Faculty of Computer Science & Information Technology

Academic Session: 2021/2022

Semester: 2

Course: WQD7007 Big Data Management

Title: Taobao User Behavior Data Analysis

Lecturer: Dr. Hoo Wai Lam

Group Project

Final Report

No	Name	ID
1	Gong Wei	17220022
2	Muhammad Amirul Daniel bin Badrul Hisham	S2115750
3	Ximin Zhang	S2118371
4	Shijie Zheng	S2108562
5	Zikun Zhao	17193835
6	Qiyi Liu	S2106632
7	Bingyan Li	S2100331
8	Sharmin Jahan	S2109943

Table of Contents

1.0 Project Background 2

1.1 Problem Statement 2

1.2 Metadata 3

1.4 Objectives 3

2.0 Methodology 3

2.1 Importing Data 4

2.2 Processing Data 5

3.0 Results & discussions 7

3.1 Identify Valuable Users 7

3.2 Analysis of the average number of users Taobao interaction 8

3.3 Bounce Rate 9

3.4 Users' Behavior Conversion Rate 9

3.5 Tools Comparison 11

4.0 Conclusion 11

4.1 Limitation and Recommendation 11

5.0 References 12

6.0 Appendices 13

1.0 Project Background

Alibaba Group Holding Co., Ltd. (abbreviation: Alibaba Group) is a company founded in 1999 in Hangzhou, Zhejiang Province by 18 founders under the leadership of Jack Ma. Alibaba Group operates a number of businesses, including: Taobao, Tmall, Juhuasuan, AliExpress, Alibaba International Marketplace, 1688, Alimama, Alibaba Cloud, Ant Group, Cainiao Network, etc. Over the past 20 years, Alibaba has completely transformed from an e-commerce company into a technology-driven platform that includes scenarios such as digital commerce, financial technology, smart logistics, cloud computing, human-land relationship, culture and entertainment, serving hundreds of millions of consumers. and tens of millions of SMEs. Alibaba is committed to developing business infrastructure in the digital economy era, helping the consumer market prosper, and promoting digitalization and intelligence in all walks of life.In the main businesses that Alibaba is responsible for, such as Taobao, based on the user's behavior on Internet products and the time and frequency of the people behind the behavior, the user's usage scenarios are deeply restored and can guide business growth. Key additions to the user model. By supplementing behavioral data, a refined and complete user portrait is constructed. The value of user behavior analysis in applications is reflected in the fact that it can help companies make more accurate judgments about the market that products enter, let market promotion personnel evaluate channel quality in a refined manner, and let product designers accurately evaluate user behavior path transformation, product revisions are excellent, and certain the impact of a new function on the product allows operators to do precise marketing and evaluate marketing results.

1.1 Problem Statement

Alibaba has grown into one of the top global e-commerce companies in the past decade. They fit well in the industry together with their top competitors, the Amazon. However, to stay relevant in this industry, Alibaba has to make sure that the people stay with their platform. But the problem comes in when, understanding the customers require more study in their shopping behavior. Many factors can influence the customers’ behavior when they are purchasing online. For example, the advertisement, the time they make the purchasing, the price and many other. Hence, studying their behavior is important as it can help in boosting the whole revenue of Alibaba. The trend on the customers’ shopping behavior could be the major strategic in helping Alibaba to make the customers stay with the platform. To do so, big data analytic comes into picture, where hidden knowledge from the data can be determined and extracted. In conjunction with that matter, to from the highlighted problem, this project will be studying the behavior of the customers by using the data from Alibaba.

1.2 Metadata

UserBehavior is a Taobao user behavior dataset provided by Alibaba for the study of implicit feedback recommendation problems. It includes five columns and about one million rows. The size of this dataset is 3.41 GB. Because the dataset is too big to execute in our computers, so we choose 4549 rows to do data analysis.

Link: https://tianchi.aliyun.com/dataset/dataDetail?dataId=649&userId=1

Columns	Type	Description
user_id	Numerical-Discrete	The id number of each user
item_id	Numerical-Discrete	The id number of each item
category_id	Numerical-Discrete	The id number of the item category
behavior_type	Categorical	User behavior type which includes pv, buy, cart and fav. ‘pv’ means product details page like the click, ‘buy’ means item purchase, ‘cart’ means add product to cart, ‘fav’ means favorite product.
time	Timestamp	The timestamp when the action occurred

1.4 Objectives

1. To process the query and visualization of the Taobao dataset by using Hadoop tools and Tableau.

2. To analyze the user's personal information attributes in order to uncover the hidden information value of users.

3. To analyze browsing, collecting, adding to shopping cart.

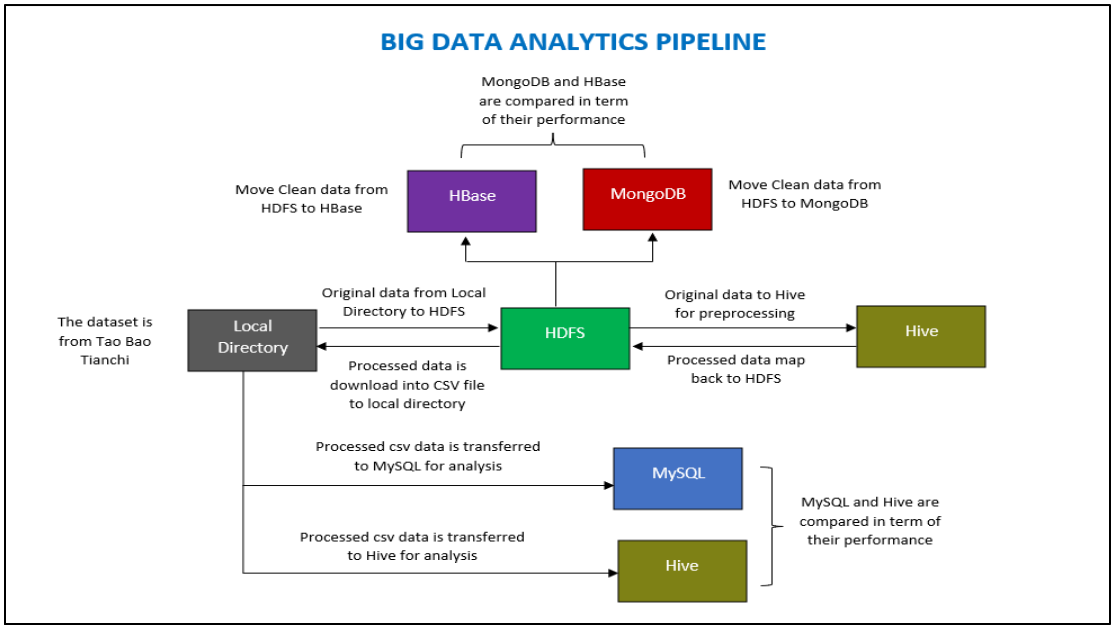
4. Identify business weaknesses such as customer churn factors during the purchase process.

2.0 Methodology

For this group assignment, our data pipeline is designed using HDFS, Hive, HBase and MySQL. Below are the explanations of the overall process of the big data pipeline.

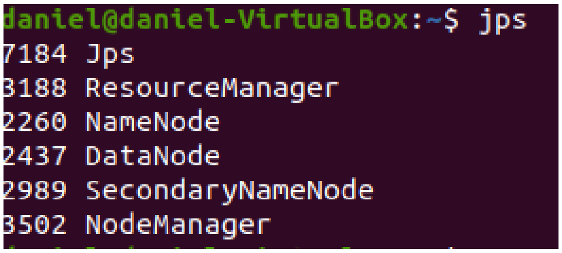
Big Data Analytics Pipeline:

The process of big data pipeline starts from the local directory, the data is uploaded to the HDFS. From HDFS, the data is preprocessed in Hive. From hive, the processed data is move to HDFS. In HDFS, the processed data is transfer to HBase and MongoDB for storing. Besides, the data is also downloaded in the local directory in the form of CSV file. From the local directory, the CSV file is uploaded to MySQL for further analysis. Plus, the analysis is also conducted on Hive for better comparison.


BIG DATA ANALYTICS PIPELINE
MongoDB and HBase are compared in term of their performance
Move Clean data from HDFS to HBase
HBase
MongoDB
Move Clean data from HDFS to MongoDB
The dataset is from Tao Bao Tianchi
Local Directory
Original data from Local Directory to HDFS
HDFS
Original data to Hive for preprocessing
Hive
Processed data is download into CSV file to local directory
Processed data map back to HDFS
Processed csv data is transferred to MySQL for analysis
MySQL
MySQL and Hive are compared in term of their performance
Processed csv data is transferred to Hive for analysis
Hive

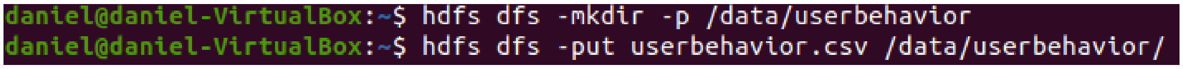
2.1 Importing Data

1. Before starting every activity, all nodes are needed to be active.

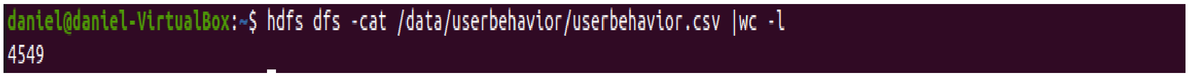


```
daniel@daniel-VirtualBox:~$ jps
7184 Jps
3188 ResourceManager
2260 NameNode
2437 DataNode
2989 SecondaryNameNode
3502 NodeManager
```

2. Create the directory /data/userbehavior in HDFS and transfer the userbehavior.csv file to that directory.

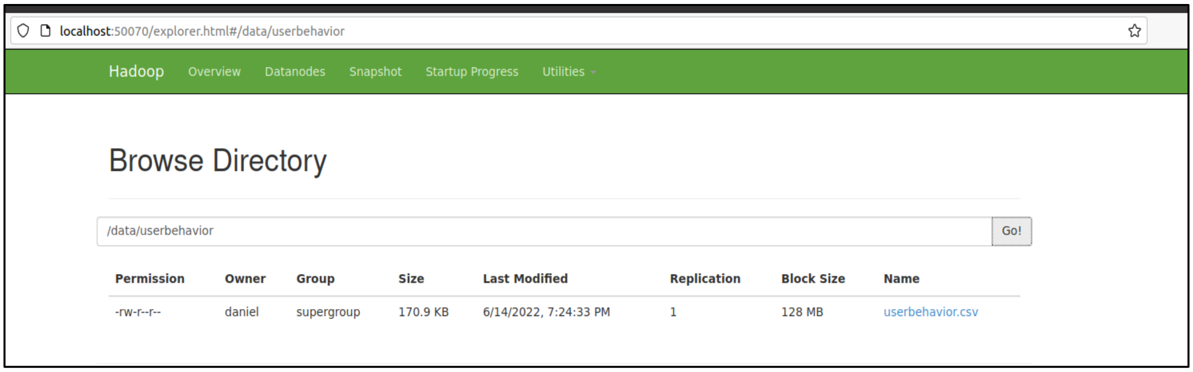


```
daniel@daniel-VirtualBox:~$ hdfs dfs -mkdir -p /data/userbehavior
daniel@daniel-VirtualBox:~$ hdfs dfs -put userbehavior.csv /data/userbehavior/
```



```
daniel@daniel-VirtualBox:~$ hdfs dfs -cat /data/userbehavior/userbehavior.csv |wc -l
4549
```

3. Enter localhost:50070 in the browser to see if the data is uploaded successfully.



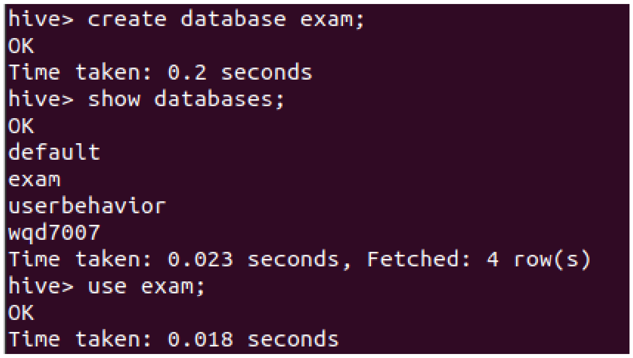
Browse Directory

/data/userbehavior

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	daniel	supergroup	170.9 KB	6/14/2022, 7:24:33 PM	1	128 MB	userbehavior.csv

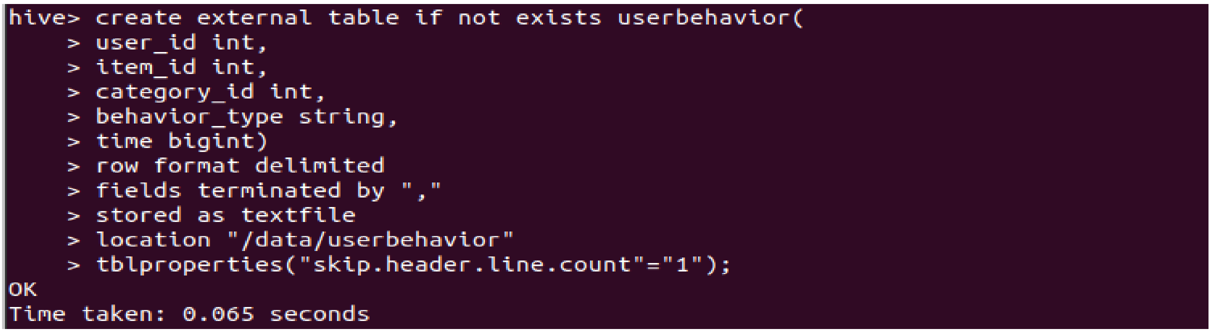
2.2 Processing Data

1. Create the database in Hive, named ‘exam’



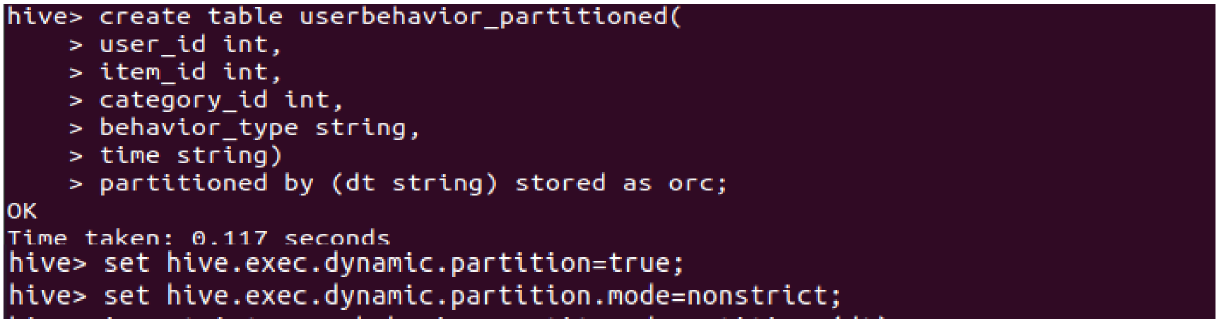
```
hive> create database exam;
OK
Time taken: 0.2 seconds
hive> show databases;
OK
default
exam
userbehavior
wqd7007
Time taken: 0.023 seconds, Fetched: 4 row(s)
hive> use exam;
OK
Time taken: 0.018 seconds
```

2. Create an external table, named ‘userbehavior’ in the ‘exam’ database and map the data from HDFS to the Hive table. Refer APPENDIX A for the results.

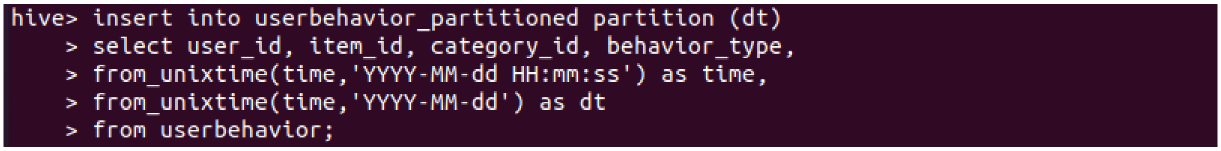


```
hive> create external table if not exists userbehavior(
    > user_id int,
    > item_id int,
    > category_id int,
    > behavior_type string,
    > time bigint)
    > row format delimited
    > fields terminated by ","
    > stored as textfile
    > location "/data/userbehavior"
    > tblproperties("skip.header.line.count"="1");
OK
Time taken: 0.065 seconds
```

3. Create an internal partitioned table userbehavior_partitioned (partitioned by dt) in the ‘exam’ database, and format the timestamp as “year-month-day hour:minute:second” by querying the userbehavior table and inserting the data into the userbehavior_partitioned table. Refer the APPENDIX B for the results.

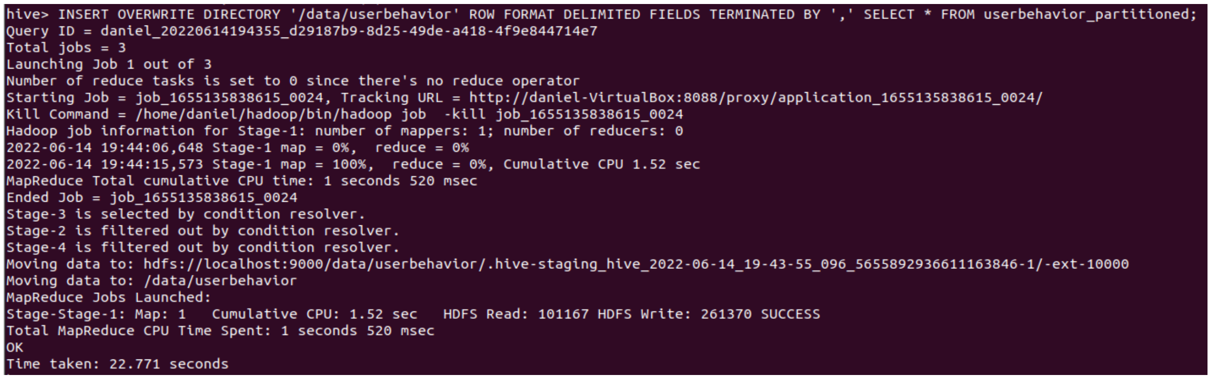


```
hive> create table userbehavior_partitioned(
    > user_id int,
    > item_id int,
    > category_id int,
    > behavior_type string,
    > time string)
    > partitioned by (dt string) stored as orc;
OK
Time taken: 0.117 seconds
hive> set hive.exec.dynamic.partition=true;
hive> set hive.exec.dynamic.partition.mode=nonstrict;
```

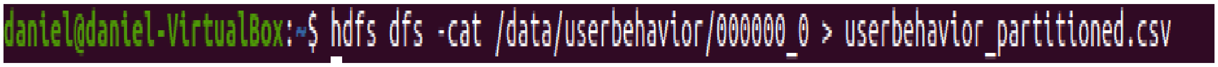


```
hive> insert into userbehavior_partitioned partition (dt)
    > select user_id, item_id, category_id, behavior_type,
    > from_unixtime(time,'YYYY-MM-dd HH:mm:ss') as time,
    > from_unixtime(time,'YYYY-MM-dd') as dt
    > from userbehavior;
```

4. Transfer the partitioned data to HDFS, and download the data into csv file. Refer APPENDIX C for the results.



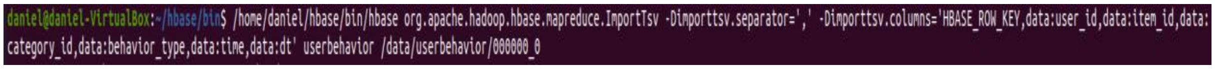
```
hive> INSERT OVERWRITE DIRECTORY '/data/userbehavior' ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' SELECT * FROM userbehavior_partitioned;
Query ID = daniel_20220614194355_d29187b9-8d25-49de-a418-4f9e844714e7
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1655135838615_0024, Tracking URL = http://daniel-VirtualBox:8088/proxy/application_1655135838615_0024/
Kill Command = /home/daniel/hadoop/bin/hadoop job  -kill job_1655135838615_0024
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2022-06-14 19:44:06,648 Stage-1 map = 0%,  reduce = 0%
2022-06-14 19:44:15,573 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.52 sec
MapReduce Total cumulative CPU time: 1 seconds 520 msec
Ended Job = job_1655135838615_0024
Stage-3 is selected by condition resolver.
Stage-2 is filtered out by condition resolver.
Stage-4 is filtered out by condition resolver.
Moving data to: hdfs://localhost:9000/data/userbehavior/.hive-staging_hive_2022-06-14_19-43-55_096_5655892936611163846-1/-ext-10000
Moving data to: /data/userbehavior
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 1.52 sec   HDFS Read: 101167 HDFS Write: 261370 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 520 msec
OK
Time taken: 22.771 seconds
```



```
daniel@daniel-VirtualBox:~$ hdfs dfs -cat /data/userbehavior/000000_0 > userbehavior_partitioned.csv
```

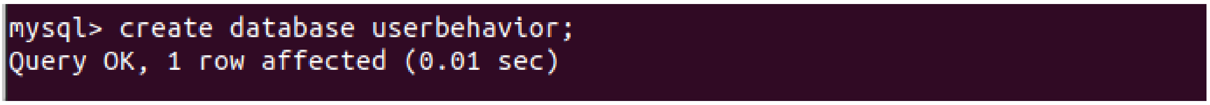
5. The data is transferred from the HDFS to HBase for storing. The table is created in HBase. Kindly refer to APPENDIX D to view the steps.

6. The data is imported to HBase. Refer APPENDIX E for the result. Apart from HBase, the data is also uploaded to MongoDB for better comparison in their performance. Kindly refer to APPENDIX F for the steps to import the dataset to MongoDB by using mongoimport.

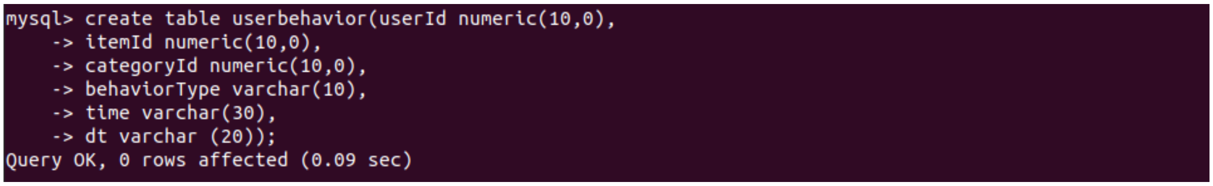


```
daniel@daniel-VirtualBox:~/hbase/bin$ /home/daniel/hbase/bin/hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator=',' -Dimporttsv.columns='HBASE_ROW_KEY,data:user_id,data:item_id,data:
category_id,data:behavior_type,data:time,data:dt' userbehavior /data/userbehavior/000000_0
```

7. The analysis of the data is conducted on MySQL. Create a database named ‘userbehavior’ and also a table named ‘userbehavior’. Refer APPENDIX G for the results of creating the database successfully.

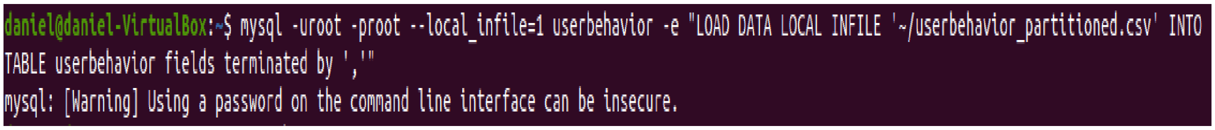


```
mysql> create database userbehavior;
Query OK, 1 row affected (0.01 sec)
```



```
mysql> create table userbehavior(userId numeric(10,0),
    -> itemId numeric(10,0),
    -> categoryId numeric(10,0),
    -> behaviorType varchar(10),
    -> time varchar(30),
    -> dt varchar (20));
Query OK, 0 rows affected (0.09 sec)
```

8. Transfer the downloaded userbehavior_partitioned.csv file into MySQL table for further analysis. Refer APPENDIX H for the results.



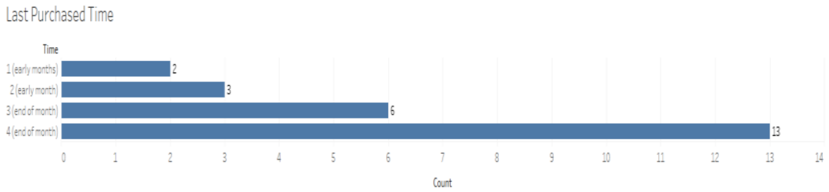
```
daniel@daniel-VirtualBox:~$ mysql -uroot -proot --local_infile=1 userbehavior -e "LOAD DATA LOCAL INFILE '~/userbehavior_partitioned.csv' INTO TABLE userbehavior fields terminated by ','"
mysql: [Warning] Using a password on the command line interface can be insecure.
```

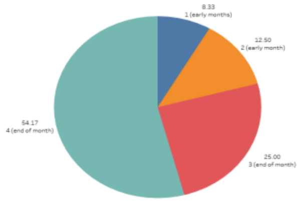
3.0 Results & discussions

In results and discussion, analysis of the dataset will be performed. The analysis is mainly conducted by using MySQL. In addition, to have a better comparison between the performance of the tools, Hive is also used for the analysis. Please refer to the APPENDIX N for the query to do the Hive analysis (Few analyses are conducted so that it can be compared with MySQL). The results and comparison also will be mentioned there in the APPENDIX N. In addition, the additional tools like Tableau are also used to enhance visualization in order to have a better understanding and clear view of the dataset.

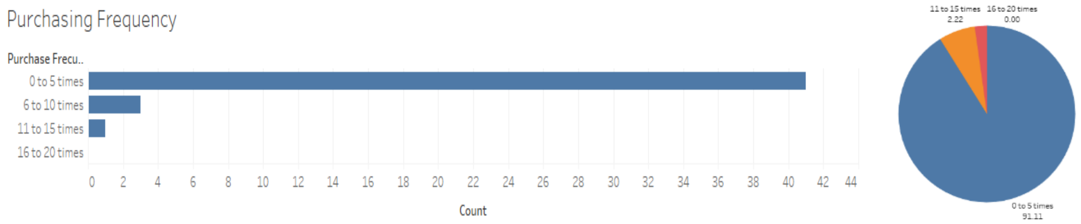
3.1 Identify Valuable Users

To calculate user value, the last purchased time (R) and consumption frequency (F) of the users are determined. The dataset is collected within 9 days of operation, which are starting from 25th November 2017 until 3rd December 2017. In total, there will be 9 days of studied. The purchased time of the users are classified into few classes. They are 0-1, 2-3, 4-5 and 6-8 days. The classes for these days are classified into 4, 3, 2 and 1 respectively. The queries and steps for performing this activity are shown in APPENDIX I. From this analysis, the pattern of purchasing time among the Taobao's users can be understood. From the result, many purchased can be seen are done at the end of the month compare to the early month. About 54.17 percent and 25 percent of the last purchasing among the users are within the last months of November compare to the early month of December which is all only below than 15 percent.

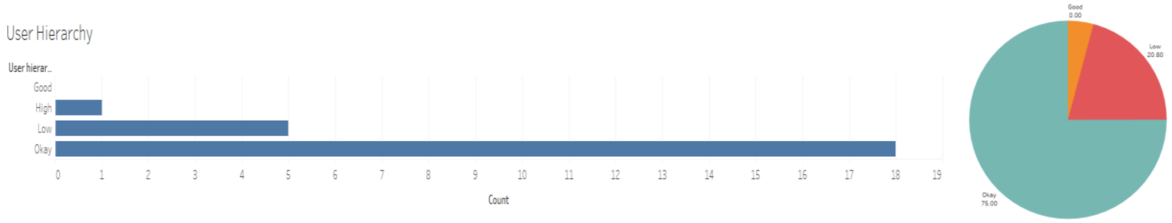




Meanwhile, for the consumption frequency (F) of the users, first, the customers who made the most purchasing are determined. Referring to APPENDIX J, it illustrates that user with the ID of 1000085 made the most purchasing within this period of study. About 12 purchasing were done in 9 days. After knowing the maximum number of purchasing, it is classified into classes, where 0 to 5 is 1, 6 to 10 is 2, 11 to 15 is 3 and finally, 16 to 20 is 4. Looking on the results in APPENDIX K, there are lots of users who are categories in the first category, which indicates that their purchasing is between 0 and 5. To ease the visualisation, bar graph and pie chart is plotted. From the graph, about 41 users make their purchasing within the period of study, that are only between 0 to 5 times. This number has made up to 91.11% of the whole users in this study. However, from this pattern, it can be concluded that, it is normal for most of the individual to make their purchasing within 0 to 5 times in 9 days. Maybe making lots of promotion on Taobao can enhance the purchasing frequency among the users.



Finally, the users are classified into their hierarchy. By integrating these 2 elements, R and F, the value of the users is determined by grouping them into categories starting from ‘low’, ‘okay’, ‘good’ and ‘high’, in the user value hierarchy. According to APPENDIX L, there are lot of users are within the ‘okay’ range of user value. To ease the visualisation, bar graph and pie chart is plotted. From the value counted, only 1 user is considered as the ‘high’ value. About 18 of the other users are ‘okay’. This has made up to 75 percent of the users that is ‘okay’ level in the user hierarchy. The dataset contains less ‘high’ user value customers but with high ‘okay’ value customers. Hence, looking into that matter many actions can be taken to improve the value of the users on Taobao platform. For example, making a month-end-sales on Taobao is suggested since more people make their purchased at the end of the month.



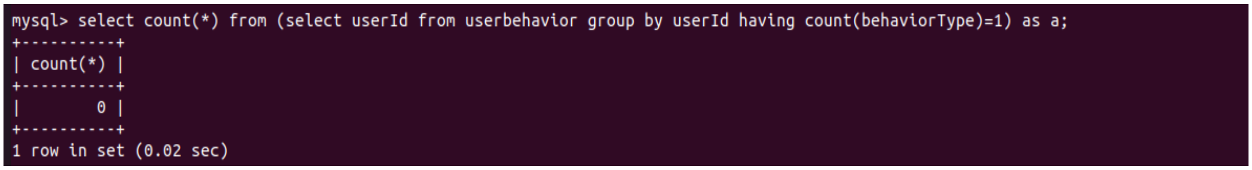
3.2 Analysis of the average number of users Taobao interaction

The average of users browses and interact with Taobao page are also determined. Referring to APPENDIX M, there are in total of 45 individual users (UV) browse into Taobao. However, number of PV (page interaction) is 4145. Average of users' interaction are equal to PV over UV. Calculating the average interaction for each user is 92.11. Looking on that interaction, 92.11 for each user is a big number. Hence, we can conclude that Taobao had made a good effort in maintaining a good average interaction.

Average Number of Users Taobao Interaction
PV/UV
92.11

3.3 Bounce Rate

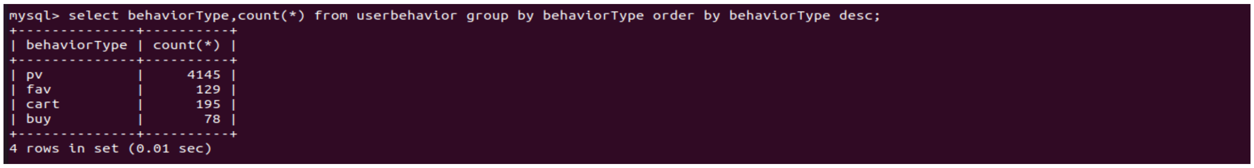
The bounce rate refers to the number of users who click only once to view the page / total user visits (PV). Besides that, bounce rate can also refer to the number of PVs that close after visiting the page as a percentage of the total PVs. However, according to results below, there is no users who click only once to view the page. The bounce rate is 0. This indicates that no one left Taobao after only browsing one page. This indicates that Taobao is attractive enough to users.



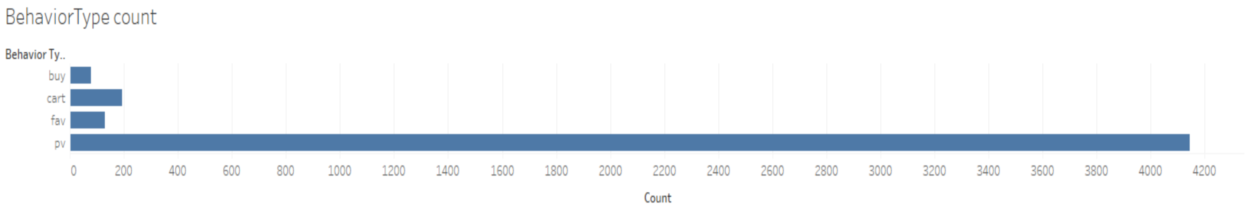
```
mysql> select count(*) from (select userId from userbehavior group by userId having count(behaviorType)=1) as a;
+----------+
| count(*) |
+----------+
|        0 |
+----------+
1 row in set (0.02 sec)
```

3.4 Users' Behavior Conversion Rate

The number of behaviorType is counted. Total number of each behavior is sorted. From the dataset, we can see that about 4145 users clicked 'pv' on the website, 195 users 'cart' the item on the website, 129 'fav', and finally, only 78 users actually purchased the item listed on the website. The picture below shows the number of users' behavior counts in the dataset. To ease the visualisation, graph is plotted and is shown in the figure below.



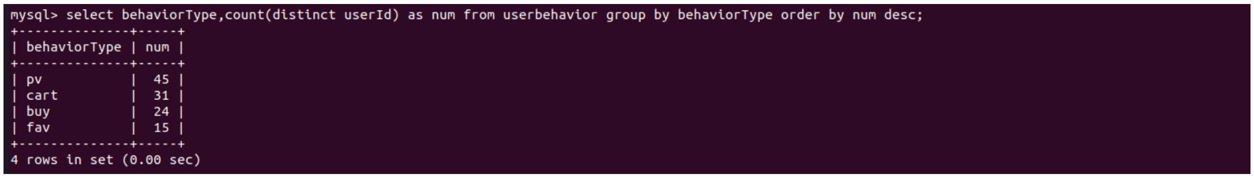
```
mysql> select behaviorType,count(*) from userbehavior group by behaviorType order by behaviorType desc;
+--------------+----------+
| behaviorType | count(*) |
+--------------+----------+
| pv           |     4145 |
| fav          |      129 |
| cart         |      195 |
| buy          |       78 |
+--------------+----------+
4 rows in set (0.01 sec)
```

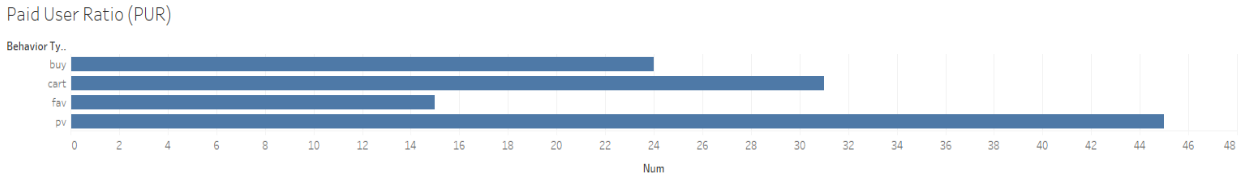

BehaviorType count

Conversion rate from browsing to purchase intention	Purchase rate using shopping cart and collection function
(fav + cart) / pv	buy / (fav + cart)
7.82%	24.07%

Collection (fav) and add to cart are referring to the intention between browsing and purchasing stages. Hence, they are counted as the same stage. We can see that the conversion rate from browsing to purchase intention is only 7.82%. Undeniably, some of the users are directly purchasing without collecting and adding to cart, but it also shows that most users browse the page more times, and use the shopping cart and collection function at minimum. However, the number of purchases rate using cart and collection function is 24.07%. This indicates that the stage from browsing to making collection and adding to cart is the key link to improve the index.

Meanwhile to understand the conversion rate for each of the individual users, the paid user ratio (PUR) is calculated. PUR is referring to the number of completed purchase over the number of individual users. Hence, in conjunction with that matter, the number of individual users for each behaviorType is calculated. Graph is plotted to ease the visualisation. According to the results below, there are 45 user 'pv', 31 users 'cart', 24 users 'buy' and 15 users 'fav'. In addition, the PUR of users who use the Taobao is 53.33%. The conversion rate of paid users is quite high. This shows that Taobao have done a great job in ensuring a high conversion rate among its customers.


```
mysql> select behaviorType,count(distinct userId) as num from userbehavior group by behaviorType order by num desc;
+--------------+-----+
| behaviorType | num |
+--------------+-----+
| pv           |  45 |
| cart         |  31 |
| buy          |  24 |
| fav          |  15 |
+--------------+-----+
4 rows in set (0.00 sec)
```


Paid User Ratio (PUR)

Percentage of PUR paying users
buy/uv
53.33%

3.5 Tools Comparison

Criteria	Data Storing		Data Access (Analysis)	
	HBase	MongoDB	MySQL	Hive (APPENDIX N)
Execution Time	It has a longer time to import to HBase	It has a better and faster execution time to import the dataset to MongoDB	It runs quicker than Hive based on our dataset. It performs much better with smaller datasets.	It runs more time than MySQL based on our dataset. It works very well in large datasets.
Structure	It prints one by one, but it is not as good as the JSON view.	It has a better look and structure. JSON is tidier to view.	It can output a table-like result, easy to read.	It can output a non-frame table-like result, easy to read.
Query	The query is simpler compare to MongoDB.	A little bit longer in term of query. But it is understandable if it is learned.	We are very familiar with its SQL query language.	HiveQL is similar to MySQL. But HiveQL does not support 'Update'.
Difficulty	Need to create a table with specific column family.	It is easier to create the database. Once import the database is created automatically.	It is easier to create database, table and do CRUD query.	It is easy to create database, table and do query language.

4.0 Conclusion

In summary, all project objectives are achieved. Firstly, query processing and visualization of the Taobao dataset is achieved through Hadoop tools and Tableau. Next, the user's personal information attributes are analyzed successfully to uncover the hidden information value of the users. And finally, browsing, collecting, adding to shopping cart and purchasing behavior are also been analyzed to determine the loss of users in the purchasing process.

4.1 Limitation and Recommendation

We successfully transferred the data from HDFS to HBase. However, only user value (distinct user) is present in the HBase. Due to time and experience limitation, we could not rectify the issue behind this problem. Detail study is suggested to be done in the future.

5.0 References

Sun, Y. D. (2021). A User Behavior Analysis Method Based on Big Data. *2021 5th Annual International Conference on Data Science and Business Analytics (ICDSBA)*. https://doi.org/10.1109/icdsba53075.2021.00101

Liu, T. (2021). Research on the Design of E-commerce Data Analysis Platform. *2021 2nd International Conference on E-Commerce and Internet Technology (ECIT)*. https://doi.org/10.1109/ecit52743.2021.00034

Guan, J., Yao, S., Xu, C., & Zhang, H. (2014). Design and Implementation of Network User Behaviors Analysis Based on Hadoop for Big Data. *Applications and Techniques in Information Security*, 44–55. https://doi.org/10.1007/978-3-662-45670-5_5

Shuai, W., Na, Z., & Sun-hao, Z. (2019). A Visual and Statistical Analysis of Taobao Double Eleven Open Data Set. *Proceedings of the 2019 3rd High Performance Computing and Cluster Technologies Conference*. https://doi.org/10.1145/3341069.3341083

6.0 Appendices

GROUPMATE TASK DISTRIBUTION

This project is a process-based. Hence, most of us do it separately. We tried in our respective machine/computer to achieved the objectives.

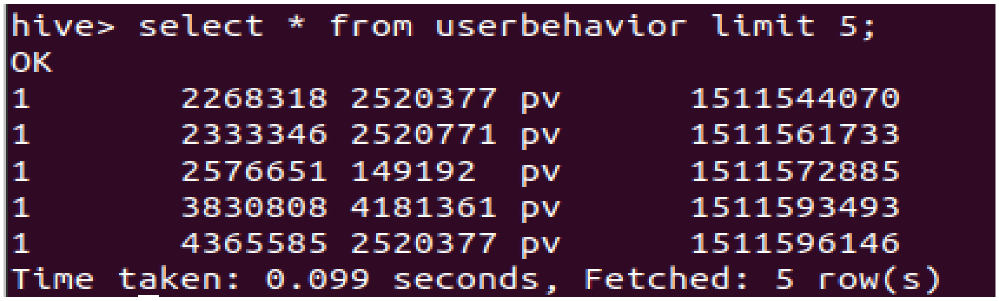
Daniel & Gong Wei - The task delegation is done according to the objective. We set up the environment for this project and do some plotting on Tableau for better visualization. For the project setup, we did some works on the data transfer and table setup. The data is transferred to Hive for pre-processing. Then it is transferred to MongoDB and HBase for storing, and finally to MySQL for analysis. We also compare the MongoDB and HBase and found MongoDB has a better and faster execution time. The analysis part will be fulfilled in other objectives.

Ximin Zhang & Zikun Zhao – Our task is about objective 2, so we compared Hive and MySQL in this objective to execute the select query, and we found hive spent more time when execute the select query, then we select MySQL to create table for further analysis. We used RFM model and classified some attributes such as purchase days, customer, shopping amounts into different categorizes that we can analyze them clearly. Also, based on the visualization, we came to conclusions and gave suggestions for online shop owners.

Shijie Zheng & Qiyi Liu – We achieved the third goal of our group project by using MySQL and gave important advice on Hadoop operations. After query and visualization, we analyzed the results to determine the user's entire process from browsing to final purchase, including browsing, favorites, adding to shopping carts, and churn at the purchase stage, which will help Alibaba locate problems and improve services. Our work is critical to the completion of group projects.

Sharmin Jahan & Bingyan Li – We have achieved the fourth objective of this project through analyzing the average number of interactions Taobao is having in their platform in terms of different set of user behavior, which indicates the strength of Taobao in maintaining loyal customers in their platform. Also, the bounce rate has been calculated in order to find the customer churn ratio in Taobao platform to identify the weakness of this platform. The analysis for obtaining this objective has been done using MySQL.

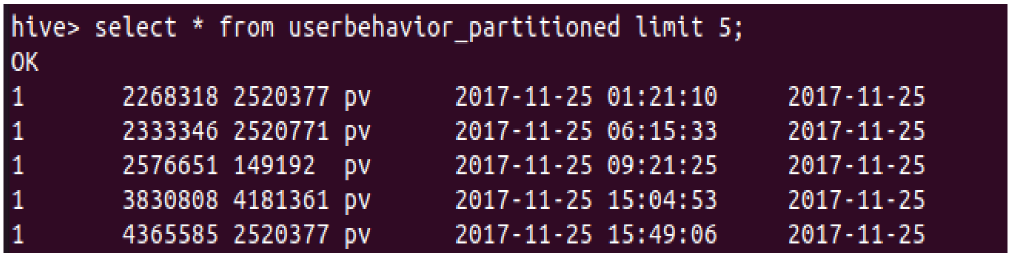
APPENDIX A



```
hive> select * from userbehavior limit 5;
OK
1       2268318 2520377 pv      1511544070
1       2333346 2520771 pv      1511561733
1       2576651 149192  pv      1511572885
1       3830808 4181361 pv      1511593493
1       4365585 2520377 pv      1511596146
Time taken: 0.099 seconds, Fetched: 5 row(s)
```

Figure 1: Hive Results after the data is uploaded successfully

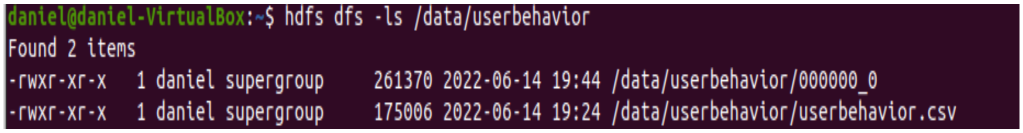
APPENDIX B



```
hive> select * from userbehavior_partitioned limit 5;
OK
1       2268318 2520377 pv      2017-11-25 01:21:10     2017-11-25
1       2333346 2520771 pv      2017-11-25 06:15:33     2017-11-25
1       2576651 149192  pv      2017-11-25 09:21:25     2017-11-25
1       3830808 4181361 pv      2017-11-25 15:04:53     2017-11-25
1       4365585 2520377 pv      2017-11-25 15:49:06     2017-11-25
```

Figure 2: Hive Result After Preprocessed (With new time column)

APPENDIX C



```
daniel@daniel-VirtualBox:~$ hdfs dfs -ls /data/userbehavior
Found 2 items
-rwxr-xr-x   1 daniel supergroup     261370 2022-06-14 19:44 /data/userbehavior/000000_0
-rwxr-xr-x   1 daniel supergroup     175006 2022-06-14 19:24 /data/userbehavior/userbehavior.csv
```

Figure 3: Preprocessed data is transferred to HDFS (000000_0)

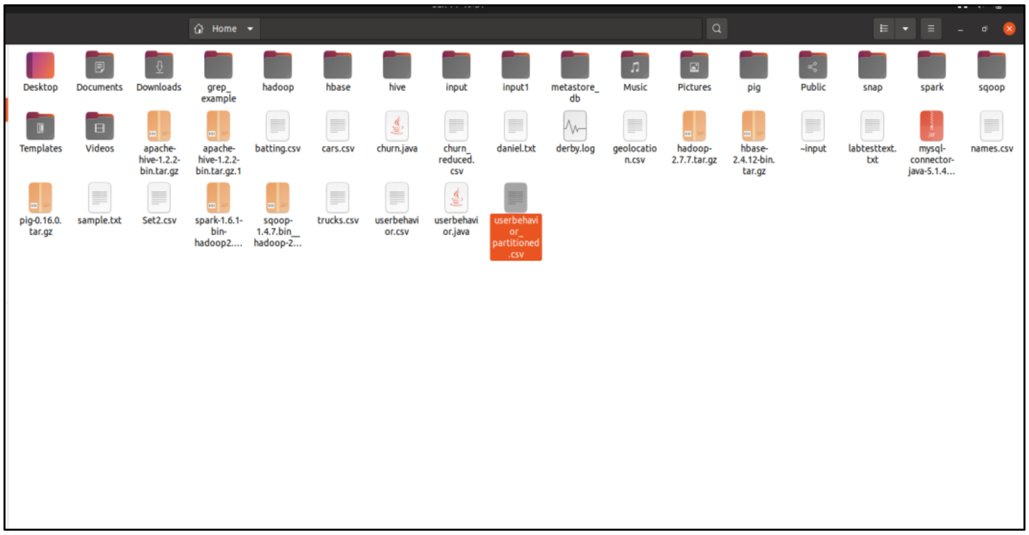
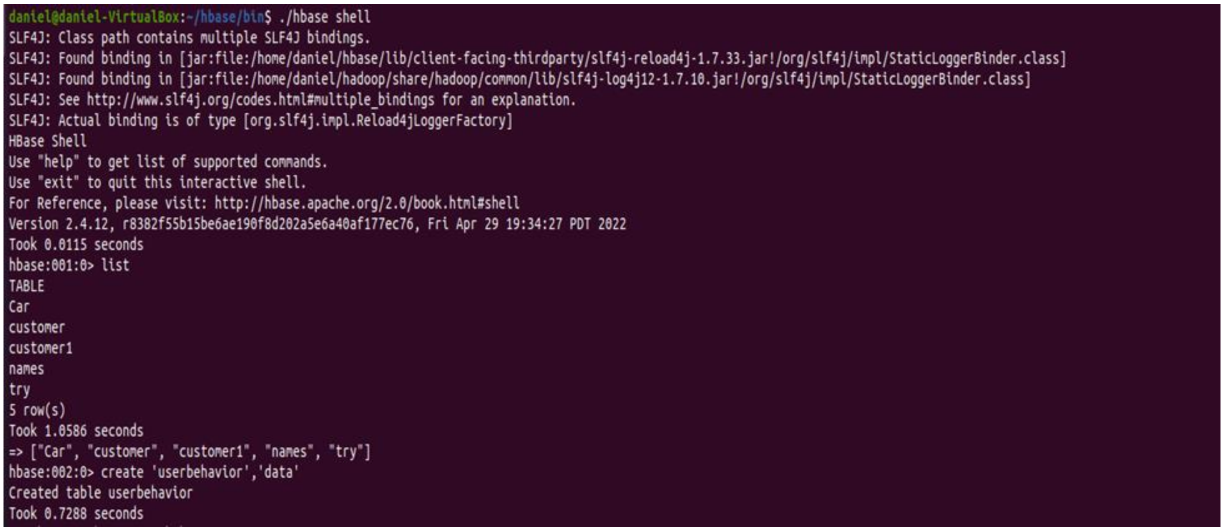


Figure 4: The new CSV preprocessed data is downloaded from HDFS to the local directory

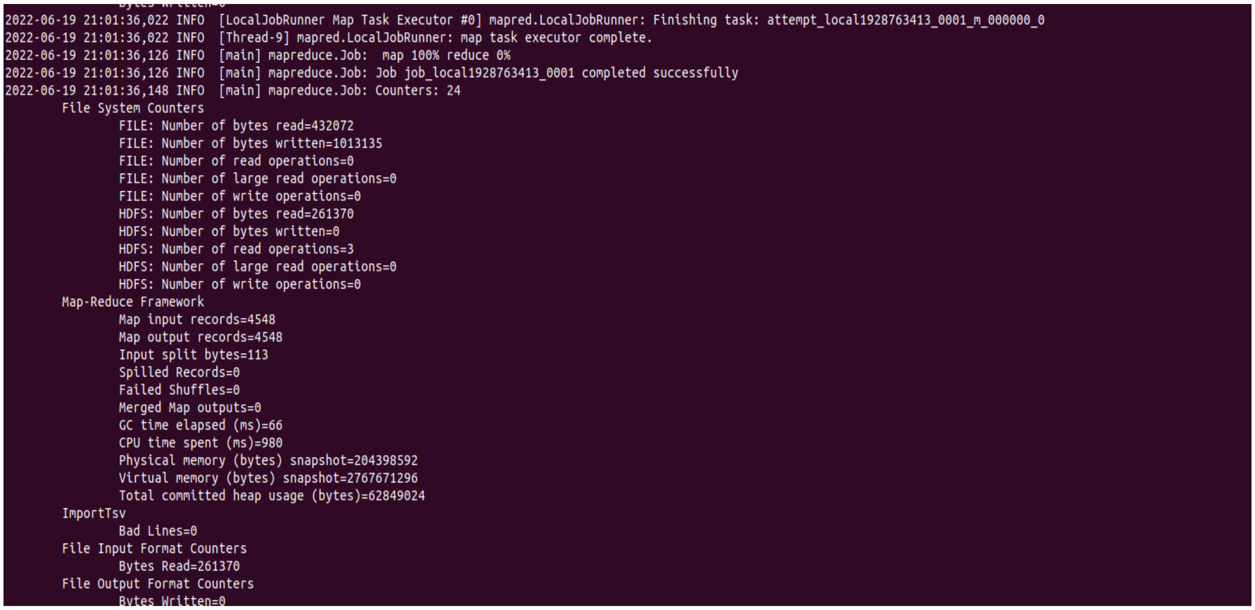
APPENDIX D



```
daniel@daniel-VirtualBox:~/hbase/bin$ ./hbase shell
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/daniel/hbase/lib/client-facing-thirdparty/slf4j-reload4j-1.7.33.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/daniel/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Reload4jLoggerFactory]
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell
Version 2.4.12, r8382f55b15be6ae190f8d202a5e6a40af177ec76, Fri Apr 29 19:34:27 PDT 2022
Took 0.0115 seconds
hbase:001:0> list
TABLE
Car
customer
customer1
names
try
5 row(s)
Took 1.0586 seconds
=> ["Car", "customer", "customer1", "names", "try"]
hbase:002:0> create 'userbehavior','data'
Created table userbehavior
Took 0.7288 seconds
```

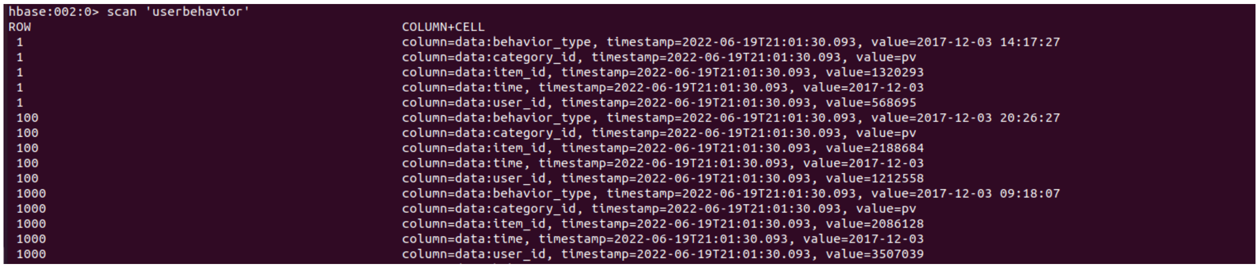
Figure 5: Create Table in HBase

APPENDIX E



```
2022-06-19 21:01:36,022 INFO  [LocalJobRunner Map Task Executor #0] mapred.LocalJobRunner: Finishing task: attempt_local1928763413_0001_m_000000_0
2022-06-19 21:01:36,022 INFO  [Thread-9] mapred.LocalJobRunner: map task executor complete.
2022-06-19 21:01:36,126 INFO  [main] mapreduce.Job:  map 100% reduce 0%
2022-06-19 21:01:36,126 INFO  [main] mapreduce.Job: Job job_local1928763413_0001 completed successfully
2022-06-19 21:01:36,148 INFO  [main] mapreduce.Job: Counters: 24
        File System Counters
                FILE: Number of bytes read=432072
                FILE: Number of bytes written=1013135
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=261370
                HDFS: Number of bytes written=0
                HDFS: Number of read operations=3
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=0
        Map-Reduce Framework
                Map input records=4548
                Map output records=4548
                Input split bytes=113
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=66
                CPU time spent (ms)=980
                Physical memory (bytes) snapshot=204398592
                Virtual memory (bytes) snapshot=2767671296
                Total committed heap usage (bytes)=62849024
        ImportTsv
                Bad Lines=0
        File Input Format Counters
                Bytes Read=261370
        File Output Format Counters
                Bytes Written=0
```

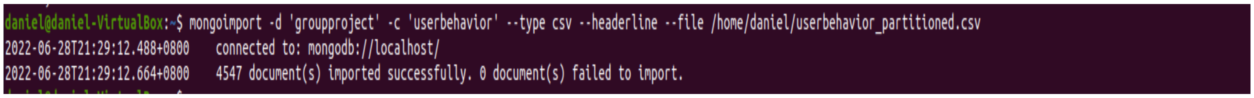
Figure 6: Result of importing to HBase



```
hbase:002:0> scan 'userbehavior'
ROW                                                  COLUMN+CELL
 1                                                   column=data:behavior_type, timestamp=2022-06-19T21:01:30.093, value=2017-12-03 14:17:27
 1                                                   column=data:category_id, timestamp=2022-06-19T21:01:30.093, value=pv
 1                                                   column=data:item_id, timestamp=2022-06-19T21:01:30.093, value=1320293
 1                                                   column=data:time, timestamp=2022-06-19T21:01:30.093, value=2017-12-03
 1                                                   column=data:user_id, timestamp=2022-06-19T21:01:30.093, value=568695
 100                                                 column=data:behavior_type, timestamp=2022-06-19T21:01:30.093, value=2017-12-03 20:26:27
 100                                                 column=data:category_id, timestamp=2022-06-19T21:01:30.093, value=pv
 100                                                 column=data:item_id, timestamp=2022-06-19T21:01:30.093, value=2188684
 100                                                 column=data:time, timestamp=2022-06-19T21:01:30.093, value=2017-12-03
 100                                                 column=data:user_id, timestamp=2022-06-19T21:01:30.093, value=1212558
 1000                                                column=data:behavior_type, timestamp=2022-06-19T21:01:30.093, value=2017-12-03 09:18:07
 1000                                                column=data:category_id, timestamp=2022-06-19T21:01:30.093, value=pv
 1000                                                column=data:item_id, timestamp=2022-06-19T21:01:30.093, value=2086128
 1000                                                column=data:time, timestamp=2022-06-19T21:01:30.093, value=2017-12-03
 1000                                                column=data:user_id, timestamp=2022-06-19T21:01:30.093, value=3507039
```

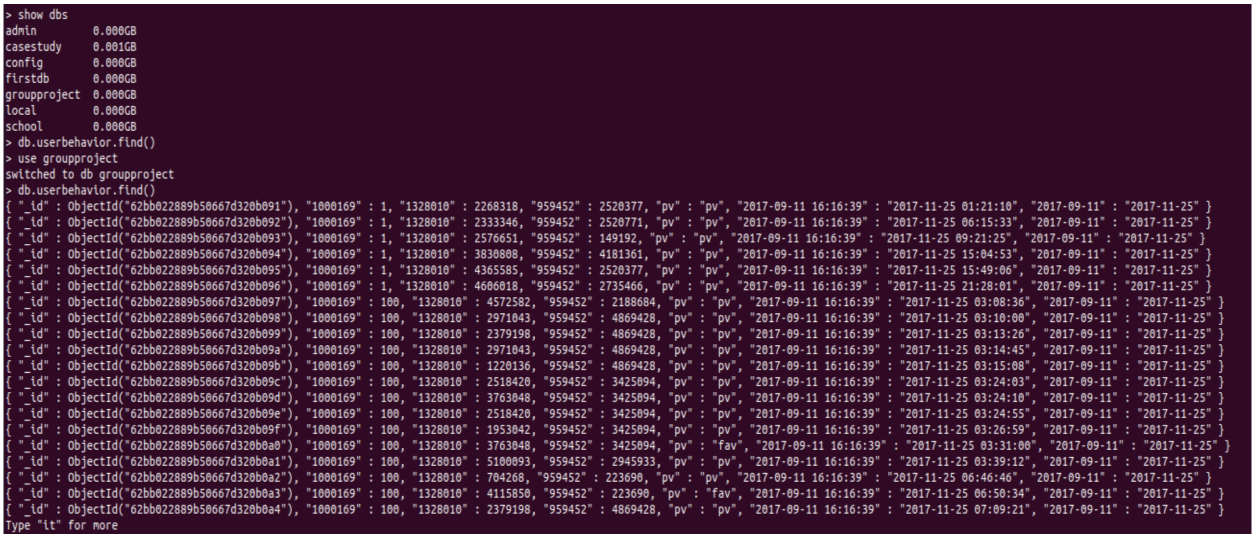
Figure 7: Sample data that is successfully imported

APPENDIX F



```
daniel@daniel-VirtualBox:~$ mongoimport -d 'groupproject' -c 'userbehavior' --type csv --headerline --file /home/daniel/userbehavior_partitioned.csv
2022-06-28T21:29:12.488+0800    connected to: mongodb://localhost/
2022-06-28T21:29:12.664+0800    4547 document(s) imported successfully. 0 document(s) failed to import.
```

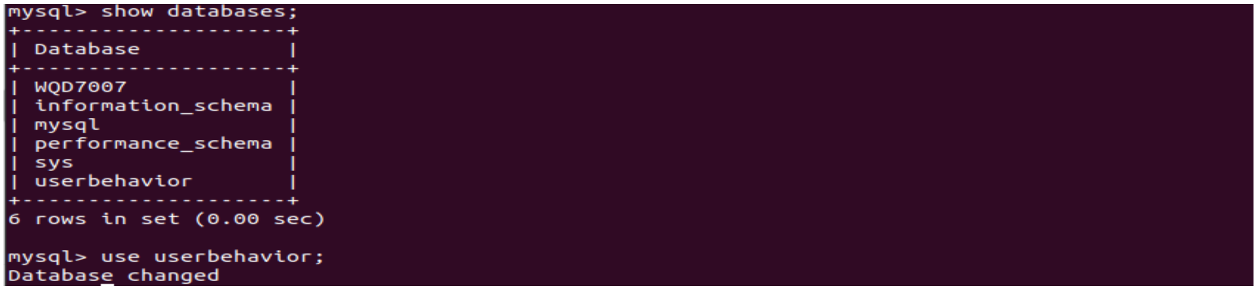
Figure 8: mongoimport the data



```
> show dbs
admin         0.000GB
casestudy     0.001GB
config        0.000GB
firstdb       0.000GB
groupproject  0.000GB
local         0.000GB
school        0.000GB
> db.userbehavior.find()
> use groupproject
switched to db groupproject
> db.userbehavior.find()
{ "_id" : ObjectId("62bb022889b50667d320b091"), "1000169" : 1, "1328010" : 2268318, "959452" : 2520377, "pv" : "pv", "2017-09-11 16:16:39" : "2017-11-25 01:21:10", "2017-09-11" : "2017-11-25" }
{ "_id" : ObjectId("62bb022889b50667d320b092"), "1000169" : 1, "1328010" : 2333346, "959452" : 2520771, "pv" : "pv", "2017-09-11 16:16:39" : "2017-11-25 06:15:33", "2017-09-11" : "2017-11-25" }
{ "_id" : ObjectId("62bb022889b50667d320b093"), "1000169" : 1, "1328010" : 2576651, "959452" : 149192, "pv" : "pv", "2017-09-11 16:16:39" : "2017-11-25 09:21:25", "2017-09-11" : "2017-11-25" }
{ "_id" : ObjectId("62bb022889b50667d320b094"), "1000169" : 1, "1328010" : 3830808, "959452" : 4181361, "pv" : "pv", "2017-09-11 16:16:39" : "2017-11-25 15:04:53", "2017-09-11" : "2017-11-25" }
{ "_id" : ObjectId("62bb022889b50667d320b095"), "1000169" : 1, "1328010" : 4365585, "959452" : 2520377, "pv" : "pv", "2017-09-11 16:16:39" : "2017-11-25 15:49:06", "2017-09-11" : "2017-11-25" }
{ "_id" : ObjectId("62bb022889b50667d320b096"), "1000169" : 1, "1328010" : 4606018, "959452" : 2735466, "pv" : "pv", "2017-09-11 16:16:39" : "2017-11-25 21:28:01", "2017-09-11" : "2017-11-25" }
{ "_id" : ObjectId("62bb022889b50667d320b097"), "1000169" : 100, "1328010" : 4572582, "959452" : 2188684, "pv" : "pv", "2017-09-11 16:16:39" : "2017-11-25 03:08:36", "2017-09-11" : "2017-11-25" }
{ "_id" : ObjectId("62bb022889b50667d320b098"), "1000169" : 100, "1328010" : 2971043, "959452" : 4869428, "pv" : "pv", "2017-09-11 16:16:39" : "2017-11-25 03:10:00", "2017-09-11" : "2017-11-25" }
{ "_id" : ObjectId("62bb022889b50667d320b099"), "1000169" : 100, "1328010" : 2379198, "959452" : 4869428, "pv" : "pv", "2017-09-11 16:16:39" : "2017-11-25 03:13:26", "2017-09-11" : "2017-11-25" }
{ "_id" : ObjectId("62bb022889b50667d320b09a"), "1000169" : 100, "1328010" : 2971043, "959452" : 4869428, "pv" : "pv", "2017-09-11 16:16:39" : "2017-11-25 03:14:45", "2017-09-11" : "2017-11-25" }
{ "_id" : ObjectId("62bb022889b50667d320b09b"), "1000169" : 100, "1328010" : 1220136, "959452" : 4869428, "pv" : "pv", "2017-09-11 16:16:39" : "2017-11-25 03:15:08", "2017-09-11" : "2017-11-25" }
{ "_id" : ObjectId("62bb022889b50667d320b09c"), "1000169" : 100, "1328010" : 2518420, "959452" : 3425094, "pv" : "pv", "2017-09-11 16:16:39" : "2017-11-25 03:24:03", "2017-09-11" : "2017-11-25" }
{ "_id" : ObjectId("62bb022889b50667d320b09d"), "1000169" : 100, "1328010" : 3763048, "959452" : 3425094, "pv" : "pv", "2017-09-11 16:16:39" : "2017-11-25 03:24:10", "2017-09-11" : "2017-11-25" }
{ "_id" : ObjectId("62bb022889b50667d320b09e"), "1000169" : 100, "1328010" : 2518420, "959452" : 3425094, "pv" : "pv", "2017-09-11 16:16:39" : "2017-11-25 03:24:55", "2017-09-11" : "2017-11-25" }
{ "_id" : ObjectId("62bb022889b50667d320b09f"), "1000169" : 100, "1328010" : 1953042, "959452" : 3425094, "pv" : "pv", "2017-09-11 16:16:39" : "2017-11-25 03:26:59", "2017-09-11" : "2017-11-25" }
{ "_id" : ObjectId("62bb022889b50667d320b0a0"), "1000169" : 100, "1328010" : 3763048, "959452" : 3425094, "pv" : "fav", "2017-09-11 16:16:39" : "2017-11-25 03:31:00", "2017-09-11" : "2017-11-25" }
{ "_id" : ObjectId("62bb022889b50667d320b0a1"), "1000169" : 100, "1328010" : 5100093, "959452" : 2945933, "pv" : "pv", "2017-09-11 16:16:39" : "2017-11-25 03:39:12", "2017-09-11" : "2017-11-25" }
{ "_id" : ObjectId("62bb022889b50667d320b0a2"), "1000169" : 100, "1328010" : 704268, "959452" : 223690, "pv" : "pv", "2017-09-11 16:16:39" : "2017-11-25 06:46:46", "2017-09-11" : "2017-11-25" }
{ "_id" : ObjectId("62bb022889b50667d320b0a3"), "1000169" : 100, "1328010" : 4115850, "959452" : 223690, "pv" : "fav", "2017-09-11 16:16:39" : "2017-11-25 06:50:34", "2017-09-11" : "2017-11-25" }
{ "_id" : ObjectId("62bb022889b50667d320b0a4"), "1000169" : 100, "1328010" : 2379198, "959452" : 4869428, "pv" : "pv", "2017-09-11 16:16:39" : "2017-11-25 07:09:21", "2017-09-11" : "2017-11-25" }
Type "it" for more
```

Figure 9: Sample of imported data to MongoDB

APPENDIX G

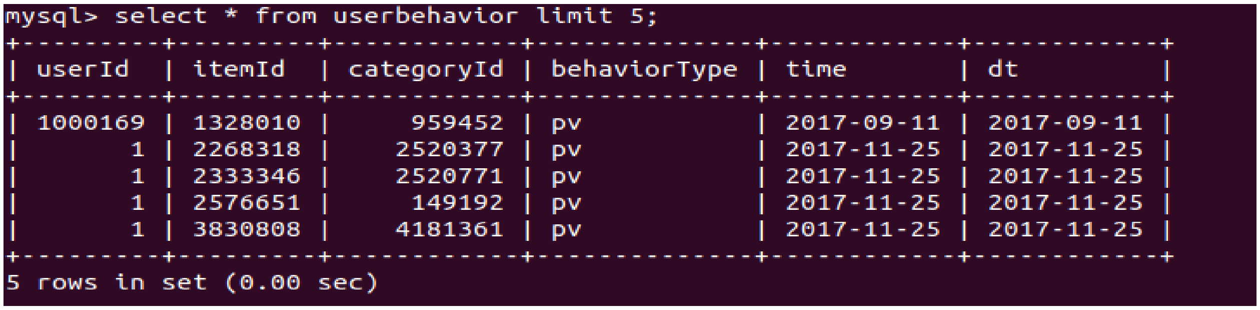


```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| WQD7007            |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| userbehavior       |
+--------------------+
6 rows in set (0.00 sec)

mysql> use userbehavior;
Database changed
```

Figure 10: Create database in MySQL

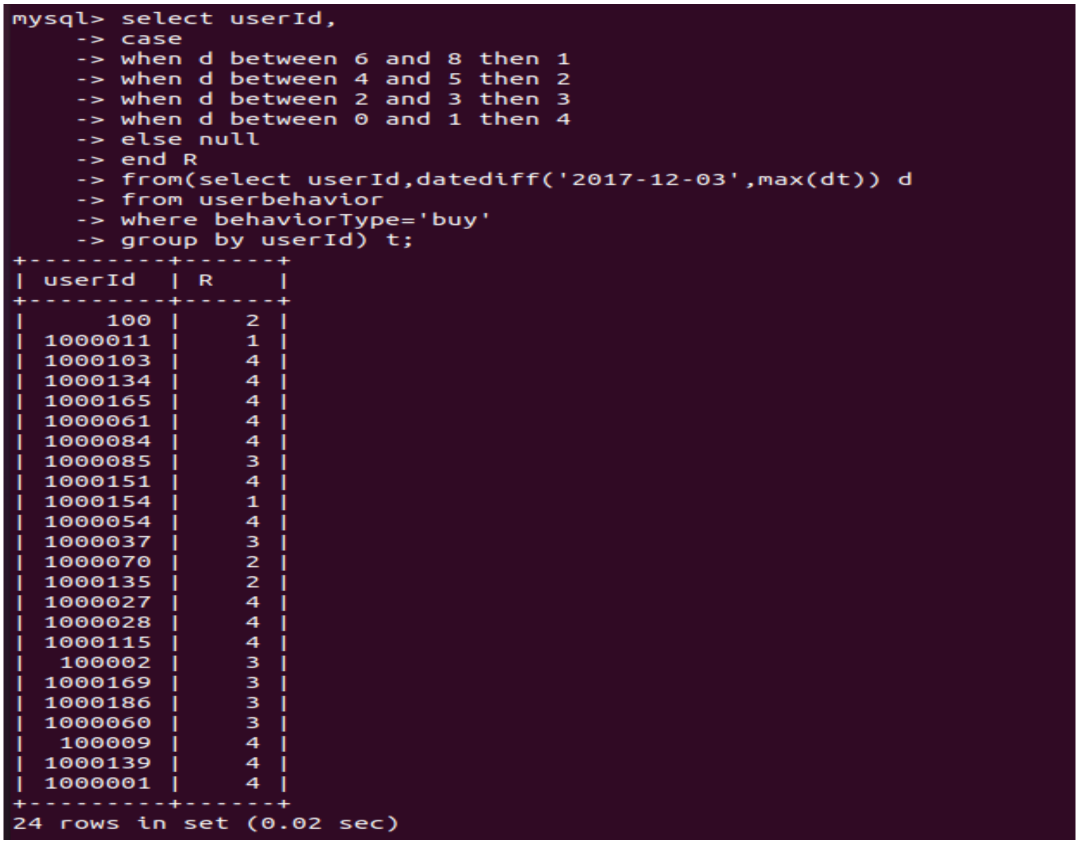
APPENDIX H



```
mysql> select * from userbehavior limit 5;
+---------+---------+------------+--------------+------------+------------+
| userId  | itemId  | categoryId | behaviorType | time       | dt         |
+---------+---------+------------+--------------+------------+------------+
| 1000169 | 1328010 |     959452 | pv           | 2017-09-11 | 2017-09-11 |
|       1 | 2268318 |    2520377 | pv           | 2017-11-25 | 2017-11-25 |
|       1 | 2333346 |    2520771 | pv           | 2017-11-25 | 2017-11-25 |
|       1 | 2576651 |     149192 | pv           | 2017-11-25 | 2017-11-25 |
|       1 | 3830808 |    4181361 | pv           | 2017-11-25 | 2017-11-25 |
+---------+---------+------------+--------------+------------+------------+
5 rows in set (0.00 sec)
```

Figure 11: Sample of uploaded table to MySQL

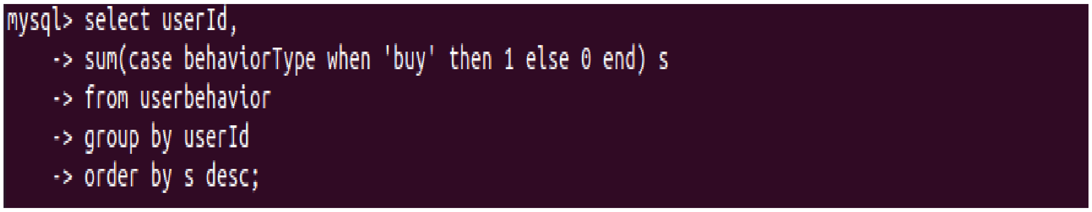
APPENDIX I



```
mysql> select userId,
    -> case
    -> when d between 6 and 8 then 1
    -> when d between 4 and 5 then 2
    -> when d between 2 and 3 then 3
    -> when d between 0 and 1 then 4
    -> else null
    -> end R
    -> from(select userId,datediff('2017-12-03',max(dt)) d
    -> from userbehavior
    -> where behaviorType='buy'
    -> group by userId) t;
+---------+------+
| userId  | R    |
+---------+------+
|     100 |    2 |
| 1000011 |    1 |
| 1000103 |    4 |
| 1000134 |    4 |
| 1000165 |    4 |
| 1000061 |    4 |
| 1000084 |    4 |
| 1000085 |    3 |
| 1000151 |    4 |
| 1000154 |    1 |
| 1000054 |    4 |
| 1000037 |    3 |
| 1000070 |    2 |
| 1000135 |    2 |
| 1000027 |    4 |
| 1000028 |    4 |
| 1000115 |    4 |
|  100002 |    3 |
| 1000169 |    3 |
| 1000186 |    3 |
| 1000060 |    3 |
|  100009 |    4 |
| 1000139 |    4 |
| 1000001 |    4 |
+---------+------+
24 rows in set (0.02 sec)
```

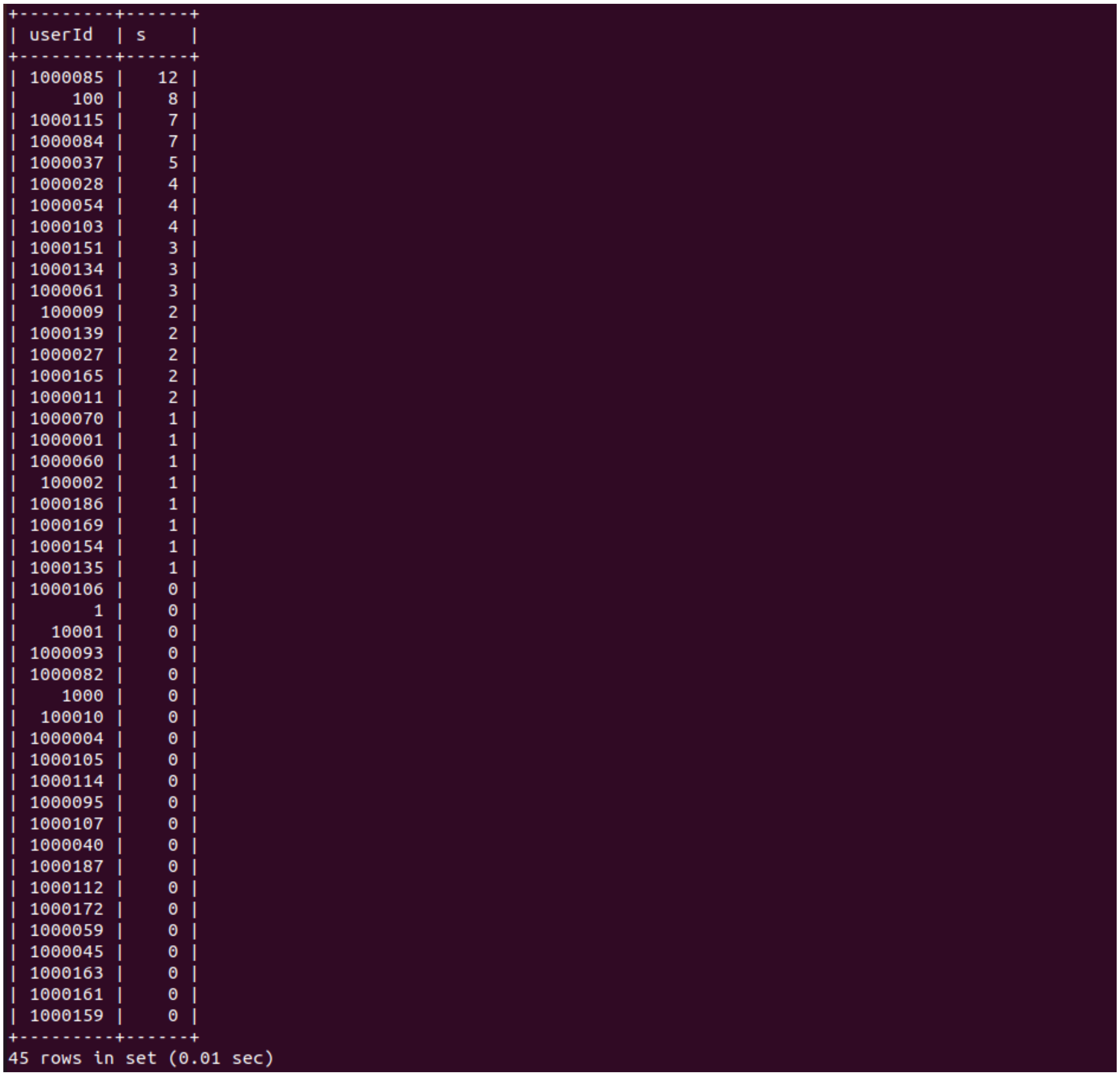
Figure 12: Last Purchase Time (MySQL)

APPENDIX J



```
mysql> select userId,
    -> sum(case behaviorType when 'buy' then 1 else 0 end) s
    -> from userbehavior
    -> group by userId
    -> order by s desc;
```

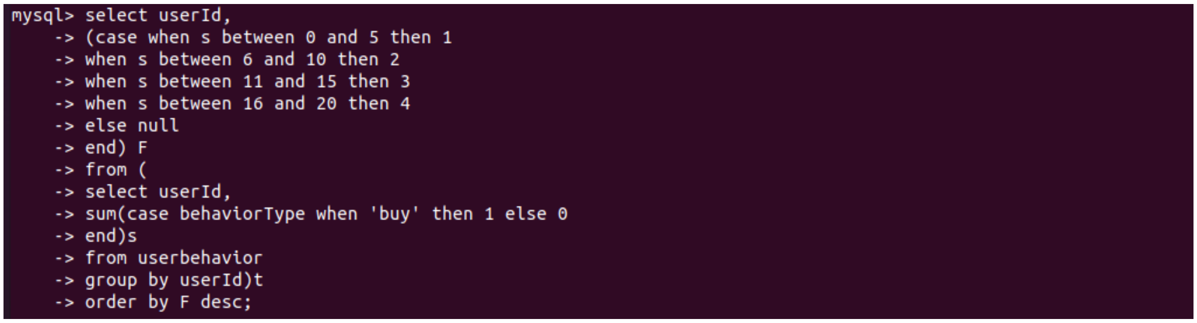
Figure 13: Query for the most purchased by the customers (MySQL)



```
+---------+------+
| userId  | s    |
+---------+------+
| 1000085 |   12 |
|     100 |    8 |
| 1000115 |    7 |
| 1000084 |    7 |
| 1000037 |    5 |
| 1000028 |    4 |
| 1000054 |    4 |
| 1000103 |    4 |
| 1000151 |    3 |
| 1000134 |    3 |
| 1000061 |    3 |
|  100009 |    2 |
| 1000139 |    2 |
| 1000027 |    2 |
| 1000165 |    2 |
| 1000011 |    2 |
| 1000070 |    1 |
| 1000001 |    1 |
| 1000060 |    1 |
|  100002 |    1 |
| 1000186 |    1 |
| 1000169 |    1 |
| 1000154 |    1 |
| 1000135 |    1 |
| 1000106 |    0 |
|       1 |    0 |
|   10001 |    0 |
| 1000093 |    0 |
| 1000082 |    0 |
|    1000 |    0 |
|  100010 |    0 |
| 1000004 |    0 |
| 1000105 |    0 |
| 1000114 |    0 |
| 1000095 |    0 |
| 1000107 |    0 |
| 1000040 |    0 |
| 1000187 |    0 |
| 1000112 |    0 |
| 1000172 |    0 |
| 1000059 |    0 |
| 1000045 |    0 |
| 1000163 |    0 |
| 1000161 |    0 |
| 1000159 |    0 |
+---------+------+
45 rows in set (0.01 sec)
```

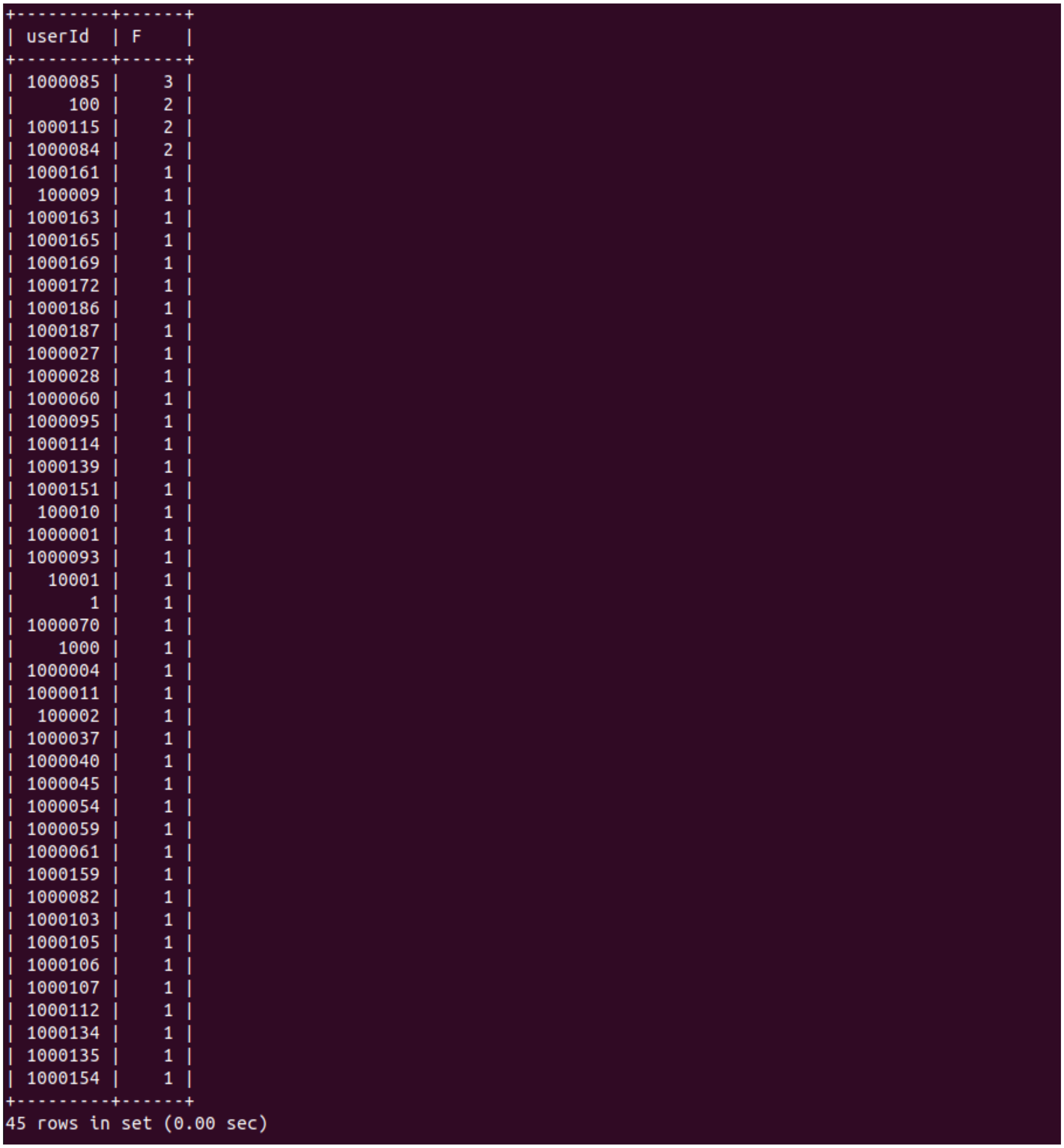
Figure 14: Result for the most purchased by the customers (MySQL)

APPENDIX K



```
mysql> select userId,
    -> (case when s between 0 and 5 then 1
    -> when s between 6 and 10 then 2
    -> when s between 11 and 15 then 3
    -> when s between 16 and 20 then 4
    -> else null
    -> end) F
    -> from (
    -> select userId,
    -> sum(case behaviorType when 'buy' then 1 else 0
    -> end)s
    -> from userbehavior
    -> group by userId)t
    -> order by F desc;
```

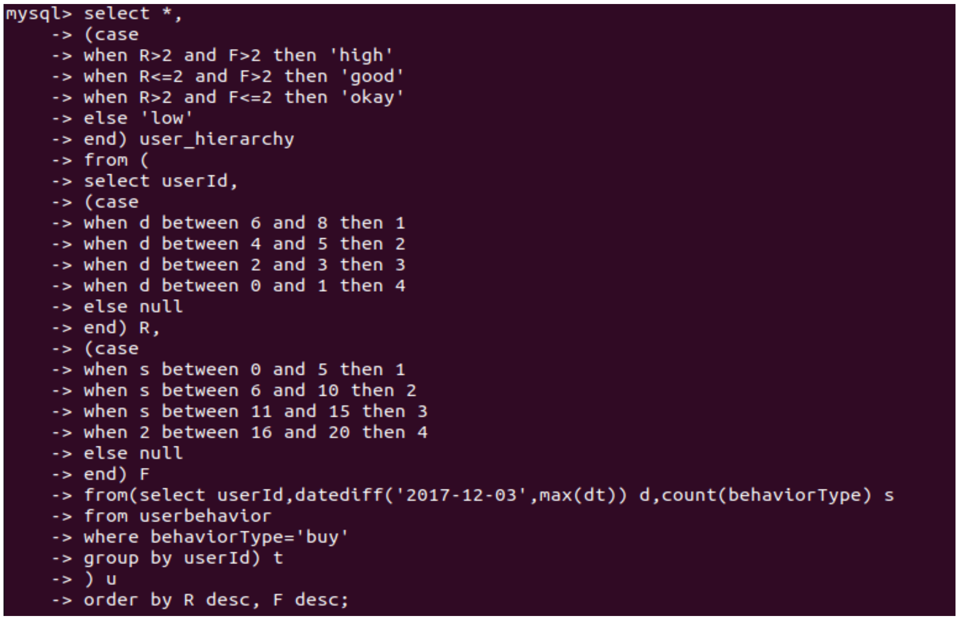
Figure 15: Class for Most frequent buy (MySQL)



```
+---------+------+
| userId  | F    |
+---------+------+
| 1000085 |    3 |
|     100 |    2 |
| 1000115 |    2 |
| 1000084 |    2 |
| 1000161 |    1 |
|  100009 |    1 |
| 1000163 |    1 |
| 1000165 |    1 |
| 1000169 |    1 |
| 1000172 |    1 |
| 1000186 |    1 |
| 1000187 |    1 |
| 1000027 |    1 |
| 1000028 |    1 |
| 1000060 |    1 |
| 1000095 |    1 |
| 1000114 |    1 |
| 1000139 |    1 |
| 1000151 |    1 |
|  100010 |    1 |
| 1000001 |    1 |
| 1000093 |    1 |
|   10001 |    1 |
|       1 |    1 |
| 1000070 |    1 |
|    1000 |    1 |
| 1000004 |    1 |
| 1000011 |    1 |
|  100002 |    1 |
| 1000037 |    1 |
| 1000040 |    1 |
| 1000045 |    1 |
| 1000054 |    1 |
| 1000059 |    1 |
| 1000061 |    1 |
| 1000159 |    1 |
| 1000082 |    1 |
| 1000103 |    1 |
| 1000105 |    1 |
| 1000106 |    1 |
| 1000107 |    1 |
| 1000112 |    1 |
| 1000134 |    1 |
| 1000135 |    1 |
| 1000154 |    1 |
+---------+------+
45 rows in set (0.00 sec)
```

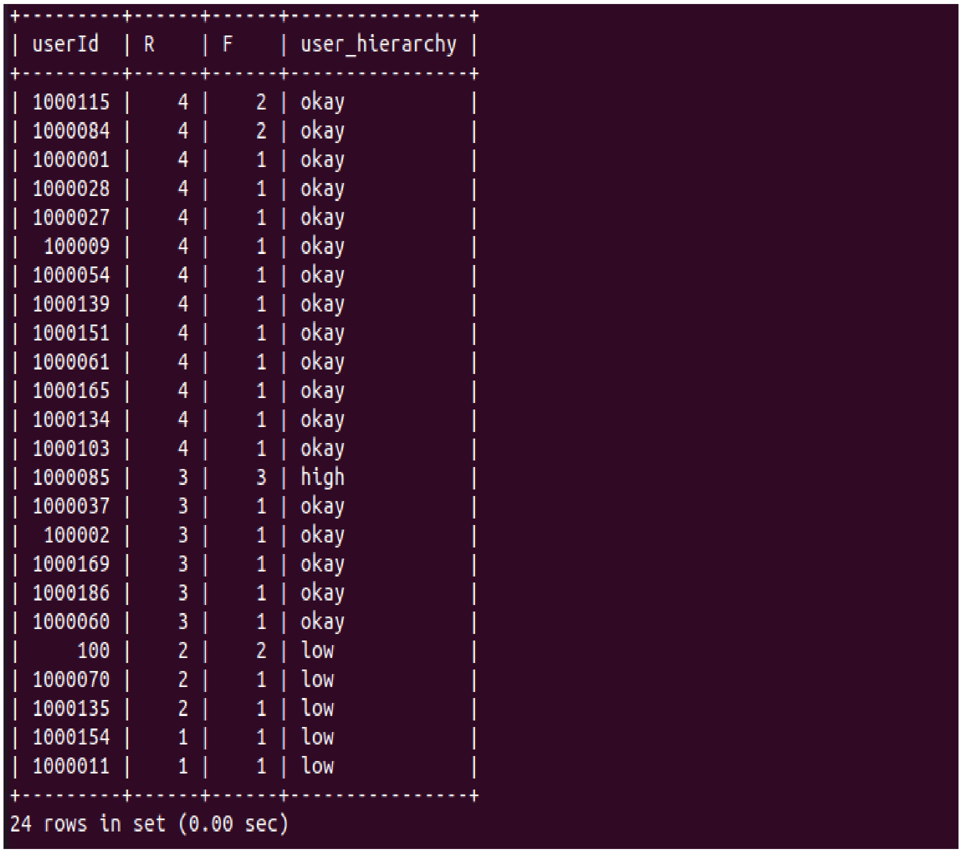
Figure 16: Result for most frequent buy class (MySQL)

APPENDIX L



```
mysql> select *,
    -> (case
    -> when R>2 and F>2 then 'high'
    -> when R<=2 and F>2 then 'good'
    -> when R>2 and F<=2 then 'okay'
    -> else 'low'
    -> end) user_hierarchy
    -> from (
    -> select userId,
    -> (case
    -> when d between 6 and 8 then 1
    -> when d between 4 and 5 then 2
    -> when d between 2 and 3 then 3
    -> when d between 0 and 1 then 4
    -> else null
    -> end) R,
    -> (case
    -> when s between 0 and 5 then 1
    -> when s between 6 and 10 then 2
    -> when s between 11 and 15 then 3
    -> when 2 between 16 and 20 then 4
    -> else null
    -> end) F
    -> from(select userId,datediff('2017-12-03',max(dt)) d,count(behaviorType) s
    -> from userbehavior
    -> where behaviorType='buy'
    -> group by userId) t
    -> ) u
    -> order by R desc, F desc;
```

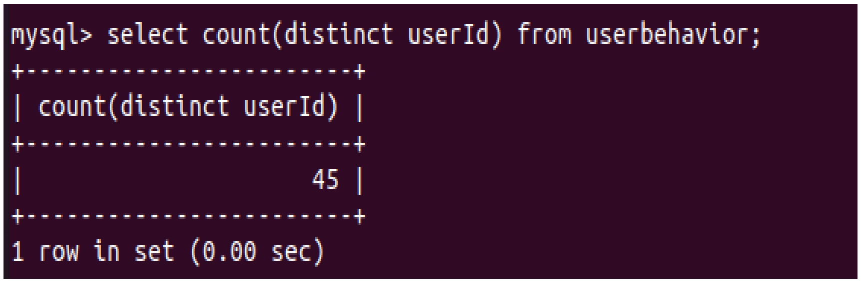
Figure 17: Query integrating R and F (MySQL)



```
+---------+------+------+----------------+
| userId  | R    | F    | user_hierarchy |
+---------+------+------+----------------+
| 1000115 |    4 |    2 | okay           |
| 1000084 |    4 |    2 | okay           |
| 1000001 |    4 |    1 | okay           |
| 1000028 |    4 |    1 | okay           |
| 1000027 |    4 |    1 | okay           |
|  100009 |    4 |    1 | okay           |
| 1000054 |    4 |    1 | okay           |
| 1000139 |    4 |    1 | okay           |
| 1000151 |    4 |    1 | okay           |
| 1000061 |    4 |    1 | okay           |
| 1000165 |    4 |    1 | okay           |
| 1000134 |    4 |    1 | okay           |
| 1000103 |    4 |    1 | okay           |
| 1000085 |    3 |    3 | high           |
| 1000037 |    3 |    1 | okay           |
|  100002 |    3 |    1 | okay           |
| 1000169 |    3 |    1 | okay           |
| 1000186 |    3 |    1 | okay           |
| 1000060 |    3 |    1 | okay           |
|     100 |    2 |    2 | low            |
| 1000070 |    2 |    1 | low            |
| 1000135 |    2 |    1 | low            |
| 1000154 |    1 |    1 | low            |
| 1000011 |    1 |    1 | low            |
+---------+------+------+----------------+
24 rows in set (0.00 sec)
```

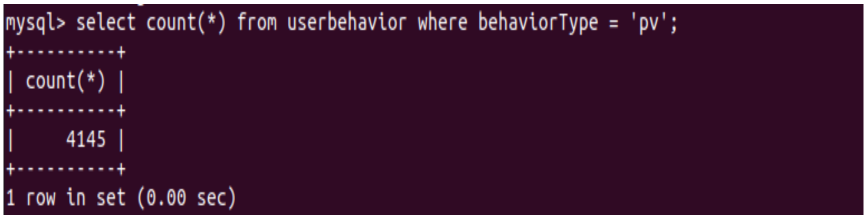
Figure 18: Result from the integration of R and F (MySQL)

APPENDIX M



```
mysql> select count(distinct userId) from userbehavior;
+------------------------+
| count(distinct userId) |
+------------------------+
|                     45 |
+------------------------+
1 row in set (0.00 sec)
```

Figure 19: Actual number of users

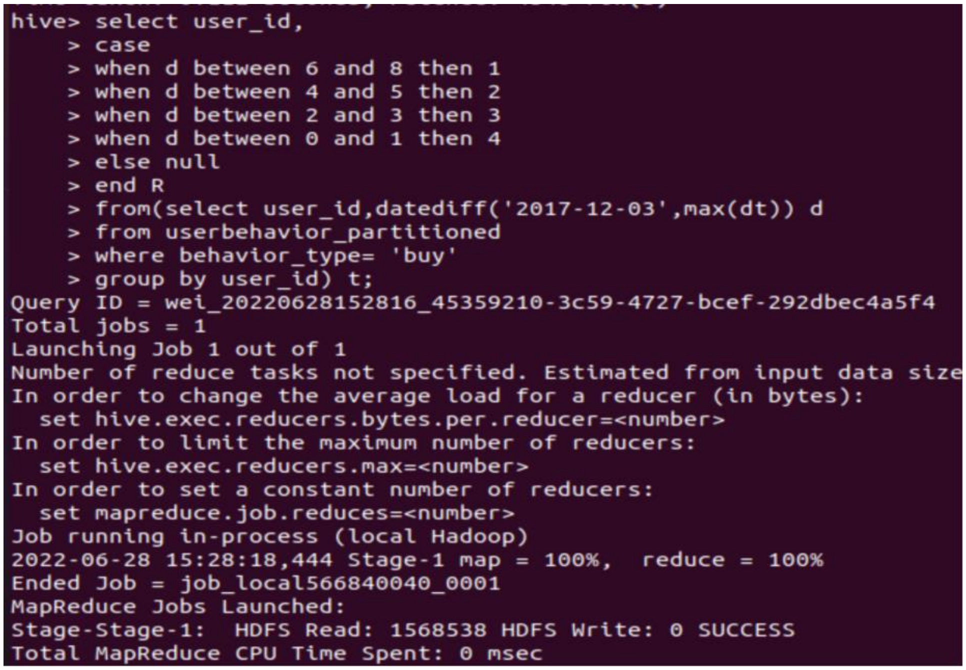


```
mysql> select count(*) from userbehavior where behaviorType = 'pv';
+----------+
| count(*) |
+----------+
|     4145 |
+----------+
1 row in set (0.00 sec)
```

Figure 20: Total number for PV

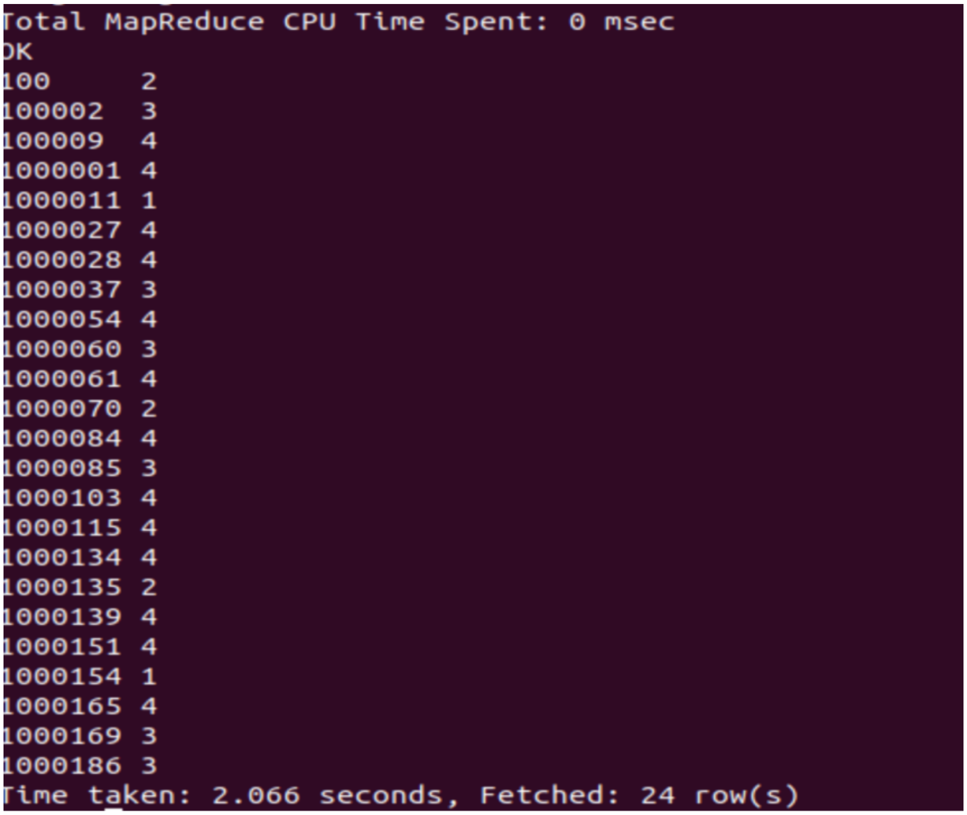
<u>Hive(To compare the results with MySQL in term of its computing time)</u>

APPENDIX N



```
hive> select user_id,
    > case
    > when d between 6 and 8 then 1
    > when d between 4 and 5 then 2
    > when d between 2 and 3 then 3
    > when d between 0 and 1 then 4
    > else null
    > end R
    > from(select user_id,datediff('2017-12-03',max(dt)) d
    > from userbehavior_partitioned
    > where behavior_type= 'buy'
    > group by user_id) t;
Query ID = wei_20220628152816_45359210-3c59-4727-bcef-292dbec4a5f4
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2022-06-28 15:28:18,444 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local566840040_0001
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 1568538 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
```

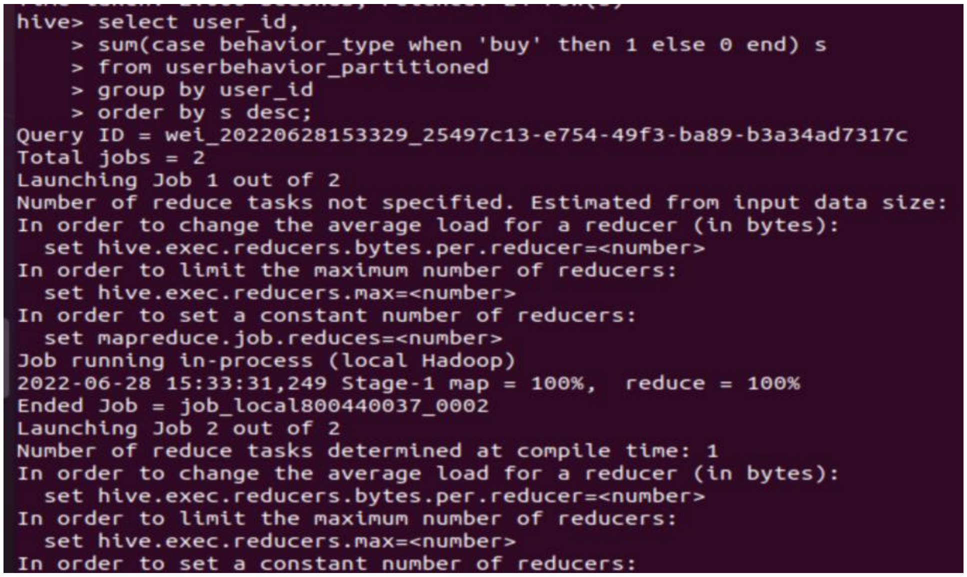
Figure 21: Last Purchase Time (Hive)



```
Total MapReduce CPU Time Spent: 0 msec
OK
100     2
100002  3
100009  4
1000001 4
1000011 1
1000027 4
1000028 4
1000037 3
1000054 4
1000060 3
1000061 4
1000070 2
1000084 4
1000085 3
1000103 4
1000115 4
1000134 4
1000135 2
1000139 4
1000151 4
1000154 1
1000165 4
1000169 3
1000186 3
Time taken: 2.066 seconds, Fetched: 24 row(s)
```

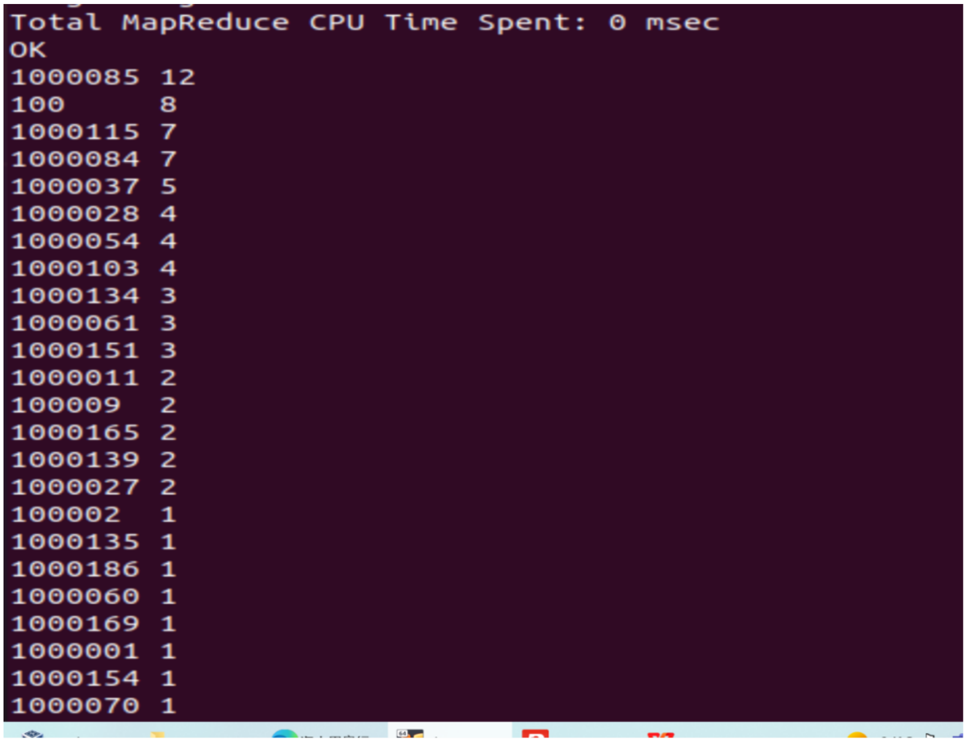
Figure 22: The results for Last Purchase Time (Hive)

From Appendix I, we know MySQL only spent 0.02 seconds to execute the select query, but Hive spent much longer time 2.066 seconds.



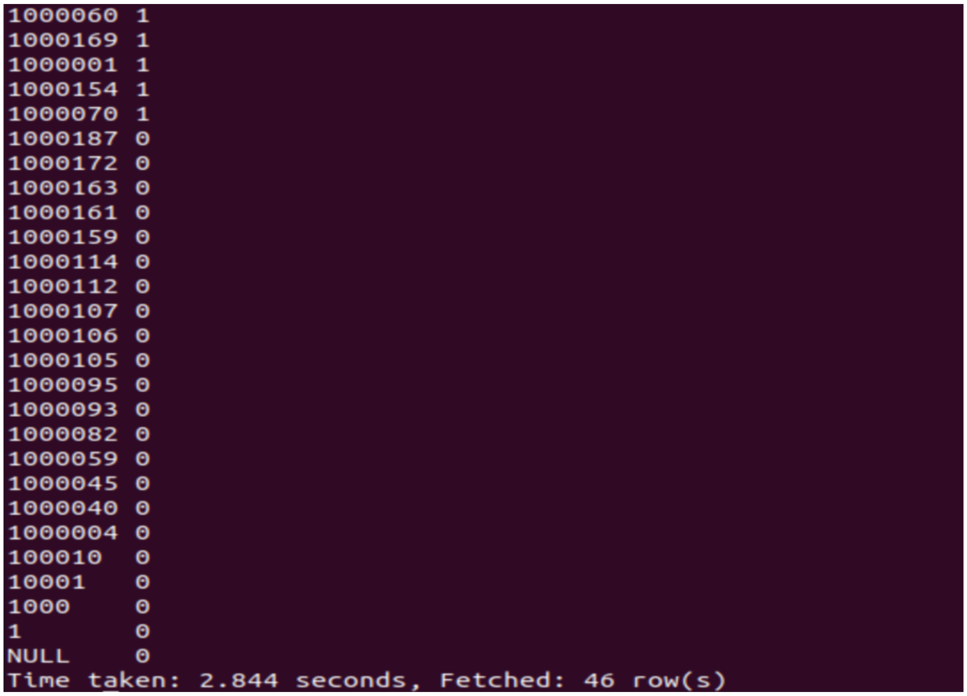
```
hive> select user_id,
    > sum(case behavior_type when 'buy' then 1 else 0 end) s
    > from userbehavior_partitioned
    > group by user_id
    > order by s desc;
Query ID = wei_20220628153329_25497c13-e754-49f3-ba89-b3a34ad7317c
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size:
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2022-06-28 15:33:31,249 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local800440037_0002
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
```

Figure 23: Query for the most purchased by the customers (Hive)



```
Total MapReduce CPU Time Spent: 0 msec
OK
1000085 12
100     8
1000115 7
1000084 7
1000037 5
1000028 4
1000054 4
1000103 4
1000134 3
1000061 3
1000151 3
1000011 2
100009  2
1000165 2
1000139 2
1000027 2
100002  1
1000135 1
1000186 1
1000060 1
1000169 1
1000001 1
1000154 1
1000070 1
```

Figure 24: Result for the most purchased by the customers -part 1 (Hive)



```
1000060 1
1000169 1
1000001 1
1000154 1
1000070 1
1000187 0
1000172 0
1000163 0
1000161 0
1000159 0
1000114 0
1000112 0
1000107 0
1000106 0
1000105 0
1000095 0
1000093 0
1000082 0
1000059 0
1000045 0
1000040 0
1000004 0
100010  0
10001   0
1000    0
1       0
NULL    0
Time taken: 2.844 seconds, Fetched: 46 row(s)
```

Figure 25: Result for the most purchased by the customers -part 2 (Hive)

From Appendix J, we know MySQL only spent 0.01 seconds to execute the select query, but Hive spent much longer time 2.844 seconds.