

**Name :-** Hashmeetsingh Obhan

**Student ID :-** 21262793

**Email ID :-** hashmeetsingh.obhan2@mail.dcu.ie

**Link for the Git Repository:-**

[**https://github.com/Hashmeetsingh/CloudTechnologies**](https://github.com/Hashmeetsingh/CloudTechnologies)

**Description of the dataset:-**

The data in the dataset comes from a social media app like Facebook or Instagram. It has several key columns, such as Id, Body, Score, UserName, and Tags, that provide detailed information on the user.

**Task 1:-** Get data from Stack Exchange (Data Acquisition/Collection)

* To acquire the data I used the Data Explorer feature of the StackExchange system using following link to run the below 4 queries.

**LINK:-** [**https://data.stackexchange.com/stackoverflow/query/new**](https://data.stackexchange.com/stackoverflow/query/new)

**Technology used:-** Used SQL like queries to fetch the data from Data Explorer feature of the StackExchange system. In the below 4 queries I have fetched mostly all the columns of POST table on the basis of ViewCount (ORDER BY ViewCount) in the descending order. I have used the in-built function of SQL like ROW\_NUMBER() as I need to fetch 50000 records at a time which made it easier to implement the task and filtered the query in the WHERE clause by RowNumber itself.

**Queries:-**

SELECT \* FROM (SELECT ROW\_NUMBER() OVER (ORDER BY VIEWCOUNT DESC) AS RowNumber, Id, PostTypeId, AcceptedAnswerId, ParentId, CreationDate, DeletionDate, Score, ViewCount, OwnerUserId, OwnerDisplayName, LastEditorUserId, LastEditorDisplayName, LastEditDate, LastActivityDate, Title, Tags, AnswerCount, CommentCount, FavoriteCount, ClosedDate, CommunityOwnedDate, ContentLicense FROM POSTS WHERE VIEWCOUNT IS NOT NULL) AS POSTSTABLE WHERE RowNumber BETWEEN 1 AND 50001

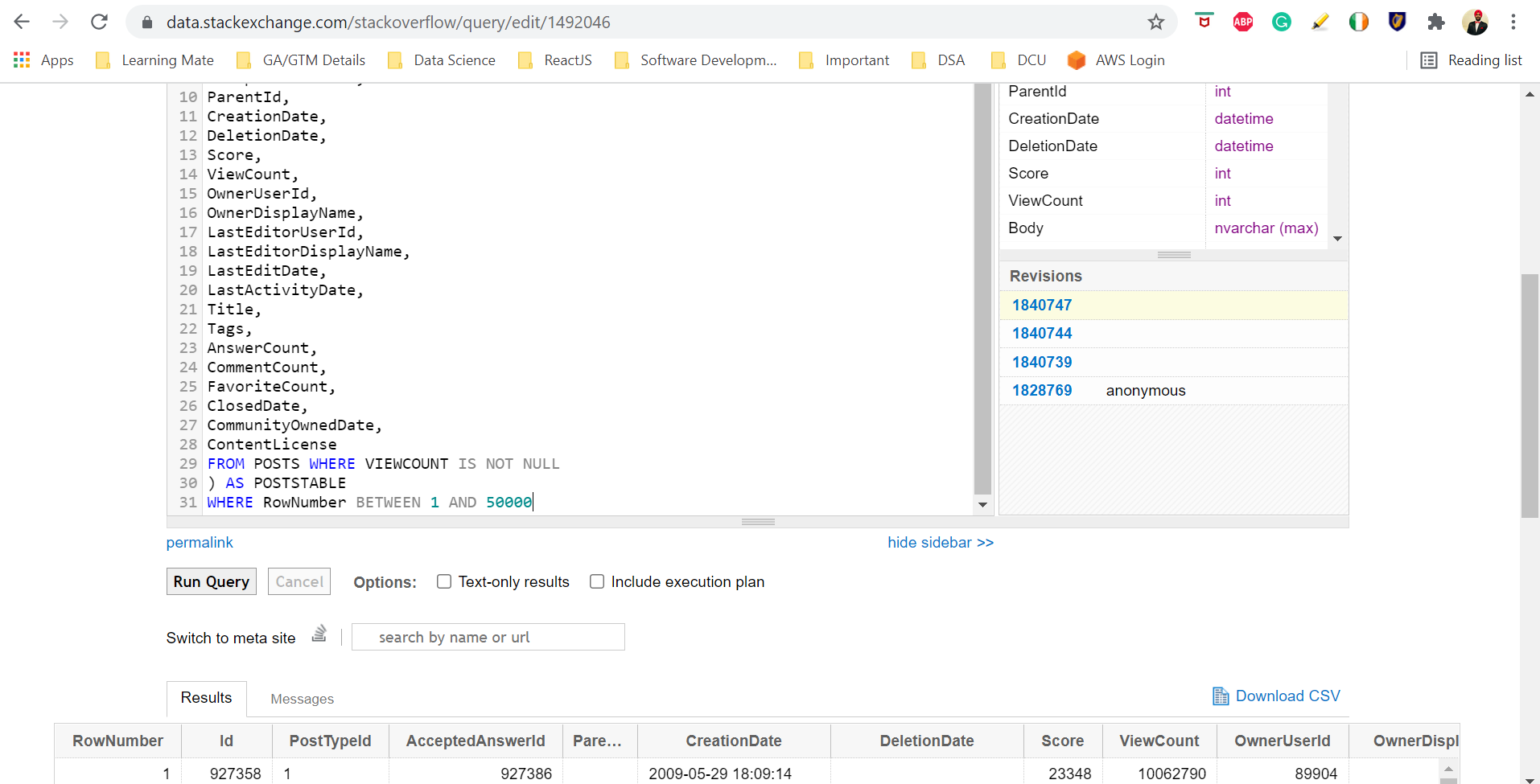


Figure:-1. 50k records of Posts table

SELECT \* FROM (SELECT ROW\_NUMBER() OVER (ORDER BY VIEWCOUNT DESC) AS RowNumber, Id, PostTypeId, AcceptedAnswerId, ParentId, CreationDate, DeletionDate, Score, ViewCount, OwnerUserId, OwnerDisplayName, LastEditorUserId, LastEditorDisplayName, LastEditDate, LastActivityDate, Title, Tags, AnswerCount, CommentCount, FavoriteCount, ClosedDate, CommunityOwnedDate, ContentLicense FROM POSTS WHERE VIEWCOUNT IS NOT NULL) AS POSTSTABLE WHERE RowNumber BETWEEN 50001 AND 100000

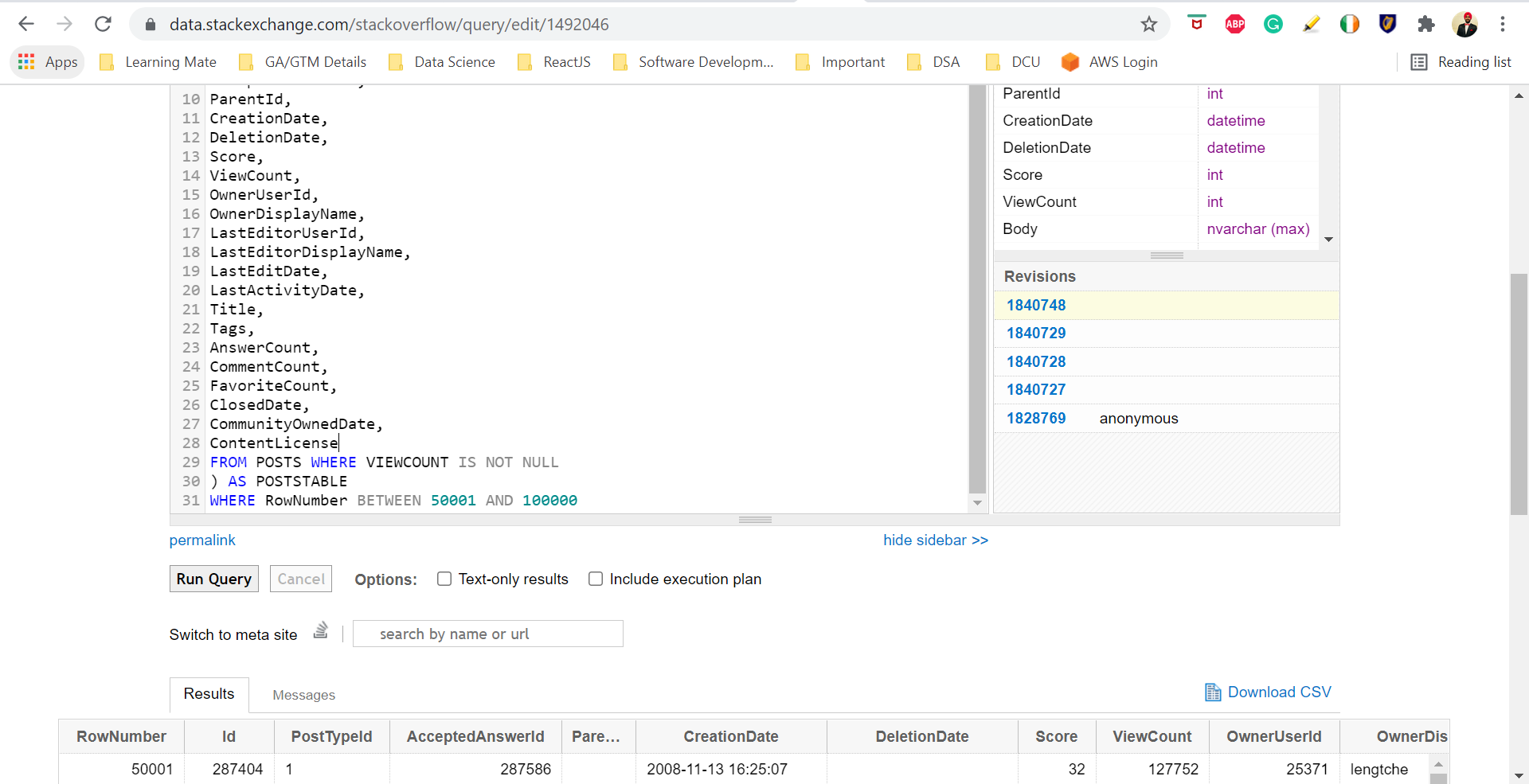


Figure:-2. 50k records of Posts table

SELECT \* FROM (SELECT ROW\_NUMBER() OVER (ORDER BY VIEWCOUNT DESC) AS RowNumber, Id, PostTypeId, AcceptedAnswerId, ParentId, CreationDate, DeletionDate, Score, ViewCount, OwnerUserId, OwnerDisplayName, LastEditorUserId, LastEditorDisplayName, LastEditDate, LastActivityDate, Title, Tags, AnswerCount, CommentCount, FavoriteCount, ClosedDate, CommunityOwnedDate, ContentLicense FROM POSTS WHERE VIEWCOUNT IS NOT NULL) AS POSTSTABLE WHERE RowNumber BETWEEN 100001 AND 150000

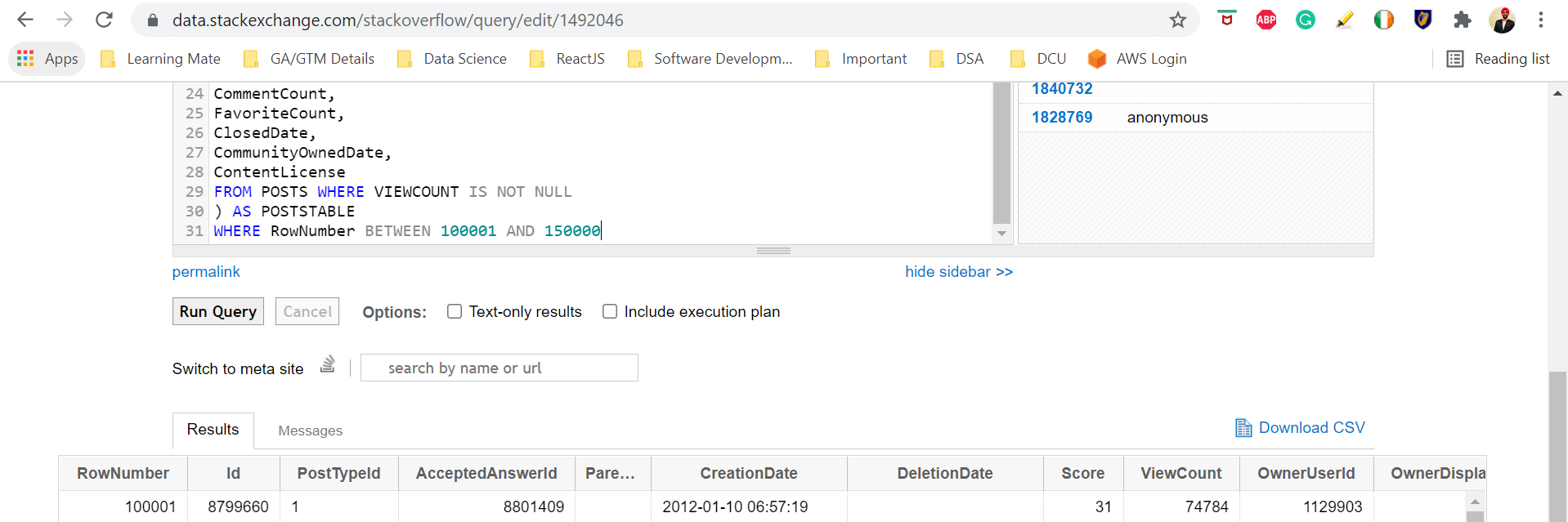


Figure:-3. 50k records of Posts table

SELECT \* FROM (SELECT ROW\_NUMBER() OVER (ORDER BY VIEWCOUNT DESC) AS RowNumber, Id, PostTypeId, AcceptedAnswerId, ParentId, CreationDate, DeletionDate, Score, ViewCount, OwnerUserId, OwnerDisplayName, LastEditorUserId, LastEditorDisplayName, LastEditDate, LastActivityDate, Title, Tags, AnswerCount, CommentCount, FavoriteCount, ClosedDate, CommunityOwnedDate, ContentLicense FROM POSTS WHERE VIEWCOUNT IS NOT NULL) AS POSTSTABLE WHERE RowNumber BETWEEN 150001 AND 200000

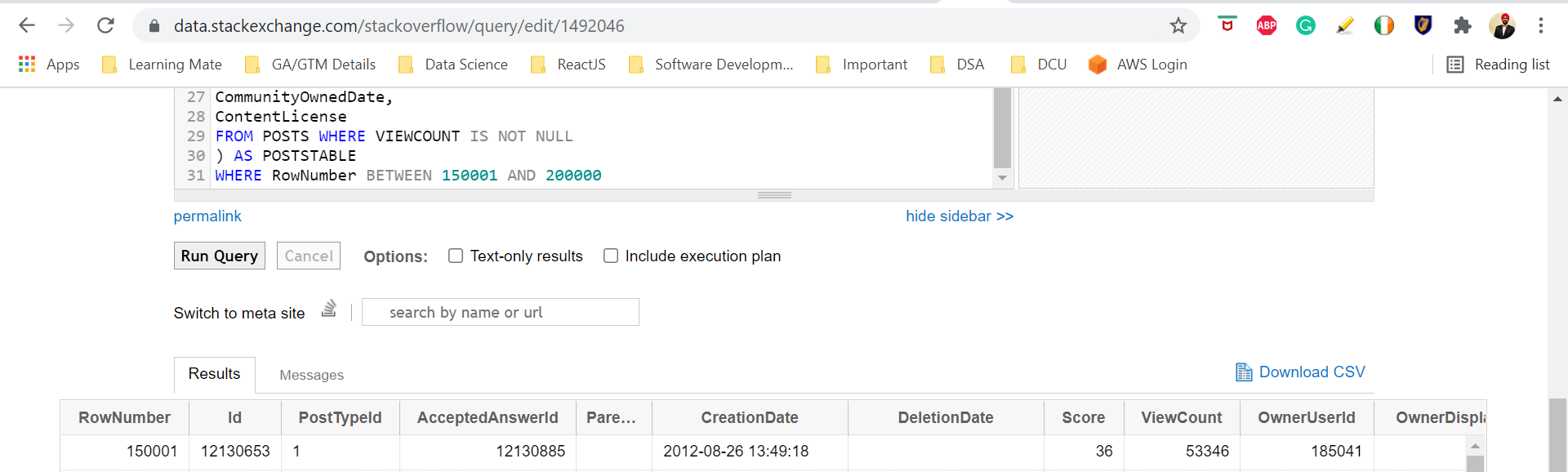


Figure:-4. 50k records of Posts table

* Following the execution of the above four queries, I downloaded the CSV files containing 50000 records each, and then combined them into a single file with 200000 records. After that I cleaned “**PostBodyData**” and “**PostTitle**” columns of that file to get the optimized result.

**Technology used:-** I used Python to clean the data as it is quite easy to clean the dataset using python and reduces complexity as well. **(6) (7) (8)**

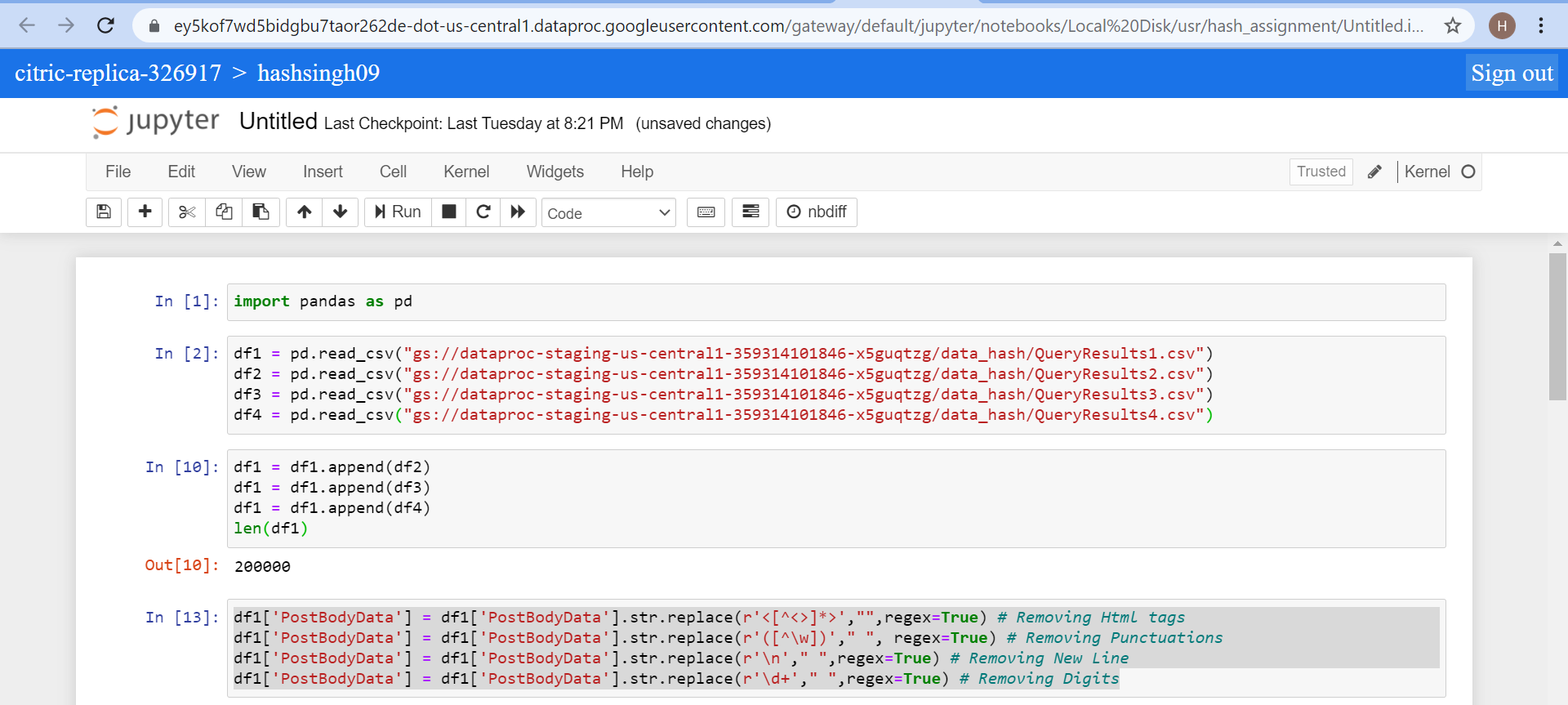


Figure:-5. Merging 4 dataset files into 1 single file

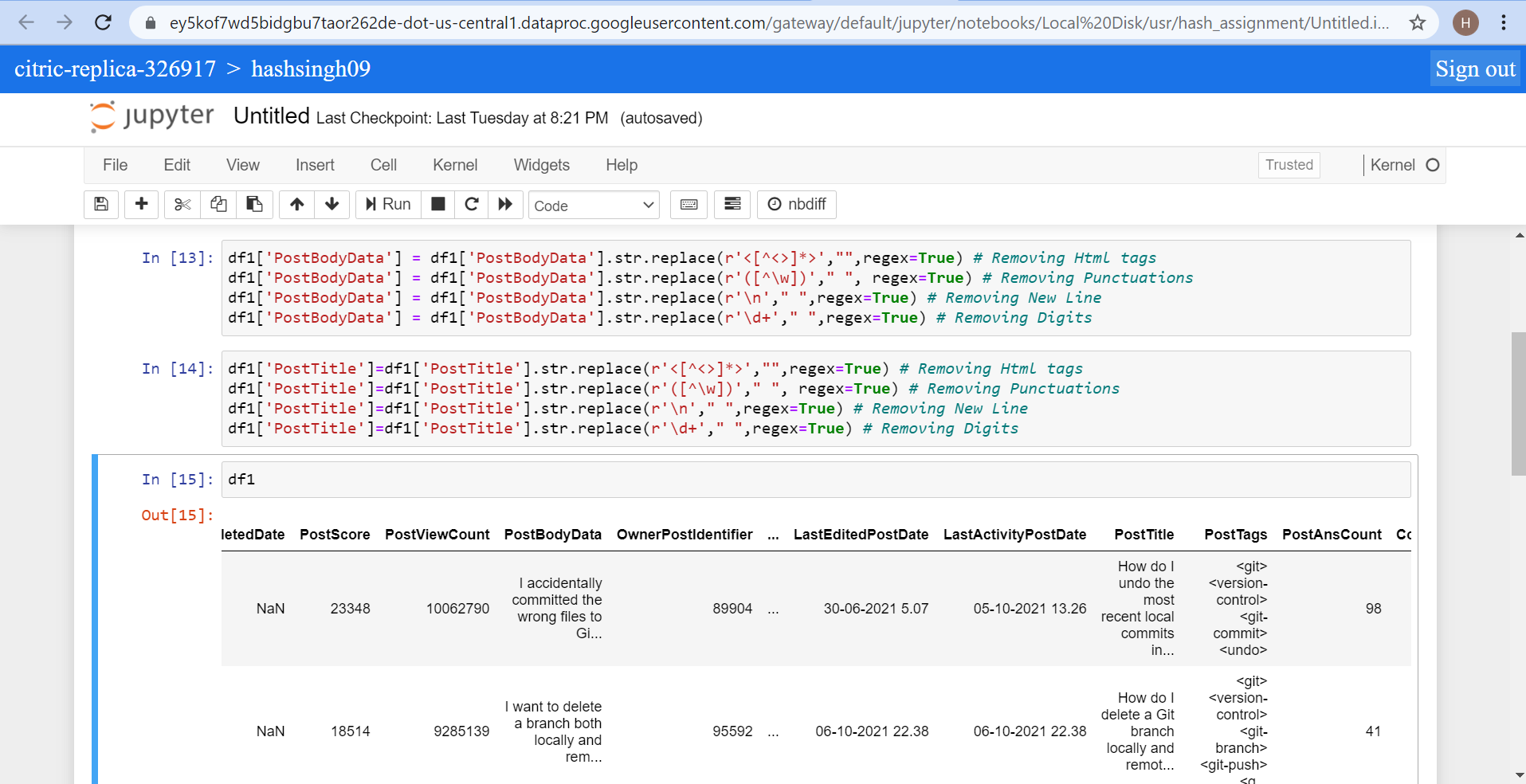


Figure:-6. Cleaning “PostBodyData” and “PostTitle” columns of the dataset

* I performed all the tasks on Google Cloud Platform (GCP) environment. Firstly, I created an instance on GCP having 1 namenode and 2 workers (2 datanodes).
* Then I uploaded the CSV file having 200000 records on GCP cloud storage as shown in below image.

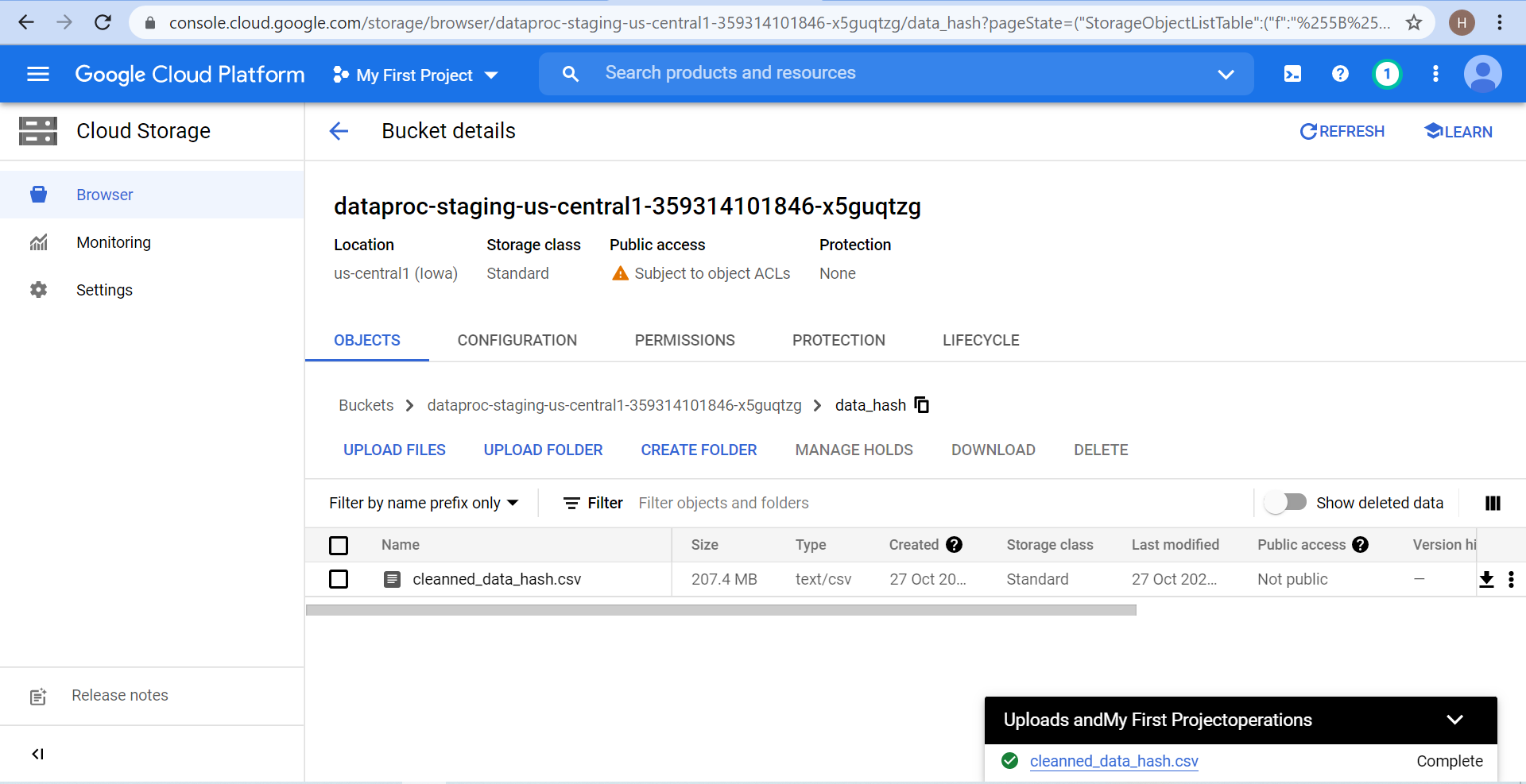


Figure:-7. Successfully uploaded the cleaned data file to GCP cloud

**Task 2:-** Load data into chosen cloud technology (Hive)

* Once uploading the file on GCP is completed I created a table on HIVE, which is pre-installed on GCP. Then I loaded the data in that table using the query. After that I created a View which has a similar structure as that of table where in I typecast few columns.

**Technology used:-** To import the data into the table, I utilized HIVE. This is owing to the fact that the HIVE is easier to implement than Pig and Mapreduce, requiring fewer development resources. Another reason I chose HIVE is that I am well familiar with SQL, which made it much easier to use. Similarly, I utilized the references described above to learn about linux commands and their functions, which I then implemented locally because JAVA, HADOOP, and HIVE were already installed on GCP, so I didn't have to do it again. I tried on AWS but got stuck in the middle and couldn't figure out how to fix the problem, then I went to GCP and everything worked fine. **(1)(2)(3)(5)**

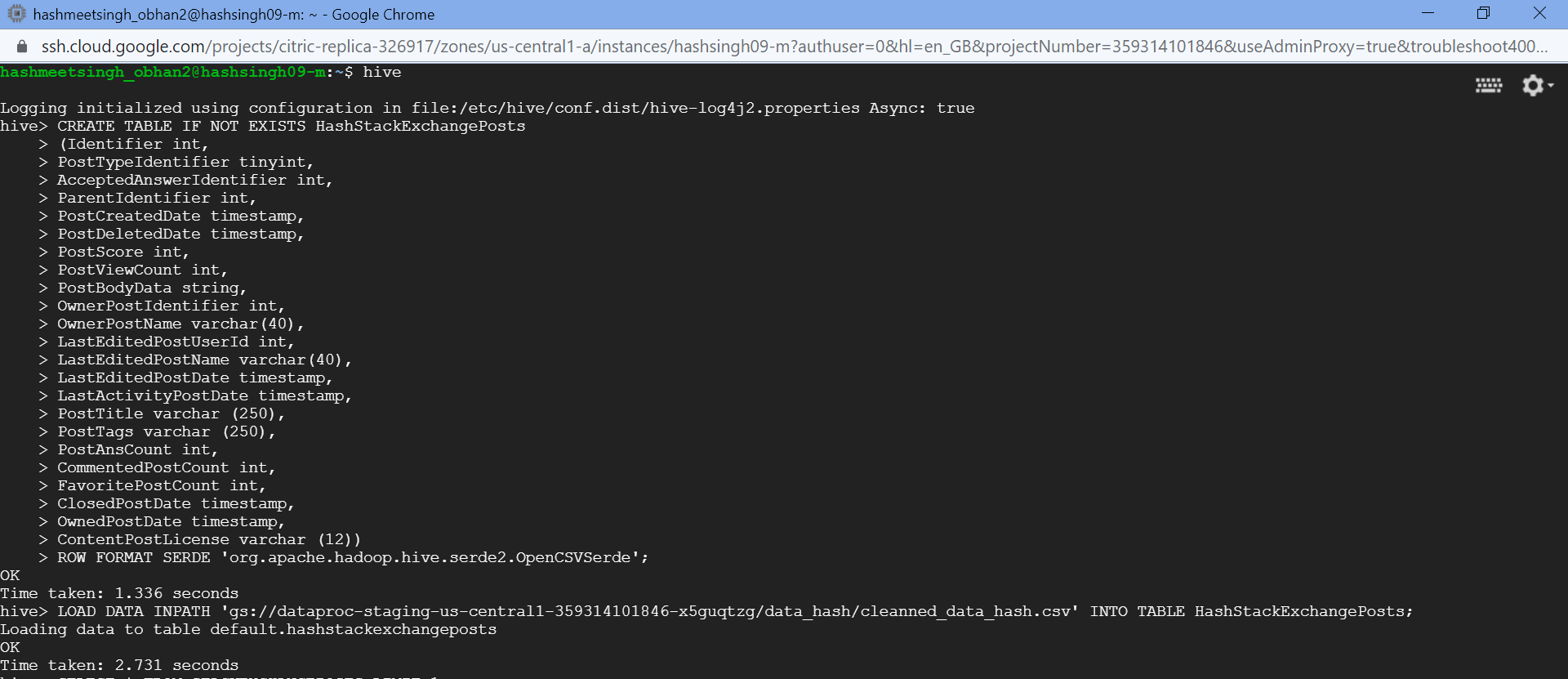


Figure:-8. Successfully loaded the data into table using HIVE

**Task 3:-** Run the Query data using Hive

**Technology used:-** Similarly like Task 2, I have used HIVE to execute the all the task 3 queries as they are similar to SQL like query language and easy to implement.

**3.1.** The top 10 posts by score

* To get the result I executed the below query on HIVE (GCP) where it returns the Id, Title and Score from the Posts table on the basis of Score in the descending order and fetched only 10 records so I have set the limit as 10.
* SELECT Identifier, PostTitle, PostScore from HashStackExchangeView ORDER BY PostScore DESC LIMIT 10;

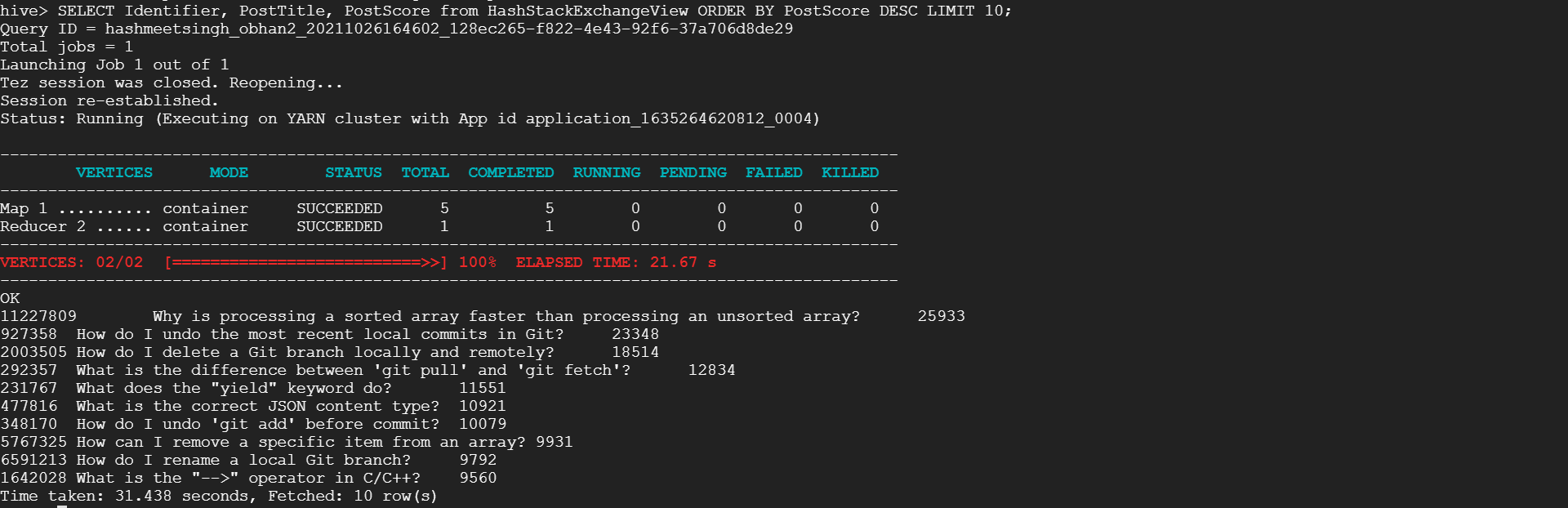


Figure:-9. Successfully executed the top 10 posts by score

**3.2.** The top 10 users by post score

* To get the result I executed the below query on HIVE (GCP) where it returns the user details like OwnerUserId, OwnerDisplayName and Score from the Posts table on the basis of Posts Score where the limit is set to 10 to fetch only 10 records.
* SELECT OwnerPostIdentifier, OwnerPostName, sum(PostScore) as PostScore from HashStackExchangeView GROUP BY OwnerPostIdentifier, OwnerPostName ORDER BY PostScore DESC LIMIT 10;

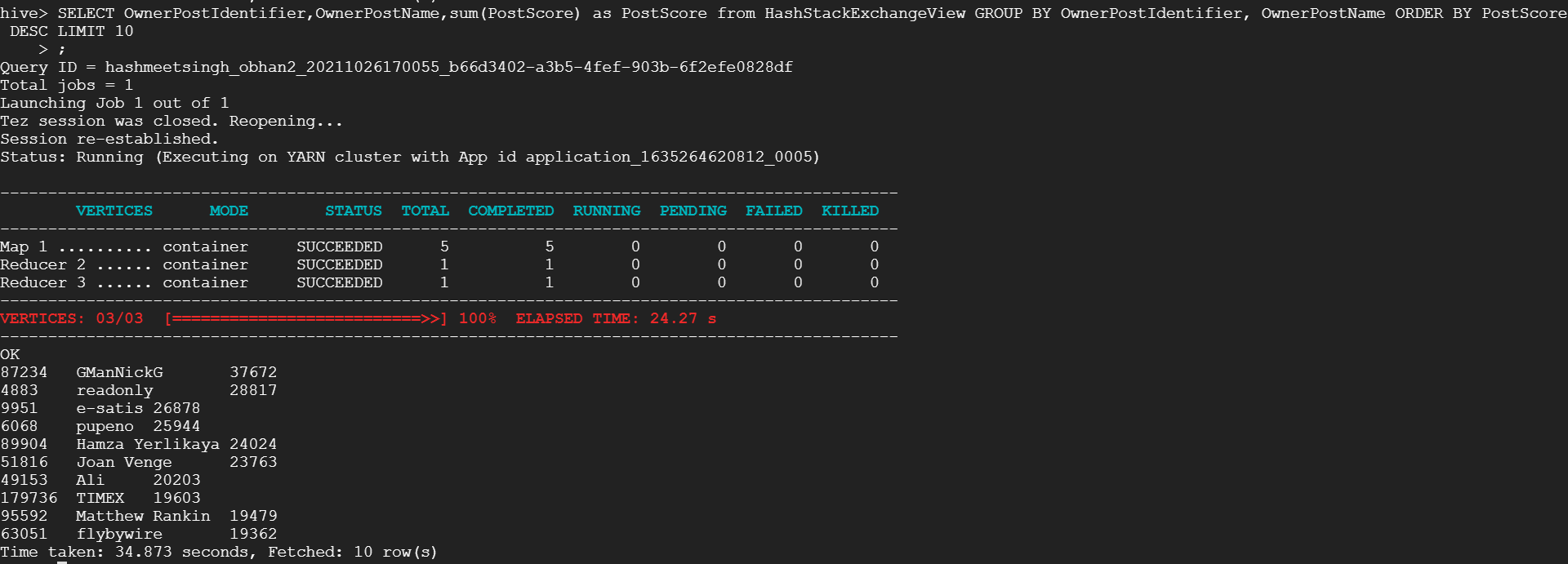
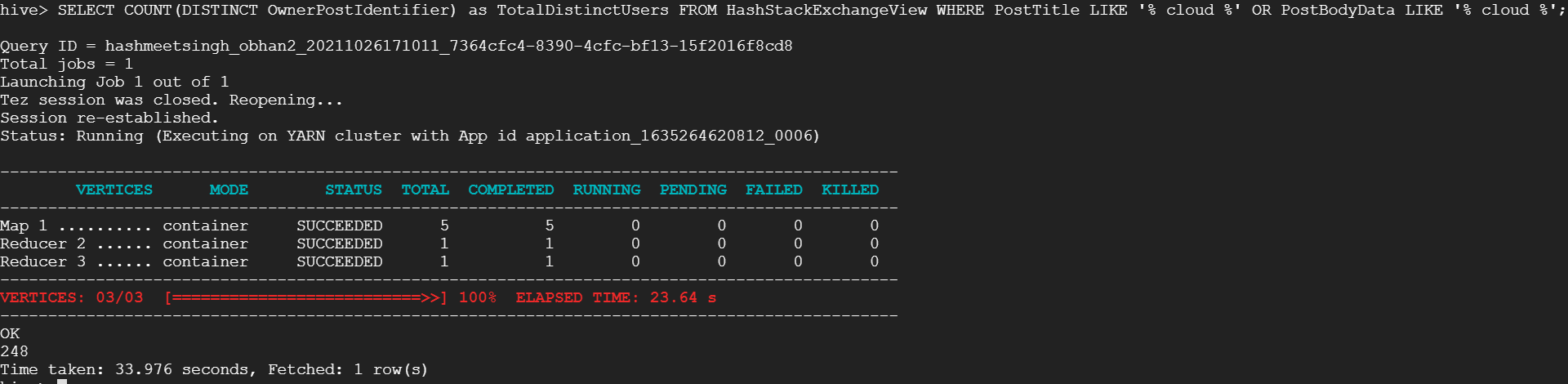


Figure:-10. Successfully executed the top 10 users by post score

**3.3.** The number of distinct users, who used the word “cloud” in one of their posts

* Similarly like above 2 queries, I executed the below query on HIVE (GCP) to obtain the result. Below query returns the number of distinct users which contains the word “cloud” either in their Body or Title column.
* SELECT COUNT(DISTINCT OwnerPostIdentifier) as TotalDistinctUsers FROM HashStackExchangeView WHERE PostTitle LIKE '% cloud %' OR PostBodyData LIKE '% cloud %';

 Figure:-11. Number of distinct users who used the word “cloud” in their posts Body/Title column

**Task 4:-** Calculate the per-user TF-IDF of the top 10 terms for each of the top 10 users.

**Technology used:-** This time I have used Python to find the top 10 terms for each of the 10 users as it was easy to develop and implement the code in python as compared to HIVE and Pig. Also, the code is much more efficient using Python and able to get the optimum result.

**Explanation of the code Snippet:-**

I started by installing the necessary libraries and packages. Then I created a connection to the HIVE database by passing the required authorization parameters to the connection string. Then, depending on their post score, I discovered the top ten user details. After that, I looked for OwnerPostIdentifier in the details I had retrieved earlier for 10 users. Last but not least, I used the sklearn library to find each user's TF IDF. Refer to Figures 9-10 and 11 for an example of how to find the TF IDF of the top 10 terms for each of the 10 users. **(4)**

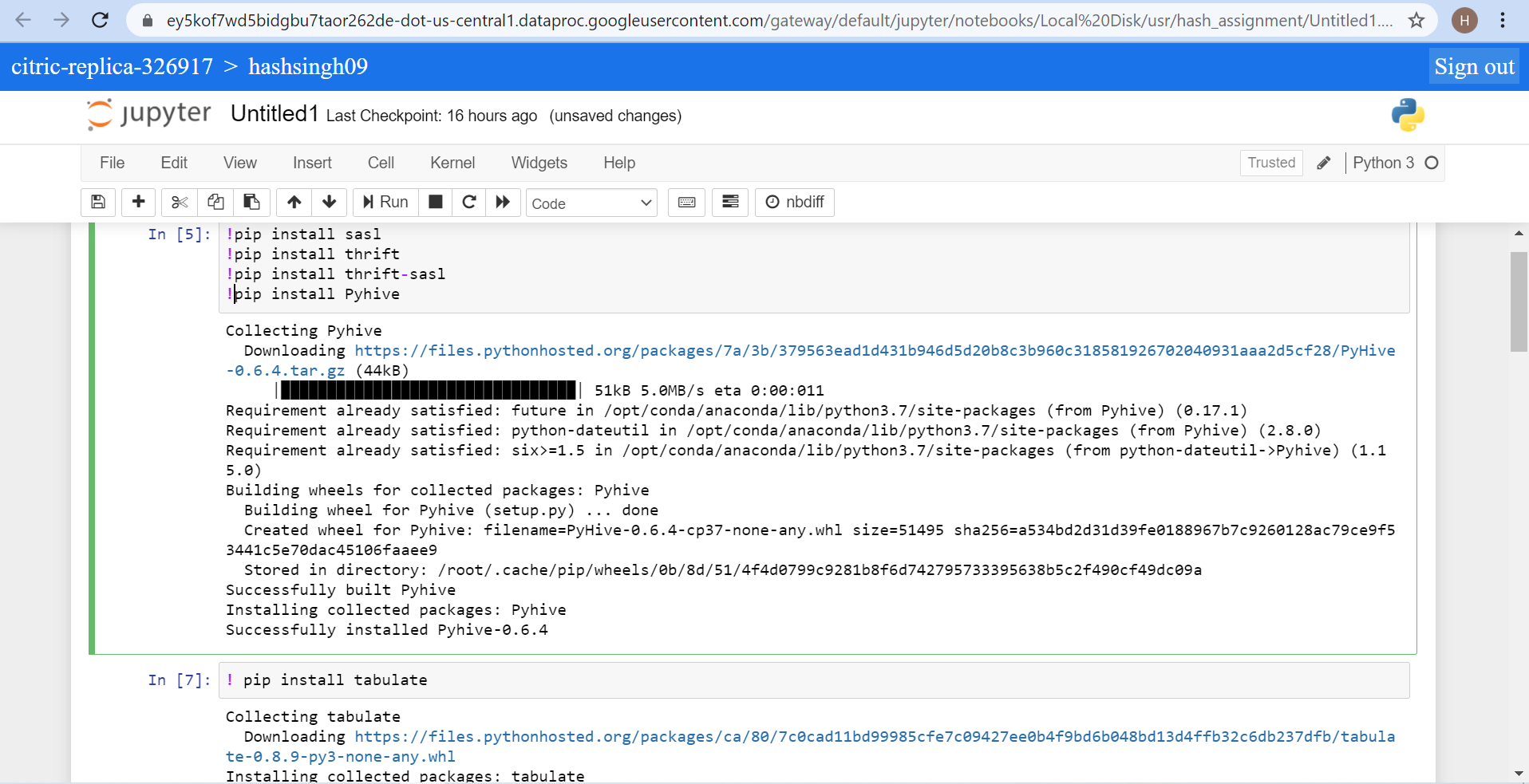


Figure:-12. Installation of necessary library and packages needed to execute the code

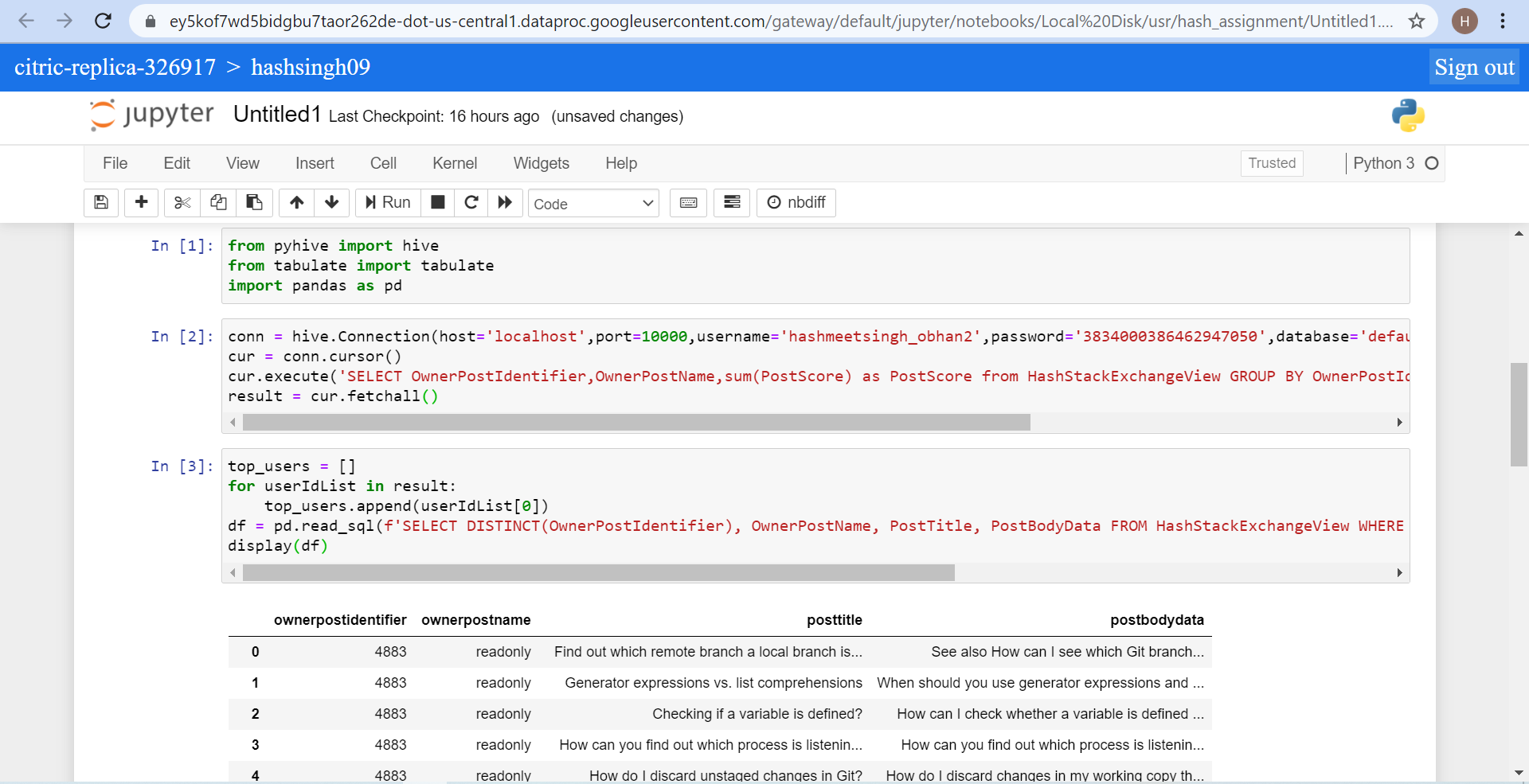


Figure:-13. Established the connection with HIVE Database and found the top 10 users

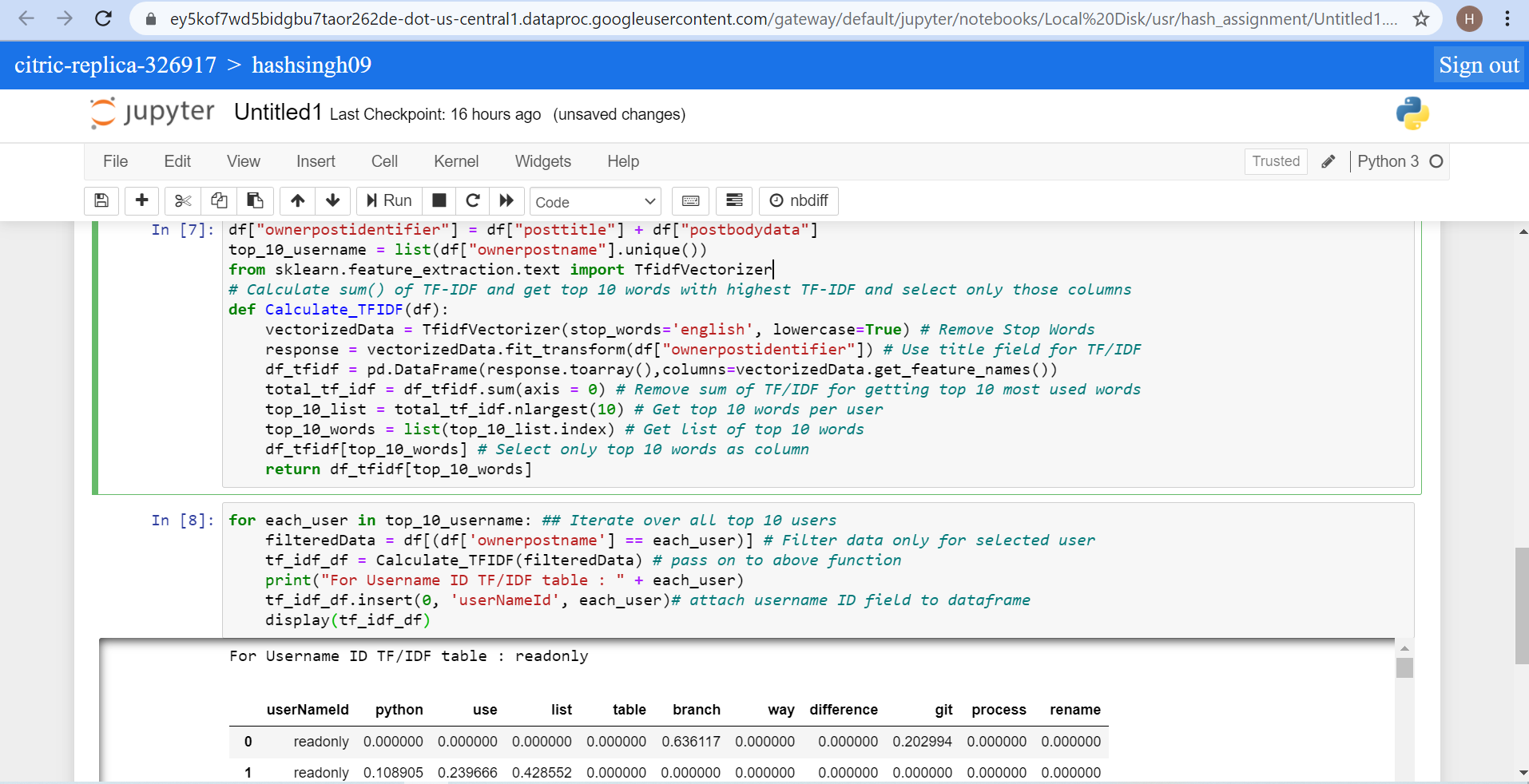
****

Figure:-14. Successfully executed the top 10 terms for each of the 10 users

**APPENDIX**

**References:-**

1. [**https://phoenixnap.com/kb/install-hive-on-ubuntu**](https://phoenixnap.com/kb/install-hive-on-ubuntu)
2. [**https://phoenixnap.com/kb/install-hadoop-ubuntu**](https://phoenixnap.com/kb/install-hadoop-ubuntu)
3. [**https://www.guru99.com/file-permissions.html**](https://www.guru99.com/file-permissions.html)
4. [**https://scikit-learn.org/stable/modules/generated/sklearn.feature\_extraction.text.TfidfVectorizer.html**](https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html)
5. [**https://www.projectpro.io/article/mapreduce-vs-pig-vs-hive/163**](https://www.projectpro.io/article/mapreduce-vs-pig-vs-hive/163)
6. [**https://stackoverflow.com/questions/45999415/removing-html-tags-in-pandas**](https://stackoverflow.com/questions/45999415/removing-html-tags-in-pandas)
7. [**https://www.pythondaddy.com/python/how-to-remove-punctuation-from-a-dataframe-in-pandas-and-python/**](https://www.pythondaddy.com/python/how-to-remove-punctuation-from-a-dataframe-in-pandas-and-python/)
8. [**https://stackoverflow.com/questions/41719259/how-to-remove-numbers-from-string-terms-in-a-pandas-dataframe**](https://stackoverflow.com/questions/41719259/how-to-remove-numbers-from-string-terms-in-a-pandas-dataframe)