect network performance.
   - In essence, we are not testing the hardware against each other when we run the test but more of the network connectivity.
   - This would still explain why when parallel processing is not significantly required, the processing time for Research Topic 2 is faster in the local machine than in AWS (Cloud).

REFLECTION

After the application of the topics for this assignment, we deeply realize and felt the advantage of running tasks on the cloud with a powerful machine. Although we are using different languages, the task to perform was the same. Just looking at the time taken to load a dataset in our local machine alone is enough to justify why running on the cloud is a need sometimes. If the data were upscaled larger to hundreds of gigabytes or even terabytes, the time calculation will not be in seconds anymore, it could be in hours or even days. In this generation when more and more data is generated by a single individual, powerful cloud machines will definitely outperform our own local machine. Imagine sitting in front of our machine waiting endlessly to wait for the output when we need it before we can proceed with other tasks makes local machines a less convincing option when we are dealing with big data.

Based on this assignment, I have also slowly gotten to learn about the value of both running analysis on a local machine and the analysis on the cloud (AWS Cloud). Understandably, the system that runs on the cloud is much more powerful and would be able to take on a larger dataset and process it accordingly. However, this assignment has taught me that it is in fact much more than that. The one feature that makes AWS Cloud stand out is its ability to go about parallel processing. However, certain algorithms do not benefit or utilize parallel processing. Hence, in this case those algorithms would be better suited to be ran on a local machine. Ultimately, I have gotten to learn that there is no objective view to this. When choosing which machine or platform to run a given algorithm, one has to take into consideration various different factors such as specifications, type of algorithm and dataset.