CIS5200 Term Project Tutorial

Authors: Bita faraji, Jay Joshi, Shradha Shinde

Instructor: Jongwook Woo

Date: 12/12/2019

Lab Tutorial

Bita Faraji (bfaraji@calstatela.edu)

Jay Joshi (jjoshi6@calstatela.edu)

Shradha Shinde (sshinde6@calstatela.edu)

12/12/2019

New York Times Data Analysis using Hive

OBJECTIVE

The New York Times (NYT) has a large reader base and plays an important role in shaping public opinion and outlook on current affairs and in setting the tone of the public discourse, especially in the U.S. The comments sections for articles in the NYT are quite active and gives insights to reader's opinion on the subject matter of the articles. Each comment can receive other reader's recommendations in the form of upvotes. This project aims at performing data analysis and sheds lights on New York Times Dataset using HIVEQL queries and presenting visualization to see the insights using Power BI, Tableau, & Excel 3-D Maps.

➢ To find out count of document type by type of material for the year 2017 and 2018.
➢ To find out count of type of material with respect to Articles for the year 2017 and 2018.

- ➢ To determine reply count for each document type : Articles/ BlogPosts month wise for year 2017 and 2018.
- ➢ To find out reply count for each comment type for year 2017 and 2018.
- ➢ To find out the degree of polarity to reveal the most positive as well negative headlines for the year 2017 based on public comments.
- ➢ To find out the most active month for the New desk for article and blogpost document type.
- ➢ To find out the highest recommendations based on New desk for the document types: Articles/ BlogPosts.
- ➢ Geo map to show most active locations by the user's reply count for the year 2017 and 2018.
- ➢ To find out the most popular Author for both the years.
- ➢ Also, to show a comparative analysis for the above objectives between the year 2017 and 2018.

INTRODUCTION

This Project aims at performing data analysis and providing insights on New York Times Comments (NYT) using HIVE and presenting the visualization in Tableau and Microsoft Power BI.

In this hands-on lab, you will learn how to:

- ➢ Load data from local desktop(windows) to Linux shell.
- ➢ Download and upload files to HDFS.
- ➢ Extract TXT file using Hive.
- ➢ Data cleaning using Hive.
- ➢ Create Hive tables to query the NYT dataset for analysis.
- ➢ Create Hive queries to analyze the sentiment of data
- ➢ Use Tableau, Power BI, Excel 3D Maps for visualization of the analyzed data.

PLATFORM SPECIFICATIONS

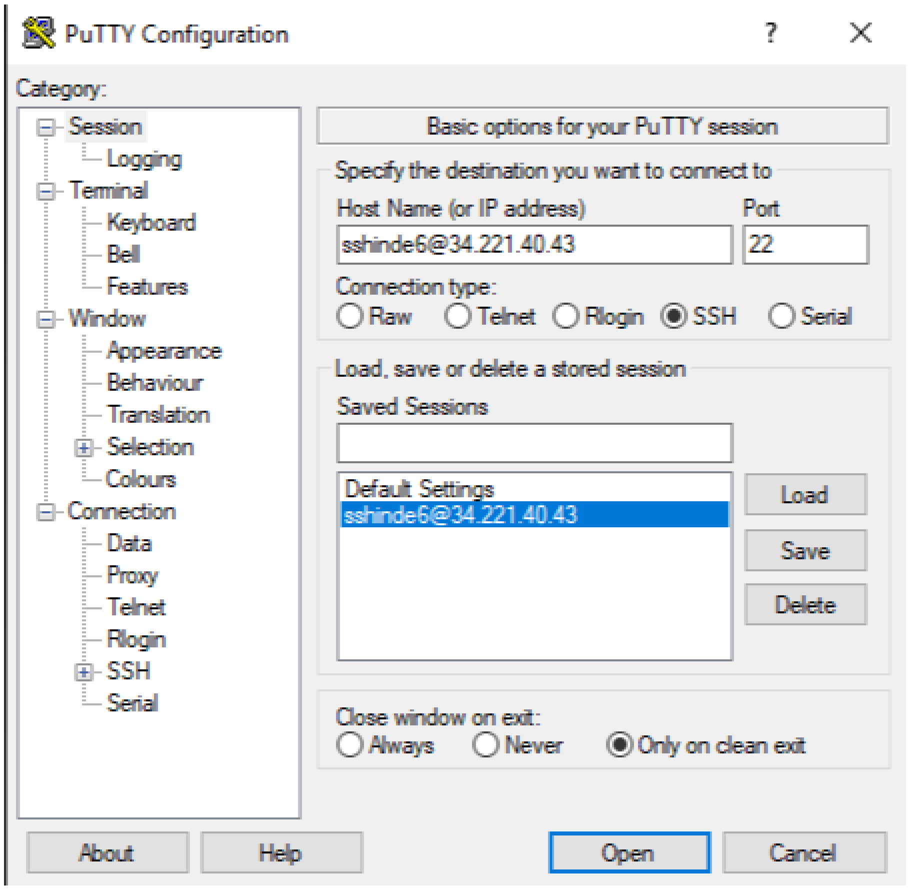
- ➢ Cluster version: Hadoop 2.8.5-amzn-4
- ➢ Cluster number of nodes: 3 nodes
- ➢ Memory size(CPU speed): 2.20 GHz
- ➢ HDFS Capacity: 147 GB
- ➢ Storage: 678 GB
- ➢ Hive Version: Hive 2.3.5-amzn-1

PREREQUISITES

- ➢ You must have Microsoft Excel 2010, 2013 or 2016 installed.
- ➢ You must have your Excel 3D-Map enabled.
- ➢ Tableau 2019.4 installed for visualization of the analyzed data.
- ➢ Power BI Desktop Version
- ➢ AWS EMR :3 nodes

DOWNLOAD THE DATASET

This step is to get data manually. You need to remotely access your AWS EMR cloud that you executed in your AWS Cloud account using ssh using the information - ip address and connect command in beeline CLI-



Specify the destination you want to connect to

Host Name (or IP address): sshinde6@34.221.40.43
Port: 22

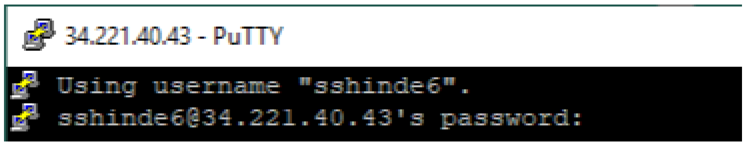
Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions:

Default Settings
sshinde6@34.221.40.43

Close window on exit: Always Never [✓ Only on clean exit]



34.221.40.43 - PuTTY

```
Using username "sshinde6".
sshinde6@34.221.40.43's password:
```

1. ArticleYear2017-

   https://raw.githubusercontent.com/shradhacsula/Data-Project/master/ArticleYear2017.txt

2. ArticleYear2018-

   https://raw.githubusercontent.com/shradhacsula/Data-Project/master/ArticleYear2018.txt

3. CommentYear2017-
   https://www.dropbox.com/preview/CommentYear2017%20.txt?role=personal

4. CommentYear2018-

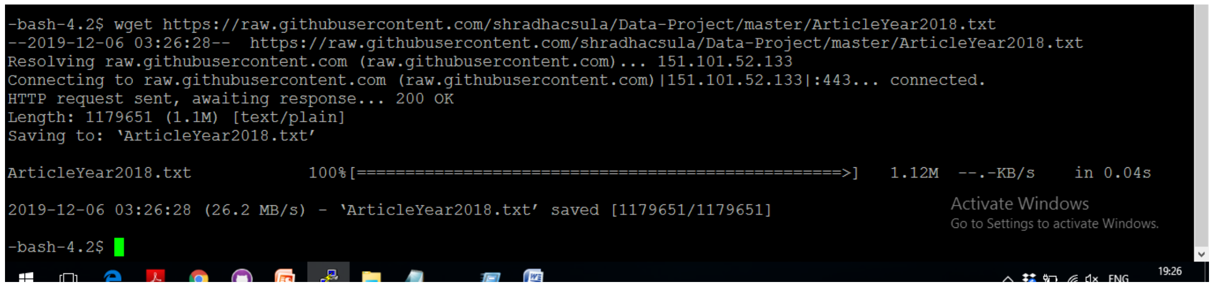
   https://www.dropbox.com/preview/CommentYear2018.txt?role=personal

UPLOAD TXT FILE TO HADOOP DIRECTORY

Before uploading the TXT file to Hadoop directory, we need to first transfer it to local directory using below commands.

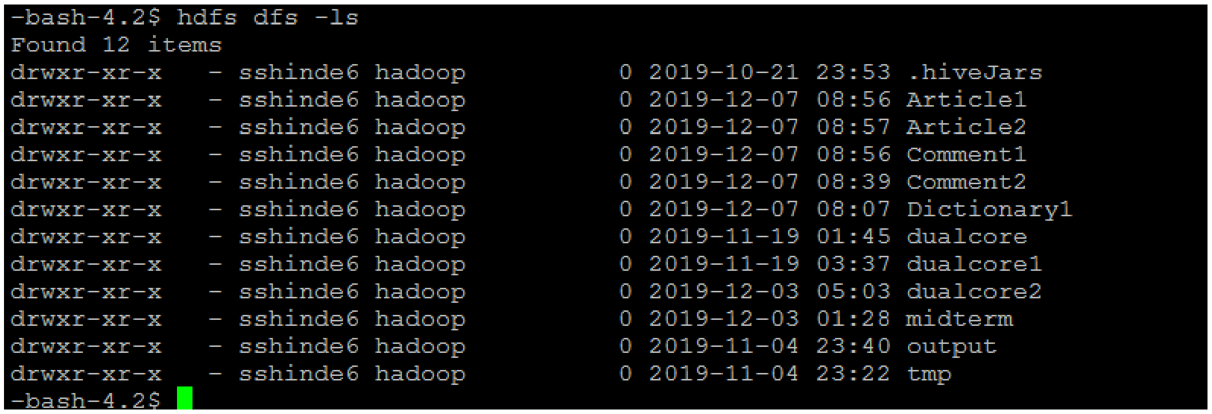
Note: Change the path and username.

```
wget https://raw.githubusercontent.com/shradhacsula/Data-Project/master/ArticleYear2017.txt

wget https://raw.githubusercontent.com/shradhacsula/Data-Project/master/ArticleYear2018.txt
```



```
-bash-4.2$ wget https://raw.githubusercontent.com/shradhacsula/Data-Project/master/ArticleYear2018.txt
--2019-12-06 03:26:28--  https://raw.githubusercontent.com/shradhacsula/Data-Project/master/ArticleYear2018.txt
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.52.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.52.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1179651 (1.1M) [text/plain]
Saving to: 'ArticleYear2018.txt'

ArticleYear2018.txt           100%[=================================================>]   1.12M  --.-KB/s    in 0.04s

2019-12-06 03:26:28 (26.2 MB/s) - 'ArticleYear2018.txt' saved [1179651/1179651]

-bash-4.2$
```

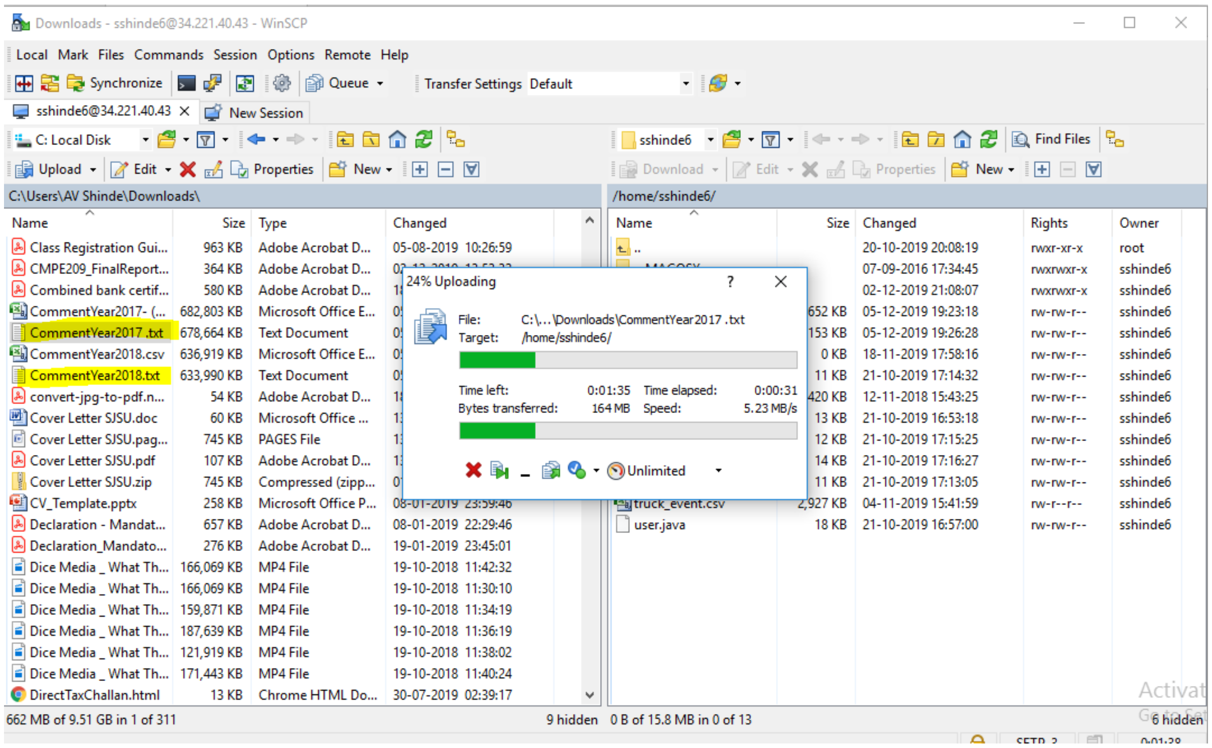
```
$ ls -al
```



```
-bash-4.2$ hdfs dfs -ls
Found 12 items
drwxr-xr-x   - sshinde6 hadoop          0 2019-10-21 23:53 .hiveJars
drwxr-xr-x   - sshinde6 hadoop          0 2019-12-07 08:56 Article1
drwxr-xr-x   - sshinde6 hadoop          0 2019-12-07 08:57 Article2
drwxr-xr-x   - sshinde6 hadoop          0 2019-12-07 08:56 Comment1
drwxr-xr-x   - sshinde6 hadoop          0 2019-12-07 08:39 Comment2
drwxr-xr-x   - sshinde6 hadoop          0 2019-12-07 08:07 Dictionary1
drwxr-xr-x   - sshinde6 hadoop          0 2019-11-19 01:45 dualcore
drwxr-xr-x   - sshinde6 hadoop          0 2019-11-19 03:37 dualcore1
drwxr-xr-x   - sshinde6 hadoop          0 2019-12-03 05:03 dualcore2
drwxr-xr-x   - sshinde6 hadoop          0 2019-12-03 01:28 midterm
drwxr-xr-x   - sshinde6 hadoop          0 2019-11-04 23:40 output
drwxr-xr-x   - sshinde6 hadoop          0 2019-11-04 23:22 tmp
-bash-4.2$
```

Repeat "Step 2" for ArticleYear2018.

Since the CommentYear2017 and CommentYear2018 are more than 25MB we downloaded the datasets to local directory using WinSCP software. We also uploaded the dictionary data set using the same methodology.


Downloads - sshinde6@34.221.40.43 - WinSCP

C:\Users\AV Shinde\Downloads\

Name	Size	Type	Changed
Class Registration Gui...	963 KB	Adobe Acrobat D...	05-08-2019 10:26:59
CMPE209_FinalReport...	364 KB	Adobe Acrobat D...	0
Combined bank certif...	580 KB	Adobe Acrobat D...	1
CommentYear2017- (...	682,803 KB	Microsoft Office E...	0
CommentYear2017 .txt	678,664 KB	Text Document	0
CommentYear2018.csv	636,919 KB	Microsoft Office E...	0
CommentYear2018.txt	633,990 KB	Text Document	0
convert-jpg-to-pdf.n...	54 KB	Adobe Acrobat D...	1
Cover Letter SJSU.doc	60 KB	Microsoft Office ...	1
Cover Letter SJSU.pag...	745 KB	PAGES File	1
Cover Letter SJSU.pdf	107 KB	Adobe Acrobat D...	1
Cover Letter SJSU.zip	745 KB	Compressed (zipp...	0
CV_Template.pptx	258 KB	Microsoft Office P...	08-01-2019 23:59:46
Declaration - Mandat...	657 KB	Adobe Acrobat D...	08-01-2019 22:29:46
Declaration_Mandato...	276 KB	Adobe Acrobat D...	19-01-2019 23:45:01
Dice Media _ What Th...	166,069 KB	MP4 File	19-10-2018 11:42:32
Dice Media _ What Th...	166,069 KB	MP4 File	19-10-2018 11:30:10
Dice Media _ What Th...	159,871 KB	MP4 File	19-10-2018 11:34:19
Dice Media _ What Th...	187,639 KB	MP4 File	19-10-2018 11:36:19
Dice Media _ What Th...	121,919 KB	MP4 File	19-10-2018 11:38:02
Dice Media _ What Th...	171,443 KB	MP4 File	19-10-2018 11:40:24
DirectTaxChallan.html	13 KB	Chrome HTML Do...	30-07-2019 02:39:17

662 MB of 9.51 GB in 1 of 311 9 hidden

/home/sshinde6/

Name	Size	Changed	Rights	Owner
..		20-10-2019 20:08:19	rwxr-xr-x	root
		07-09-2016 17:34:45	rwxrwxr-x	sshinde6
		02-12-2019 21:08:07	rwxrwxr-x	sshinde6
	652 KB	05-12-2019 19:23:18	rw-rw-r--	sshinde6
	153 KB	05-12-2019 19:26:28	rw-rw-r--	sshinde6
	0 KB	18-11-2019 17:58:16	rw-rw-r--	sshinde6
	11 KB	21-10-2019 17:14:32	rw-rw-r--	sshinde6
	420 KB	12-11-2018 15:43:25	rw-rw-r--	sshinde6
	13 KB	21-10-2019 16:53:18	rw-rw-r--	sshinde6
	12 KB	21-10-2019 17:15:25	rw-rw-r--	sshinde6
	14 KB	21-10-2019 17:16:27	rw-rw-r--	sshinde6
	11 KB	21-10-2019 17:13:05	rw-rw-r--	sshinde6
truck_event.csv	2,927 KB	04-11-2019 15:41:59	rw-r--r--	sshinde6
user.java	18 KB	21-10-2019 16:57:00	rw-rw-r--	sshinde6

0 B of 15.8 MB in 0 of 13 6 hidden

24% Uploading

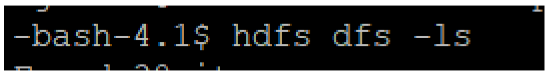
File: C:\...\Downloads\CommentYear2017 .txt
Target: /home/sshinde6/

Time left: 0:01:35 Time elapsed: 0:00:31
Bytes transferred: 164 MB Speed: 5.23 MB/s

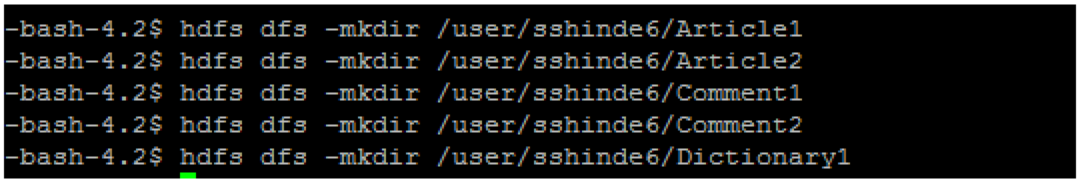
Now we have to upload all the TXT files to HDFS folder. Run the following HDFS commands to create and list the article1, article , comment1, comment2, dictionary1 directories in HDFS.

```
Hdfs dfs -mkdir /user/sshinde6/Article1

Hdfs dfs -mkdir /user/sshinde6/ Article2

Hdfs dfs -mkdir /user/sshinde6/Comment1

Hdfs dfs -mkdir /user/sshinde6/ Comment2

Hdfs dfs -mkdir /user/sshinde6/Dictionary1

hdfs dfs -put CommentYear2017.txt /user/sshinde6/Comment1/

hdfs dfs -put CommentYear2018.txt /user/sshinde6/Comment2/

hdfs dfs -put ArticleYear2017.txt /user/sshinde6/Article1/

hdfs dfs -put ArticleYear2018.txt /user/sshinde6/Article2/

hdfs dfs -put dictionary.txt /user/sshinde6/Dictionary1/
```

st



```
-bash-4.1$ hdfs dfs -ls
```



```
-bash-4.2$ hdfs dfs -mkdir /user/sshinde6/Article1
-bash-4.2$ hdfs dfs -mkdir /user/sshinde6/Article2
-bash-4.2$ hdfs dfs -mkdir /user/sshinde6/Comment1
-bash-4.2$ hdfs dfs -mkdir /user/sshinde6/Comment2
-bash-4.2$ hdfs dfs -mkdir /user/sshinde6/Dictionary1
```

Give permissions

Run the following HDFS command to make your beeline command works:

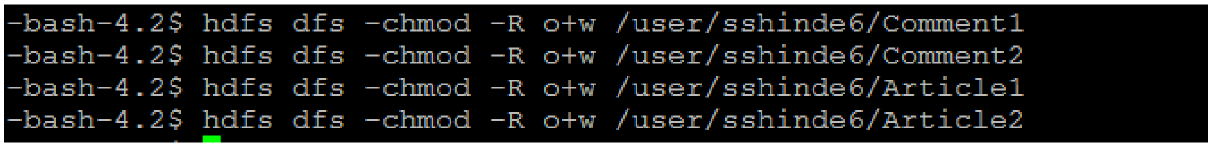
-bash-4.1$ hdfs dfs -chmod -R o+w /user/sshinde6/Comment1/

-bash-4.1$ hdfs dfs -chmod -R o+w /user/sshinde6/Comment2/

-bash-4.1$ hdfs dfs -chmod -R o+w /user/sshinde6/Article1/

-bash-4.1$ hdfs dfs -chmod -R o+w /user/sshinde6/Article2/

-bash-4.1$ hdfs dfs -chmod -R o+w /user/sshinde6/Dictionary1/



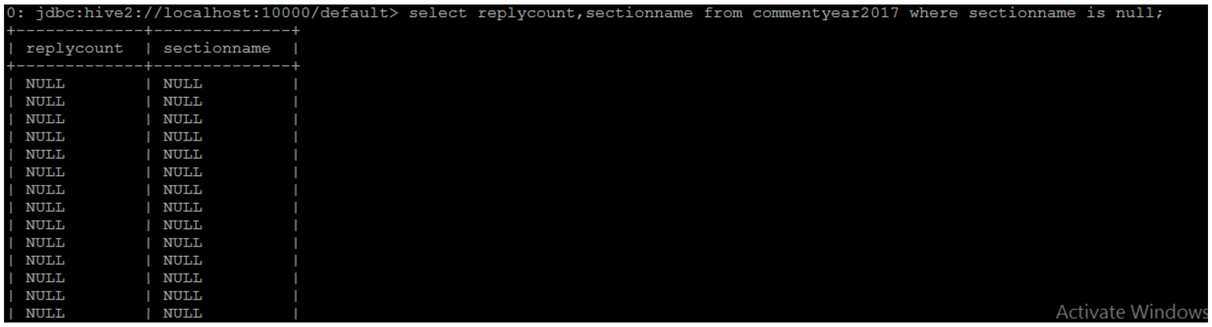
```
-bash-4.2$ hdfs dfs -chmod -R o+w /user/sshinde6/Comment1
-bash-4.2$ hdfs dfs -chmod -R o+w /user/sshinde6/Comment2
-bash-4.2$ hdfs dfs -chmod -R o+w /user/sshinde6/Article1
-bash-4.2$ hdfs dfs -chmod -R o+w /user/sshinde6/Article2
```

<u>DATA CLEANING</u>

Removing Null Values

Null values were removed from tables. For example, section name and replycount columns had null values as shown below:

Before:

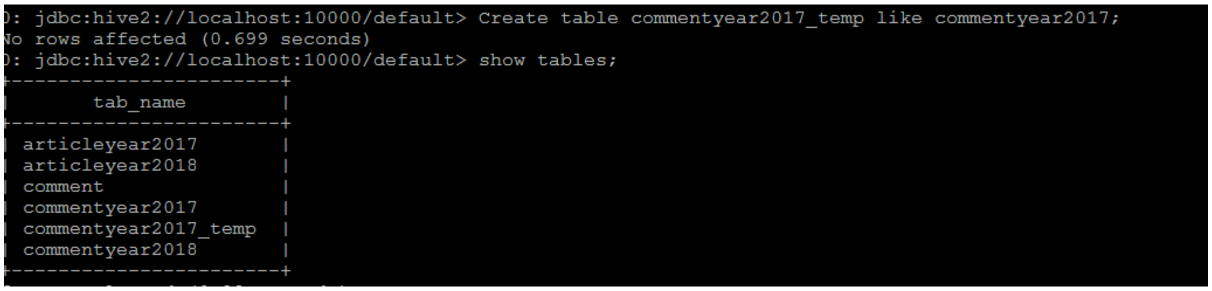


```
0: jdbc:hive2://localhost:10000/default> select replycount,sectionname from commentyear2017 where sectionname is null;
+-------------+--------------+
| replycount  | sectionname  |
+-------------+--------------+
| NULL        | NULL         |
| NULL        | NULL         |
| NULL        | NULL         |
| NULL        | NULL         |
| NULL        | NULL         |
| NULL        | NULL         |
| NULL        | NULL         |
| NULL        | NULL         |
| NULL        | NULL         |
| NULL        | NULL         |
| NULL        | NULL         |
| NULL        | NULL         |
| NULL        | NULL         |
| NULL        | NULL         |
```

Below steps were performed to remove null values

Step 1:

```
#External Table was created
create table commentyear2017_temp like commentyear2017;
show tables;
```



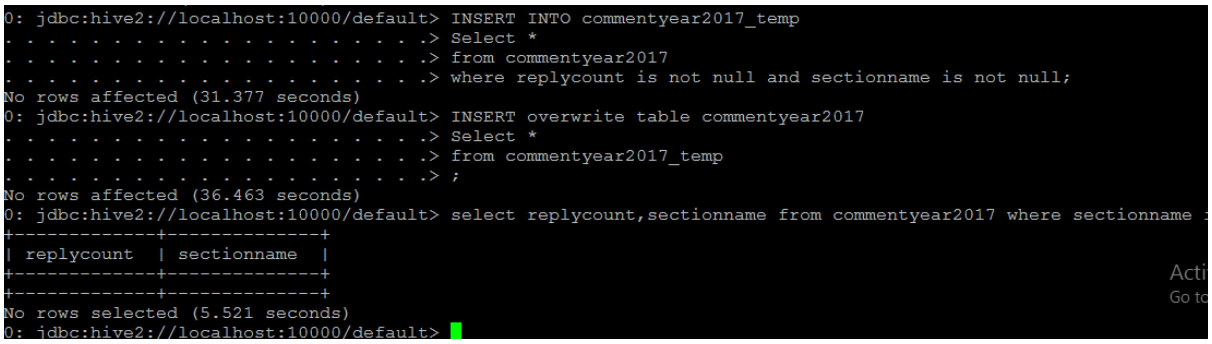
```
0: jdbc:hive2://localhost:10000/default> Create table commentyear2017_temp like commentyear2017;
No rows affected (0.699 seconds)
0: jdbc:hive2://localhost:10000/default> show tables;
+-----------------------+
|       tab_name        |
+-----------------------+
| articleyear2017       |
| articleyear2018       |
| comment               |
| commentyear2017       |
| commentyear2017_temp  |
| commentyear2018       |
+-----------------------+
```

Step 2:

```
#Inserted data from original table

INSERT INTO commentyear2017_temp
Select *
From commentyear2017
Where replycount is not null and sectionname is not null;
```

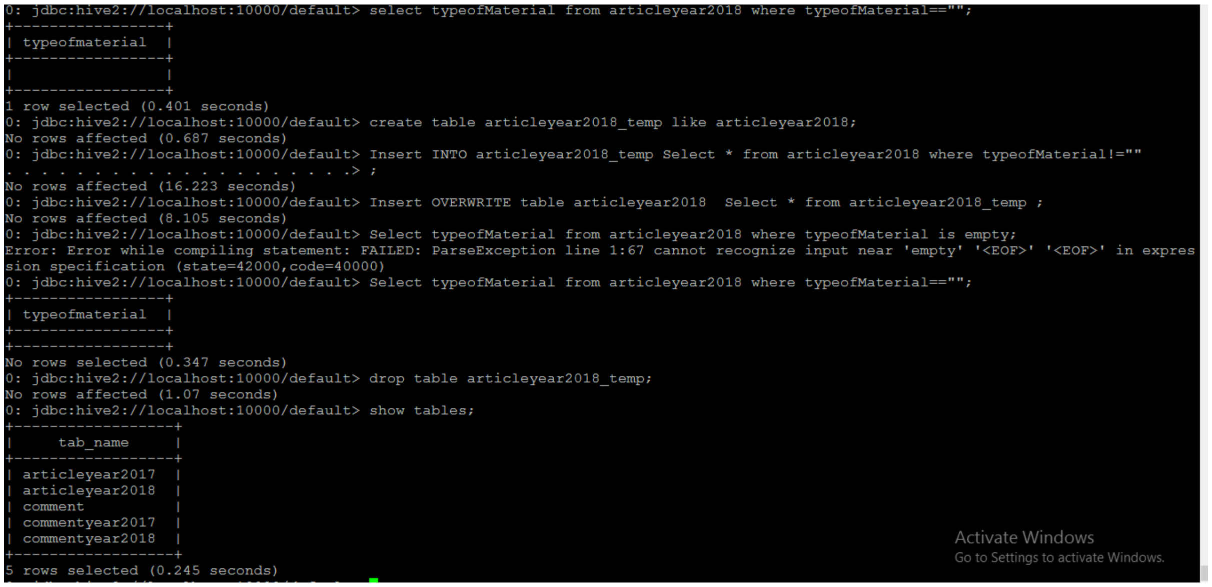
All null values from columns were removed. Rechecked using below query:

```
Inserted overwrite table commentyear2017
Select *
From commentyear2017_temp;
Select replycount, sectionname from commentyear2017 where sectionname is null;
```



```
0: jdbc:hive2://localhost:10000/default> INSERT INTO commentyear2017_temp
. . . . . . . . . . . . . . . . . . . . .> Select *
. . . . . . . . . . . . . . . . . . . . .> from commentyear2017
. . . . . . . . . . . . . . . . . . . . .> where replycount is not null and sectionname is not null;
No rows affected (31.377 seconds)
0: jdbc:hive2://localhost:10000/default> INSERT overwrite table commentyear2017
. . . . . . . . . . . . . . . . . . . . .> Select *
. . . . . . . . . . . . . . . . . . . . .> from commentyear2017_temp
. . . . . . . . . . . . . . . . . . . . .> ;
No rows affected (36.463 seconds)
0: jdbc:hive2://localhost:10000/default> select replycount,sectionname from commentyear2017 where sectionname
+-------------+--------------+
| replycount  | sectionname  |
+-------------+--------------+
+-------------+--------------+
No rows selected (5.521 seconds)
0: jdbc:hive2://localhost:10000/default>
```

Repeated same process for removing empty values from articleyear2018 table



```
0: jdbc:hive2://localhost:10000/default> select typeofMaterial from articleyear2018 where typeofMaterial=="";
+-----------------+
| typeofmaterial  |
+-----------------+
|                 |
+-----------------+
1 row selected (0.401 seconds)
0: jdbc:hive2://localhost:10000/default> create table articleyear2018_temp like articleyear2018;
No rows affected (0.687 seconds)
0: jdbc:hive2://localhost:10000/default> Insert INTO articleyear2018_temp Select * from articleyear2018 where typeofMaterial!=""
. . . . . . . . . . . . . . . . . . . . .> ;
No rows affected (16.223 seconds)
0: jdbc:hive2://localhost:10000/default> Insert OVERWRITE table articleyear2018  Select * from articleyear2018_temp ;
No rows affected (8.105 seconds)
0: jdbc:hive2://localhost:10000/default> Select typeofMaterial from articleyear2018 where typeofMaterial is empty;
Error: Error while compiling statement: FAILED: ParseException line 1:67 cannot recognize input near 'empty' '<EOF>' '<EOF>' in expres
sion specification (state=42000,code=40000)
0: jdbc:hive2://localhost:10000/default> Select typeofMaterial from articleyear2018 where typeofMaterial=="";
+-----------------+
| typeofmaterial  |
+-----------------+
+-----------------+
No rows selected (0.347 seconds)
0: jdbc:hive2://localhost:10000/default> drop table articleyear2018_temp;
No rows affected (1.07 seconds)
0: jdbc:hive2://localhost:10000/default> show tables;
+------------------+
|     tab_name     |
+------------------+
| articleyear2017  |
| articleyear2018  |
| comment          |
| commentyear2017  |
| commentyear2018  |
+------------------+
5 rows selected (0.245 seconds)
```

CREATE HIVE TABLE TO QUERY NEW YORK TIMES DATA

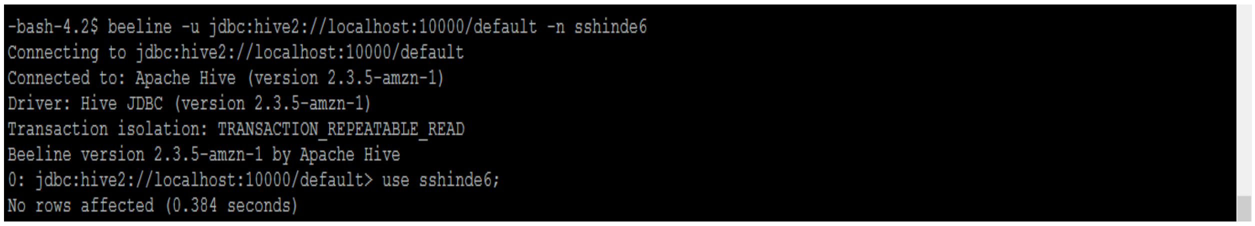
Open beeline CLI (Command Line Shell Interface) that is equivalent to hive CLI environment as follows, which you have done in the previous lab.

```
beeline
```

Beeline is for multiple user's access to Hive Server 2 of a Hadoop cluster.

Use the below command to connect to beeline:

beeline -u jdbc:hive2://localhost:10000/default -n sshinde6



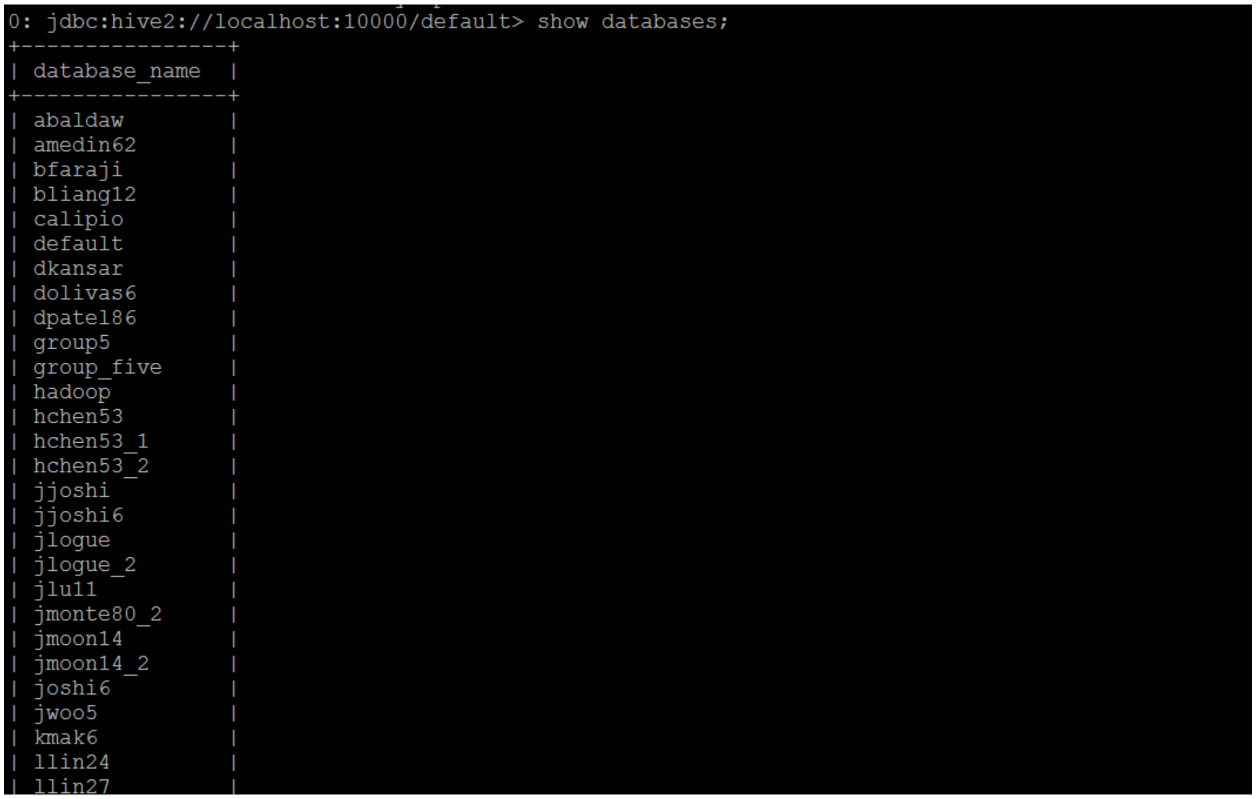
```
-bash-4.2$ beeline -u jdbc:hive2://localhost:10000/default -n sshinde6
Connecting to jdbc:hive2://localhost:10000/default
Connected to: Apache Hive (version 2.3.5-amzn-1)
Driver: Hive JDBC (version 2.3.5-amzn-1)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 2.3.5-amzn-1 by Apache Hive
0: jdbc:hive2://localhost:10000/default> use sshinde6;
No rows affected (0.384 seconds)
```

NOTE: Now we have to create your own database with your username to separate your tables with other users you have to use your username. For example, the user should run the following command.

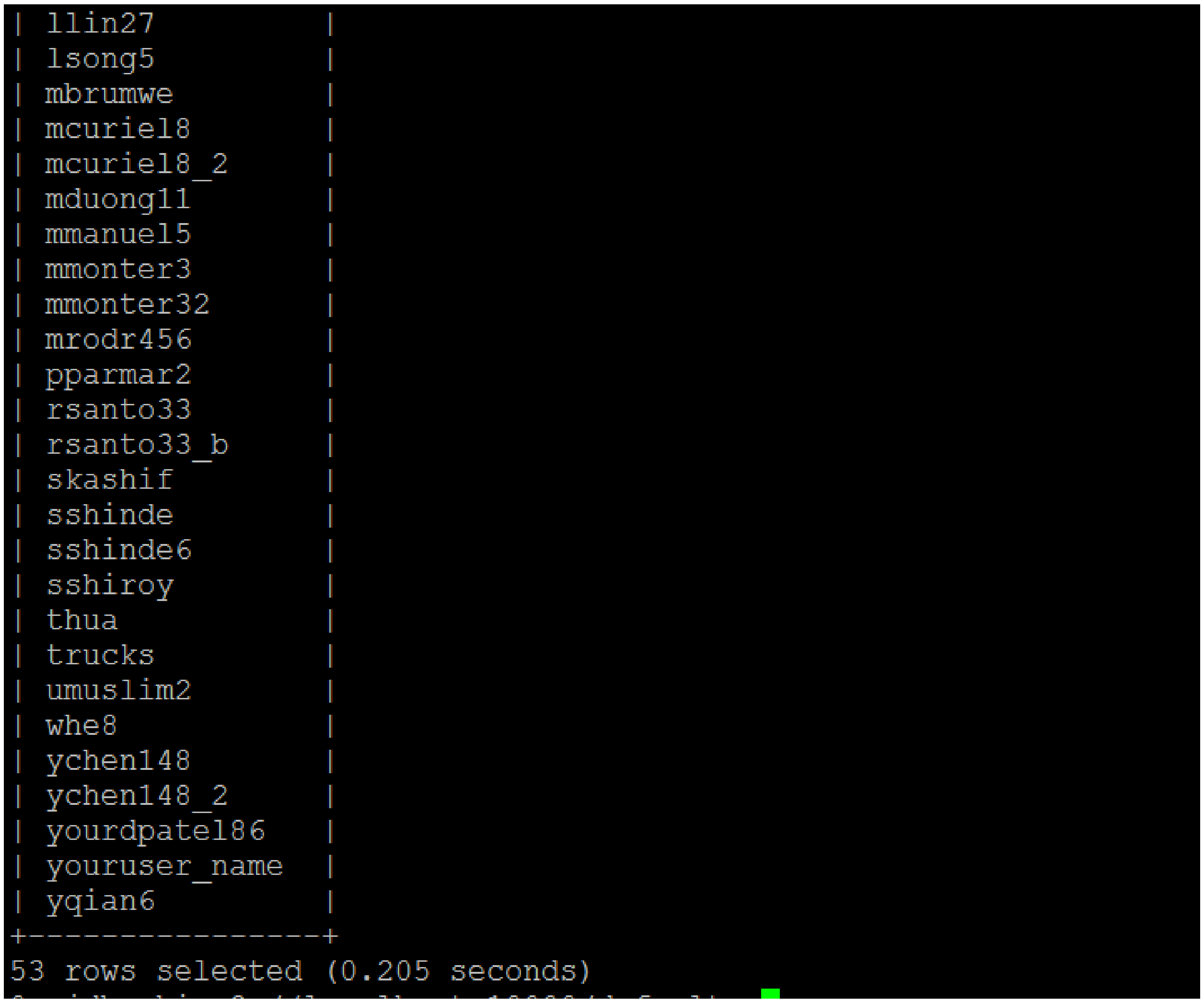
0: jdbc:hive2://localhost:1000/default >CREATE DATABASE sshinde;

No rows affected (0.277 seconds)

0: jdbc:hive2://localhost:1000/default > show databases;



```
0: jdbc:hive2://localhost:10000/default> show databases;
+----------------+
| database_name  |
+----------------+
| abaldaw        |
| amedin62       |
| bfaraji        |
| bliang12       |
| calipio        |
| default        |
| dkansar        |
| dolivas6       |
| dpatel86       |
| group5         |
| group_five     |
| hadoop         |
| hchen53        |
| hchen53_1      |
| hchen53_2      |
| jjoshi         |
| jjoshi6        |
| jlogue         |
| jlogue_2       |
| jlu11          |
| jmonte80_2     |
| jmoon14        |
| jmoon14_2      |
| joshi6         |
| jwoo5          |
| kmak6          |
| llin24         |
| llin27         |
```



```
| llin27          |
| lsong5          |
| mbrumwe         |
| mcuriel8        |
| mcuriel8_2      |
| mduong11        |
| mmanuel5        |
| mmonter3        |
| mmonter32       |
| mrodr456        |
| pparmar2        |
| rsanto33        |
| rsanto33_b      |
| skashif         |
| sshinde         |
| sshinde6        |
| sshiroy         |
| thua            |
| trucks          |
| umuslim2        |
| whe8            |
| ychen148        |
| ychen148_2      |
| yourdpatel86    |
| youruser_name   |
| yqian6          |
+-----------------+
53 rows selected (0.205 seconds)
```

The following hive statement creates an external table for ArticleYear2017, ArticleYear2018. External tables preserve the data in the original file format, while allowing Hive to perform queries against the data within the file.

In the hive shell CLI, you need to copy and paste the following HiveQL code to create an external table CommentYear2017.

```
create external table if not exists CommentYear2017(Month_Name STRING,approveDate
STRING,articleID STRING,articleWordCount BIGINT,commentBody STRING,commentID
STRING,commentSequence STRING,commentTitle STRING,commentType STRING,createDate
STRING,depth INT,editorsSelection INT,inReplyTo STRING,newDesk STRING,parentID
STRING,parentUserDisplayName STRING, permID STRING, picURL STRING, printPage INT,
recommendations INT, recommendedFlag INT, replyCount INT, reportAbuseFlag INT, sectionName
STRING, sharing INT, status STRING, timespeople INT, trusted INT, updateDate STRING,
userDisplayName STRING, userID STRING, userLocation STRING, userTitle STRING, userURL
STRING,typeofmaterial STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE location "/user/sshinde6/Comment1/"
TBLPROPERTIES ('skip.header.line.count'='1');
```

In the hive shell CLI, you need to copy and paste the following HiveQL code to create an external table CommentYear2018.

```
create external table if not exists CommentYear2018(Month_Name STRING,approveDate
STRING,articleID STRING,articleWordCount BIGINT,commentBody STRING,commentID
STRING,commentSequence STRING,commentTitle STRING,commentType STRING,createDate
STRING,depth INT,editorsSelection INT,inReplyTo STRING,newDesk STRING,parentID
STRING,parentUserDisplayName STRING, permID STRING, picURL STRING, printPage INT,
recommendations INT, recommendedFlag INT, replyCount INT, reportAbuseFlag INT, sectionName
STRING, sharing INT, status STRING, timespeople INT, trusted INT, updateDate STRING,
userDisplayName STRING, userID STRING, userLocation STRING, userTitle STRING, userURL
STRING,typeofmaterial STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE location "/user/sshinde6/Comment2/"
TBLPROPERTIES ('skip.header.line.count'='1');
```

In the hive shell CLI, you need to copy and paste the following HiveQL code to create an external table ArticleYear2017.

```
create external table if not exists articleyear2017(Month_Name STRING,articleID STRING,abstract
STRING,byline STRING,documentType STRING,headline STRING,keywords STRING,multimedia INT,
newDesk STRING,printPage INT,pubDate TIMESTAMP,source STRING,
typeOfMaterial STRING,
webURL STRING,
articleWordCount BIGINT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE location "/user/sshinde6/Article1/"
TBLPROPERTIES ('skip.header.line.count'='1');
```

In the hive shell CLI, you need to copy and paste the following HiveQL code to create an external table ArticleYear2018**.**

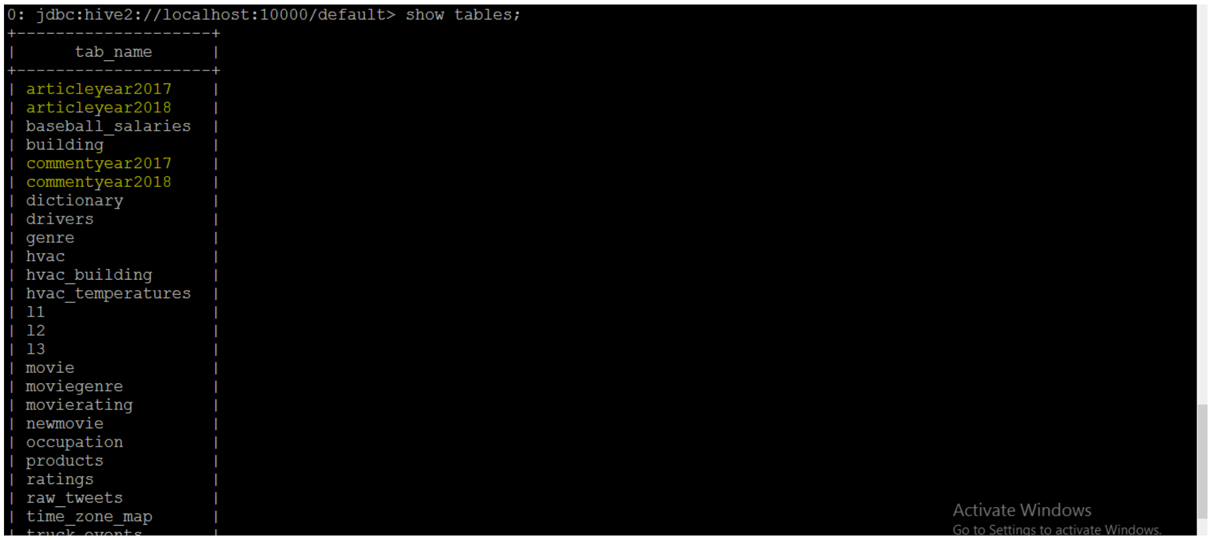
```
create external table if not exists articleyear2018(Month_Name STRING,articleID STRING,abstract
STRING,byline STRING,documentType STRING,headline STRING,keywords STRING,multimedia INT,
newDesk STRING,printPage INT,pubDate TIMESTAMP,source STRING,
typeOfMaterial STRING,
webURL STRING,
articleWordCount BIGINT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE location "/user/sshinde6/Article2/"
TBLPROPERTIES ('skip.header.line.count'='1');
```

In the hive shell CLI, you need to copy and paste the following HiveQL code to create an external table dictionary.

```
CREATE EXTERNAL TABLE if not exists dictionary (type string,length int,word string,pos string,
stemmed string,
polarity string )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
LOCATION "/user/sshinde6/Dictionary1/"
```

Now you may see if those tables are created with “show tables”:

0: jdbc:hive2://localhost:1000/default > show tables;



```
0: jdbc:hive2://localhost:10000/default> show tables;
+--------------------+
|      tab_name      |
+--------------------+
| articleyear2017    |
| articleyear2018    |
| baseball_salaries  |
| building           |
| commentyear2017    |
| commentyear2018    |
| dictionary         |
| drivers            |
| genre              |
| hvac               |
| hvac_building      |
| hvac_temperatures  |
| l1                 |
| l2                 |
| l3                 |
| movie              |
| moviegenre         |
| movierating        |
| newmovie           |
| occupation         |
| products           |
| ratings            |
| raw_tweets         |
| time_zone_map      |
```

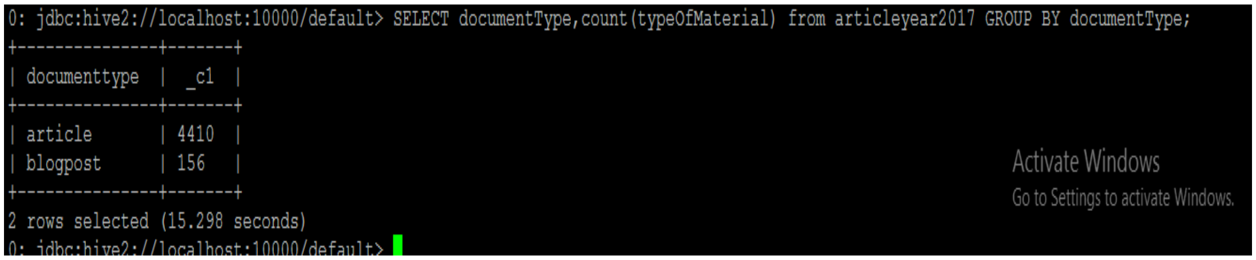
QUERYING ON THE DATASET

Query 1: Show the count of document type by type of material for the year 2017 and 2018?

In this query, we have tried to determine what number of articles and blogpost are present in NYT for both the years respectively.

For year 2017:

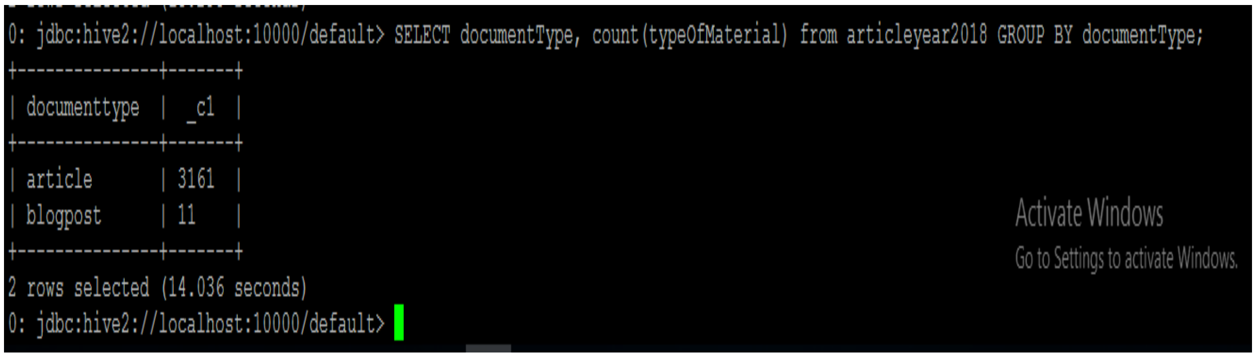
```
SELECT documentType,count(typeOfMaterial) from articleyear2017 GROUP BY documentType;
```



```
0: jdbc:hive2://localhost:10000/default> SELECT documentType,count(typeOfMaterial) from articleyear2017 GROUP BY documentType;
+--------------+-------+
| documenttype |  _c1  |
+--------------+-------+
| article      | 4410  |
| blogpost     | 156   |
+--------------+-------+
2 rows selected (15.298 seconds)
0: jdbc:hive2://localhost:10000/default>
```

For year 2018:

```
SELECT documentType, count(typeOfMaterial) from articleyear2018 GROUP BY documentType;
```

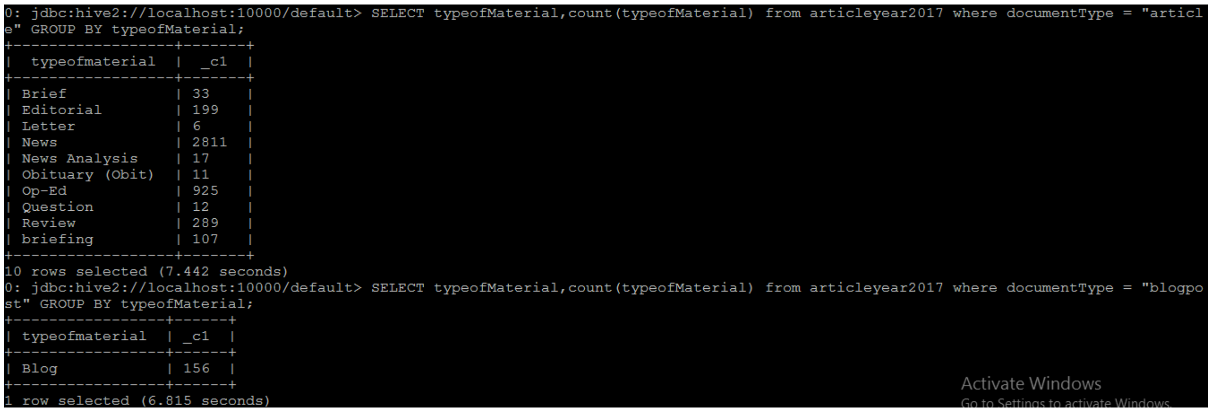


```
0: jdbc:hive2://localhost:10000/default> SELECT documentType, count(typeOfMaterial) from articleyear2018 GROUP BY documentType;
+--------------+-------+
| documenttype |  _c1  |
+--------------+-------+
| article      | 3161  |
| blogpost     | 11    |
+--------------+-------+
2 rows selected (14.036 seconds)
0: jdbc:hive2://localhost:10000/default>
```

Show the count of type of material with respect to Articles for the year 2017 and 2018?

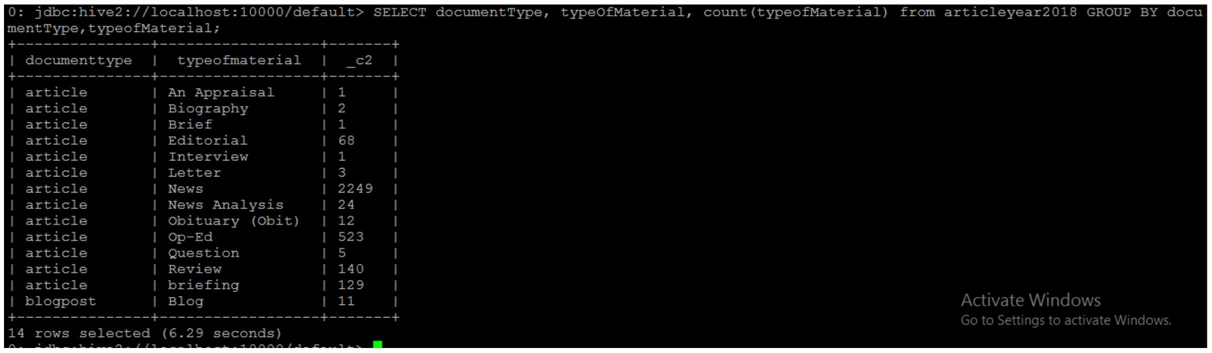
In this query, we have tried to determine what number of type of materials for article document type which are present in NYT for both the years respectively.

```
SELECT documentType, typeOfMaterial, count(typeOfMaterial) from articleyear2017 GROUP BY documentType, typeOfMaterial;
```



```
0: jdbc:hive2://localhost:10000/default> SELECT typeofMaterial,count(typeofMaterial) from articleyear2017 where documentType = "articl
e" GROUP BY typeofMaterial;
+------------------+-------+
|  typeofmaterial  |  _c1  |
+------------------+-------+
| Brief            | 33    |
| Editorial        | 199   |
| Letter           | 6     |
| News             | 2811  |
| News Analysis    | 17    |
| Obituary (Obit)  | 11    |
| Op-Ed            | 925   |
| Question         | 12    |
| Review           | 289   |
| briefing         | 107   |
+------------------+-------+
10 rows selected (7.442 seconds)
0: jdbc:hive2://localhost:10000/default> SELECT typeofMaterial,count(typeofMaterial) from articleyear2017 where documentType = "blogpo
st" GROUP BY typeofMaterial;
+-----------------+------+
| typeofmaterial  | _c1  |
+-----------------+------+
| Blog            | 156  |
+-----------------+------+
1 row selected (6.815 seconds)
```

```
SELECT documentType, typeOfMaterial, count(typeOfMaterial) from articleyear2018 GROUP BY documentType, typeOfMaterial;
```



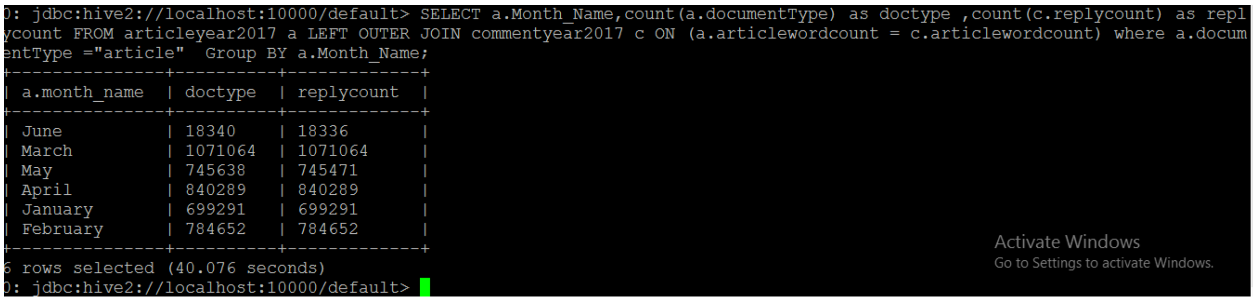
```
0: jdbc:hive2://localhost:10000/default> SELECT documentType, typeOfMaterial, count(typeofMaterial) from articleyear2018 GROUP BY docu
mentType,typeofMaterial;
+---------------+------------------+-------+
| documenttype  |  typeofmaterial  |  _c2  |
+---------------+------------------+-------+
| article       | An Appraisal     | 1     |
| article       | Biography        | 2     |
| article       | Brief            | 1     |
| article       | Editorial        | 68    |
| article       | Interview        | 1     |
| article       | Letter           | 3     |
| article       | News             | 2249  |
| article       | News Analysis    | 24    |
| article       | Obituary (Obit)  | 12    |
| article       | Op-Ed            | 523   |
| article       | Question         | 5     |
| article       | Review           | 140   |
| article       | briefing         | 129   |
| blogpost      | Blog             | 11    |
+---------------+------------------+-------+
14 rows selected (6.29 seconds)
```

Query 2: What is the reply count for the document type month wise for year 2017 & 2018?

Below query shows the reply count for each document type for month January to June for year 2017 & 2018. Month name is from articleyear2017 and reply count is from commentyear2017 table. Left outer join is used to get the desired output and is Grouped by month for each of the document type: article and blogpost.

For Year 2017:

```
SELECT a.Month_Name,count(a.documentType) as doctype ,count(c.replycount) as replycount
FROM articleyear2017 a
LEFT OUTER JOIN commentyear2017 c
ON (a.articlewordcount = c.articlewordcount)
where a.documentType ="article"
Group BY a.Month_Name;
```

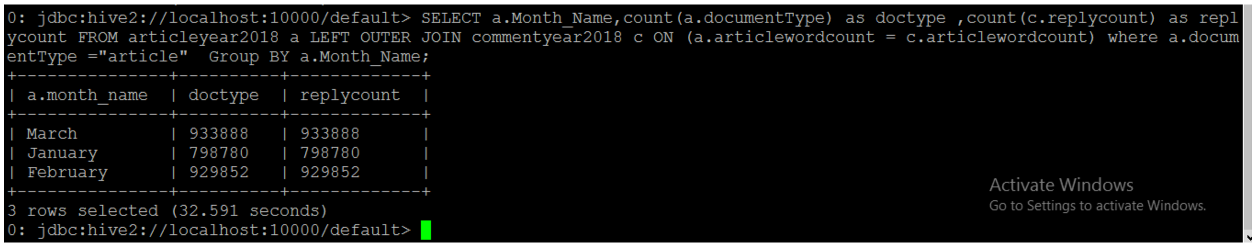


```
0: jdbc:hive2://localhost:10000/default> SELECT a.Month_Name,count(a.documentType) as doctype ,count(c.replycount) as repl
ycount FROM articleyear2017 a LEFT OUTER JOIN commentyear2017 c ON (a.articlewordcount = c.articlewordcount) where a.docum
entType ="article"  Group BY a.Month_Name;
+---------------+----------+-------------+
| a.month_name  | doctype  | replycount  |
+---------------+----------+-------------+
| June          | 18340    | 18336       |
| March         | 1071064  | 1071064     |
| May           | 745638   | 745471      |
| April         | 840289   | 840289      |
| January       | 699291   | 699291      |
| February      | 784652   | 784652      |
+---------------+----------+-------------+
6 rows selected (40.076 seconds)
0: jdbc:hive2://localhost:10000/default>
```

Activate Windows
Go to Settings to activate Windows.

For Year 2018:

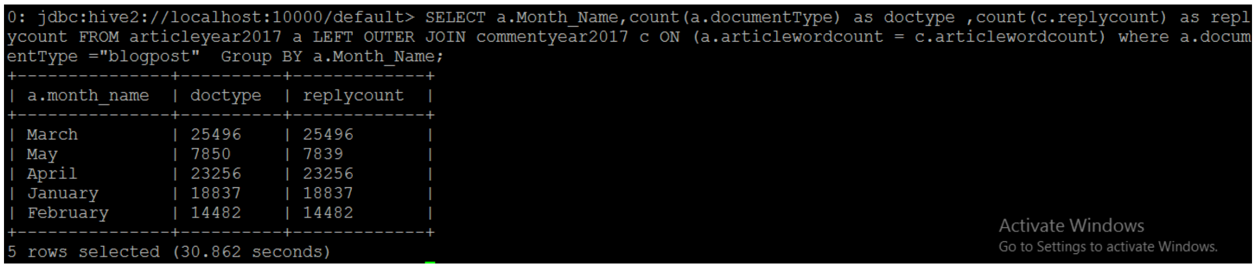
```
SELECT a.Month_Name,count(a.documentType) as doctype ,count(c.replycount) as replycount
FROM articleyear2018 a
LEFT OUTER JOIN commentyear2018 c
ON (a.articlewordcount = c.articlewordcount)
where a.documentType ="article"
Group BY a.Month_Name;
```



```
0: jdbc:hive2://localhost:10000/default> SELECT a.Month_Name,count(a.documentType) as doctype ,count(c.replycount) as repl
ycount FROM articleyear2018 a LEFT OUTER JOIN commentyear2018 c ON (a.articlewordcount = c.articlewordcount) where a.docum
entType ="article"  Group BY a.Month_Name;
+---------------+----------+-------------+
| a.month_name  | doctype  | replycount  |
+---------------+----------+-------------+
| March         | 933888   | 933888      |
| January       | 798780   | 798780      |
| February      | 929852   | 929852      |
+---------------+----------+-------------+
3 rows selected (32.591 seconds)
0: jdbc:hive2://localhost:10000/default>
```

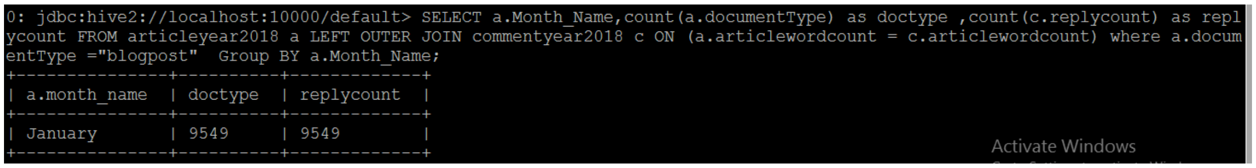
Activate Windows
Go to Settings to activate Windows.

```
SELECT a.Month_Name,count(a.documentType) as doctype ,count(c.replycount) as replycount
FROM articleyear2017 a
LEFT OUTER JOIN commentyear2017 c
ON (a.articlewordcount = c.articlewordcount)
where a.documentType ="blogpost"
Group BY a.Month_Name;
```



```
0: jdbc:hive2://localhost:10000/default> SELECT a.Month_Name,count(a.documentType) as doctype ,count(c.replycount) as repl
ycount FROM articleyear2017 a LEFT OUTER JOIN commentyear2017 c ON (a.articlewordcount = c.articlewordcount) where a.docum
entType ="blogpost"  Group BY a.Month_Name;
+---------------+----------+-------------+
| a.month_name  | doctype  | replycount  |
+---------------+----------+-------------+
| March         | 25496    | 25496       |
| May           | 7850     | 7839        |
| April         | 23256    | 23256       |
| January       | 18837    | 18837       |
| February      | 14482    | 14482       |
+---------------+----------+-------------+
5 rows selected (30.862 seconds)
```

```
SELECT a.Month_Name,count(a.documentType) as doctype ,count(c.replycount) as replycount
FROM articleyear2018 a
LEFT OUTER JOIN commentyear2018 c
ON (a.articlewordcount = c.articlewordcount)
where a.documentType ="blogpost"
Group BY a.Month_Name;
```



```
0: jdbc:hive2://localhost:10000/default> SELECT a.Month_Name,count(a.documentType) as doctype ,count(c.replycount) as repl
ycount FROM articleyear2018 a LEFT OUTER JOIN commentyear2018 c ON (a.articlewordcount = c.articlewordcount) where a.docum
entType ="blogpost"  Group BY a.Month_Name;
+---------------+----------+-------------+
| a.month_name  | doctype  | replycount  |
+---------------+----------+-------------+
| January       | 9549     | 9549        |
+---------------+----------+-------------+
```

Query 3: What is the reply count for each comment type.

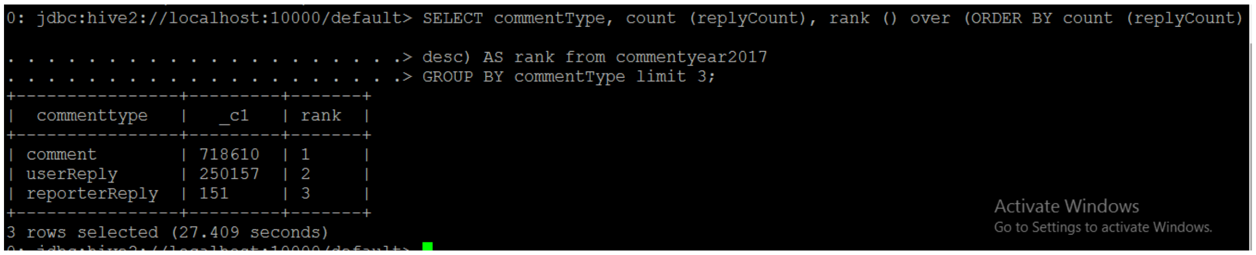
Below query shows the reply count received for top 3 comment type for year 2017 & 2018. Rank is used to get the desired output serially and is Grouped by comment type.

For Year 2017:

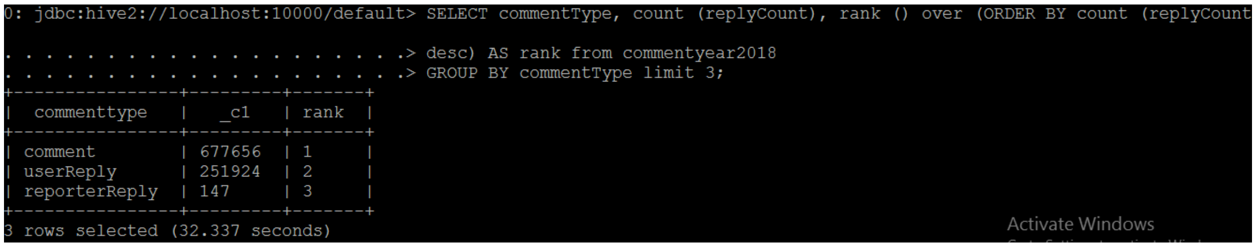
```
SELECT commentType, count (replyCount), rank () over (ORDER BY count (replyCount)
desc) AS rank from commentyear2017
GROUP BY commentType limit 3;
```

For Year 2018:

```
SELECT commentType, count (replyCount), rank () over (ORDER BY count (replyCount)
desc) AS rank from commentyear2018
GROUP BY commentType limit 3;
```



```
0: jdbc:hive2://localhost:10000/default> SELECT commentType, count (replyCount), rank () over (ORDER BY count (replyCount)

. . . . . . . . . . . . . . . . . . . . .> desc) AS rank from commentyear2017
. . . . . . . . . . . . . . . . . . . . .> GROUP BY commentType limit 3;
+----------------+---------+-------+
|  commenttype   |   _c1   | rank  |
+----------------+---------+-------+
| comment        | 718610  | 1     |
| userReply      | 250157  | 2     |
| reporterReply  | 151     | 3     |
+----------------+---------+-------+
3 rows selected (27.409 seconds)
```



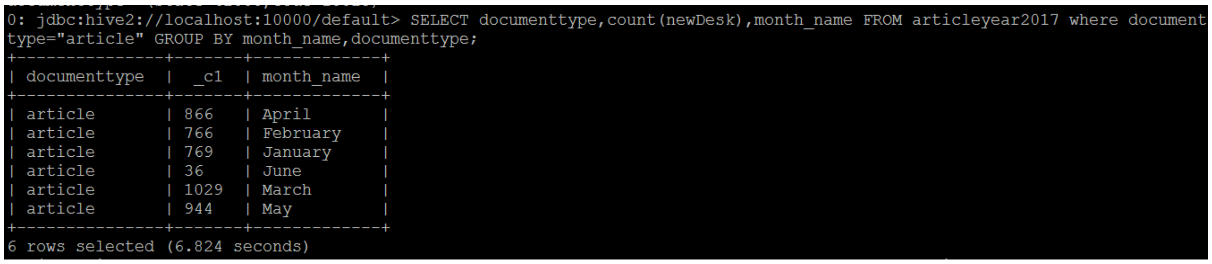
```
0: jdbc:hive2://localhost:10000/default> SELECT commentType, count (replyCount), rank () over (ORDER BY count (replyCount

. . . . . . . . . . . . . . . . . . . . .> desc) AS rank from commentyear2018
. . . . . . . . . . . . . . . . . . . . .> GROUP BY commentType limit 3;
+----------------+---------+-------+
|  commenttype   |   _c1   | rank  |
+----------------+---------+-------+
| comment        | 677656  | 1     |
| userReply      | 251924  | 2     |
| reporterReply  | 147     | 3     |
+----------------+---------+-------+
3 rows selected (32.337 seconds)
```

Query 4: What is the count of new desk month wise?

NewDesk is a column which has various field values like letter, foreign, editorial, brief, etc. In this query we have tried to find out the count of NewDesk received for both the years month wise for document type: Article/ Blogpost

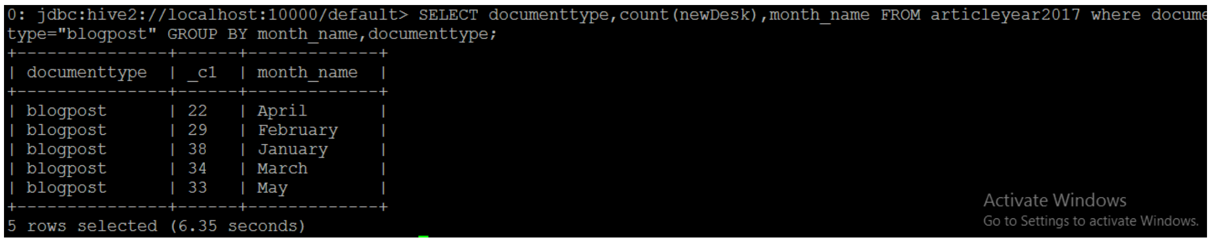
For year 2017:

```
SELECT documenttype, count(newDesk),month_name FROM articleyear2017 where
documenttype=”article” GROUP BY month_name;
```



```
0: jdbc:hive2://localhost:10000/default> SELECT documenttype,count(newDesk),month_name FROM articleyear2017 where document
type="article" GROUP BY month_name,documenttype;
+---------------+-------+-------------+
| documenttype  |  _c1  | month_name  |
+---------------+-------+-------------+
| article       | 866   | April       |
| article       | 766   | February    |
| article       | 769   | January     |
| article       | 36    | June        |
| article       | 1029  | March       |
| article       | 944   | May         |
+---------------+-------+-------------+
6 rows selected (6.824 seconds)
```

SELECT documenttype, count(newDesk),month_name FROM articleyear2017 where documenttype=”blogpost” GROUP BY month_name;

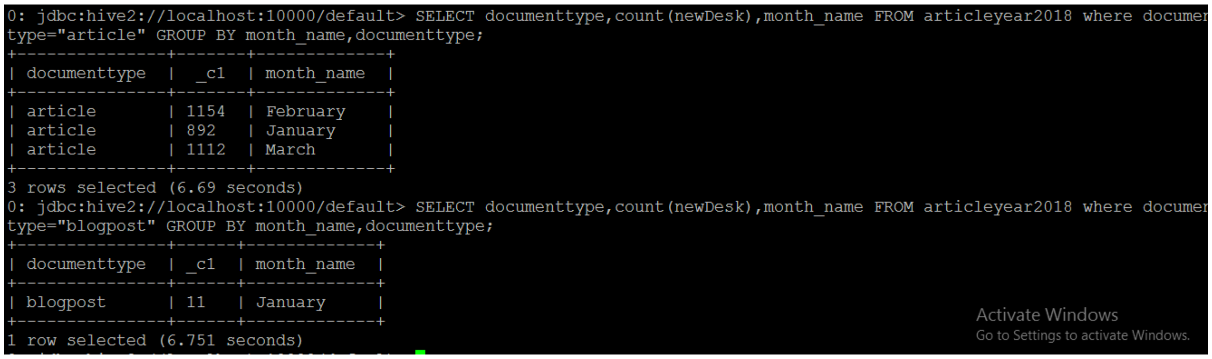


```
0: jdbc:hive2://localhost:10000/default> SELECT documenttype,count(newDesk),month_name FROM articleyear2017 where docume
type="blogpost" GROUP BY month_name,documenttype;
+---------------+------+-------------+
| documenttype  | _c1  | month_name  |
+---------------+------+-------------+
| blogpost      | 22   | April       |
| blogpost      | 29   | February    |
| blogpost      | 38   | January     |
| blogpost      | 34   | March       |
| blogpost      | 33   | May         |
+---------------+------+-------------+
5 rows selected (6.35 seconds)
```

For year 2018:

SELECT documenttype, count(newDesk),month_name FROM articleyear2018 where documenttype=”article” GROUP BY month_name;

SELECT documenttype, count(newDesk),month_name FROM articleyear2018 where documenttype=”blogpost” GROUP BY month_name;



```
0: jdbc:hive2://localhost:10000/default> SELECT documenttype,count(newDesk),month_name FROM articleyear2018 where documen
type="article" GROUP BY month_name,documenttype;
+---------------+-------+-------------+
| documenttype  |  _c1  | month_name  |
+---------------+-------+-------------+
| article       | 1154  | February    |
| article       | 892   | January     |
| article       | 1112  | March       |
+---------------+-------+-------------+
3 rows selected (6.69 seconds)
0: jdbc:hive2://localhost:10000/default> SELECT documenttype,count(newDesk),month_name FROM articleyear2018 where documen
type="blogpost" GROUP BY month_name,documenttype;
+---------------+------+-------------+
| documenttype  | _c1  | month_name  |
+---------------+------+-------------+
| blogpost      | 11   | January     |
+---------------+------+-------------+
1 row selected (6.751 seconds)
```

Query 5: What is the count of new desk based on recommendations?

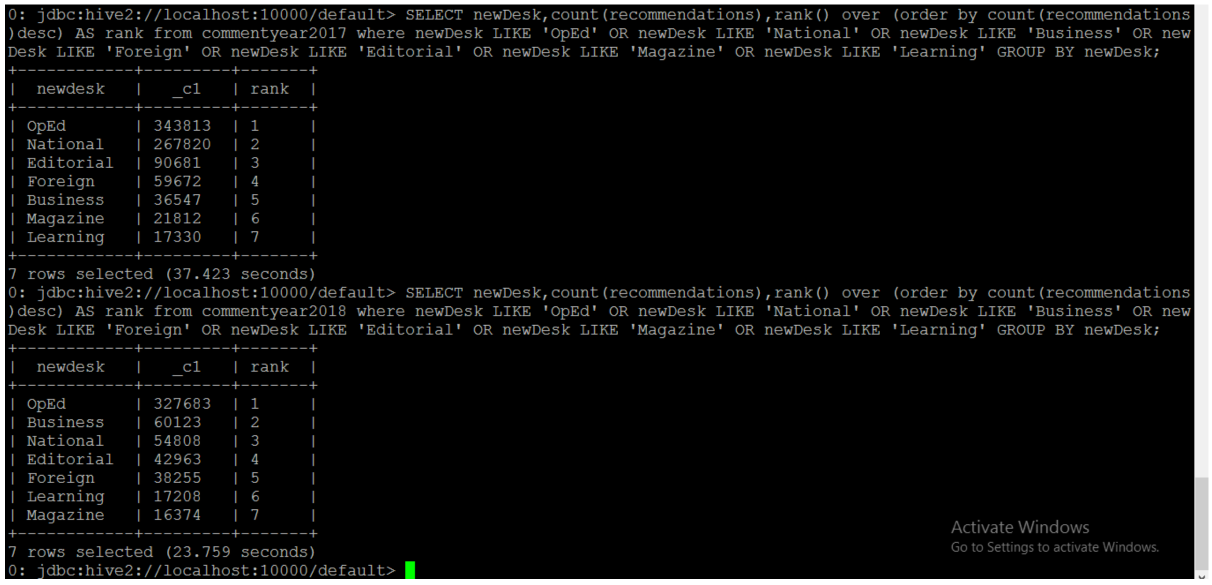
As explained above that newDesk has various filed values and each of them receive some sort of recommendations from the people, which we have shown in the query below for both the years.

For year 2017:

SELECT newDesk,count(recommendations),rank() over (order by count(recommendations)desc) AS rank from commentyear2017 where newDesk LIKE 'OpEd' OR newDesk LIKE 'National' OR newDesk LIKE 'Business' OR newDesk LIKE 'Foreign' OR newDesk LIKE 'Editorial' OR newDesk LIKE 'Magazine' OR newDesk LIKE 'Learning' GROUP BY newDesk;

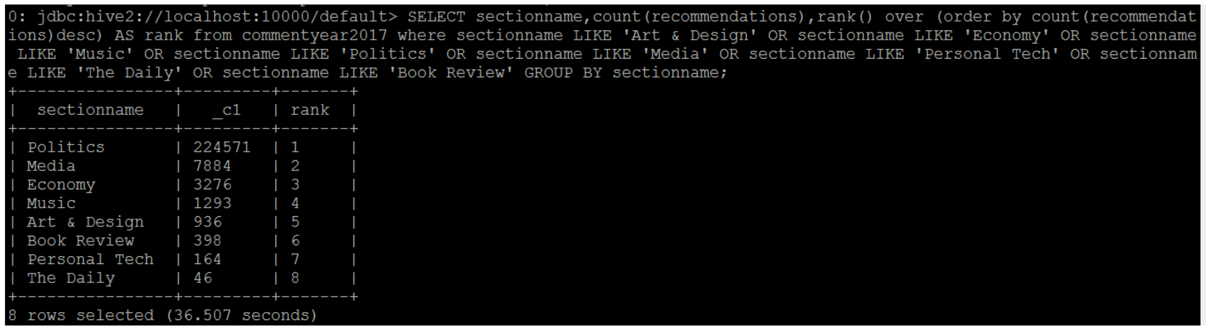
For year 2018:

SELECT newDesk,count(recommendations),rank() over (order by count(recommendations)desc) AS rank from commentyear2018 where newDesk LIKE 'OpEd' OR newDesk LIKE 'National' OR newDesk LIKE 'Business' OR newDesk LIKE 'Foreign' OR newDesk LIKE 'Editorial' OR newDesk LIKE 'Magazine' OR newDesk LIKE 'Learning' GROUP BY newDesk;



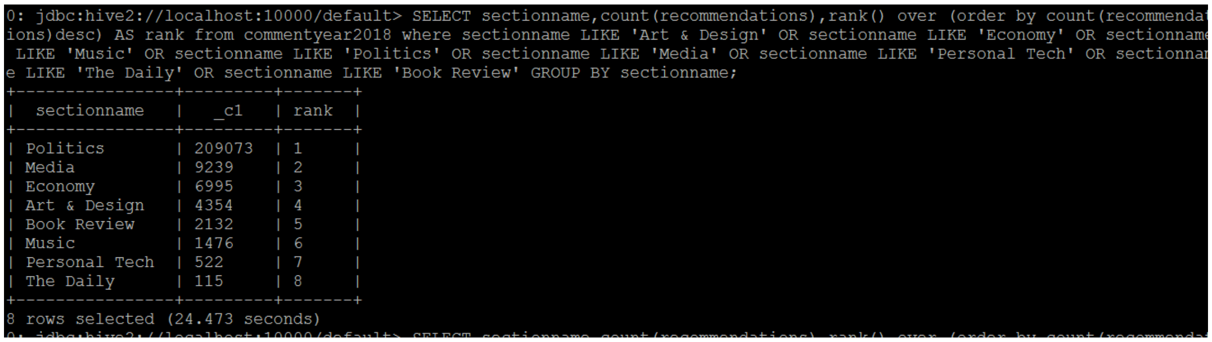
```
0: jdbc:hive2://localhost:10000/default> SELECT newDesk,count(recommendations),rank() over (order by count(recommendations
)desc) AS rank from commentyear2017 where newDesk LIKE 'OpEd' OR newDesk LIKE 'National' OR newDesk LIKE 'Business' OR new
Desk LIKE 'Foreign' OR newDesk LIKE 'Editorial' OR newDesk LIKE 'Magazine' OR newDesk LIKE 'Learning' GROUP BY newDesk;
+------------+---------+-------+
|  newdesk   |   _c1   | rank  |
+------------+---------+-------+
| OpEd       | 343813  | 1     |
| National   | 267820  | 2     |
| Editorial  | 90681   | 3     |
| Foreign    | 59672   | 4     |
| Business   | 36547   | 5     |
| Magazine   | 21812   | 6     |
| Learning   | 17330   | 7     |
+------------+---------+-------+
7 rows selected (37.423 seconds)
0: jdbc:hive2://localhost:10000/default> SELECT newDesk,count(recommendations),rank() over (order by count(recommendations
)desc) AS rank from commentyear2018 where newDesk LIKE 'OpEd' OR newDesk LIKE 'National' OR newDesk LIKE 'Business' OR new
Desk LIKE 'Foreign' OR newDesk LIKE 'Editorial' OR newDesk LIKE 'Magazine' OR newDesk LIKE 'Learning' GROUP BY newDesk;
+------------+---------+-------+
|  newdesk   |   _c1   | rank  |
+------------+---------+-------+
| OpEd       | 327683  | 1     |
| Business   | 60123   | 2     |
| National   | 54808   | 3     |
| Editorial  | 42963   | 4     |
| Foreign    | 38255   | 5     |
| Learning   | 17208   | 6     |
| Magazine   | 16374   | 7     |
+------------+---------+-------+
7 rows selected (23.759 seconds)
0: jdbc:hive2://localhost:10000/default>
```

SELECT sectionname,count(recommendations),rank() over (order by count(recommendations)desc) AS rank from commentyear2017 where sectionname LIKE ‘Art & Design’ OR sectionname LIKE ‘Economy’OR sectionname LIKE ‘Music’ OR sectionname LIKE ‘Media’ OR sectionname LIKE ‘Personal Tech’OR sectionname LIKE ‘The Daily’ OR sectionname LIKE 'Book Review' GROUP BY sectionname;



```
0: jdbc:hive2://localhost:10000/default> SELECT sectionname,count(recommendations),rank() over (order by count(recommendat
ions)desc) AS rank from commentyear2017 where sectionname LIKE 'Art & Design' OR sectionname LIKE 'Economy' OR sectionname
 LIKE 'Music' OR sectionname LIKE 'Politics' OR sectionname LIKE 'Media' OR sectionname LIKE 'Personal Tech' OR sectionnam
e LIKE 'The Daily' OR sectionname LIKE 'Book Review' GROUP BY sectionname;
+----------------+---------+-------+
|  sectionname   |   _c1   | rank  |
+----------------+---------+-------+
| Politics       | 224571  | 1     |
| Media          | 7884    | 2     |
| Economy        | 3276    | 3     |
| Music          | 1293    | 4     |
| Art & Design   | 936     | 5     |
| Book Review    | 398     | 6     |
| Personal Tech  | 164     | 7     |
| The Daily      | 46      | 8     |
+----------------+---------+-------+
8 rows selected (36.507 seconds)
```

SELECT sectionname,count(recommendations),rank() over (order by count(recommendations)desc) AS rank from commentyear2018 where sectionname LIKE ‘Art & Design’ OR sectionname LIKE ‘Economy’OR sectionname LIKE ‘Music’ OR sectionname LIKE ‘Media’ OR sectionname LIKE ‘Personal Tech’OR sectionname LIKE ‘The Daily’ OR sectionname LIKE 'Book Review' GROUP BY sectionname;

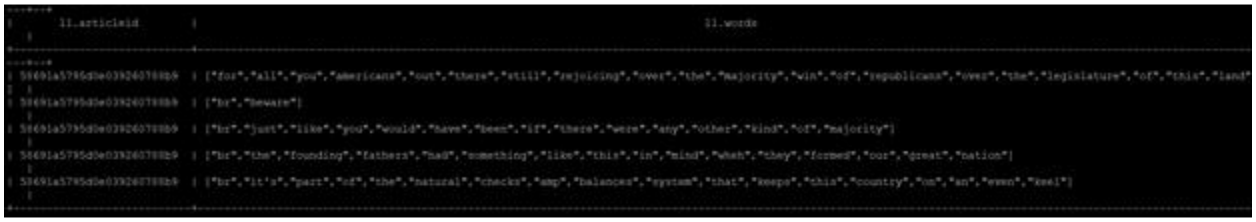


```
0: jdbc:hive2://localhost:10000/default> SELECT sectionname,count(recommendations),rank() over (order by count(recommendat
ions)desc) AS rank from commentyear2018 where sectionname LIKE 'Art & Design' OR sectionname LIKE 'Economy' OR sectionname
 LIKE 'Music' OR sectionname LIKE 'Politics' OR sectionname LIKE 'Media' OR sectionname LIKE 'Personal Tech' OR sectionnam
e LIKE 'The Daily' OR sectionname LIKE 'Book Review' GROUP BY sectionname;
+----------------+---------+-------+
|  sectionname   |   _c1   | rank  |
+----------------+---------+-------+
| Politics       | 209073  | 1     |
| Media          | 9239    | 2     |
| Economy        | 6995    | 3     |
| Art & Design   | 4354    | 4     |
| Book Review    | 2132    | 5     |
| Music          | 1476    | 6     |
| Personal Tech  | 522     | 7     |
| The Daily      | 115     | 8     |
+----------------+---------+-------+
8 rows selected (24.473 seconds)
```

Query 6: What is the degree of polarity by most positive headlines for the year 2017?

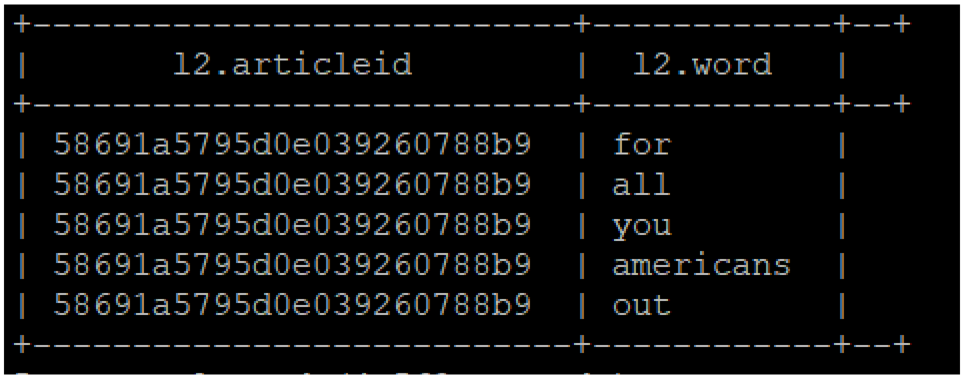
Here we created a view using the function Sentences() which splits the string present in the comment body into arrays of sentences , where each sentence is an array of words. You have your data set as arrays of words which are then lateral view exploded at the first level using the function Explode().

```
create view IF NOT EXISTS l1 as
select articleid,words
from commentyear2017
lateral view explode(sentences(lower(commentbody))) dummy as words;
```



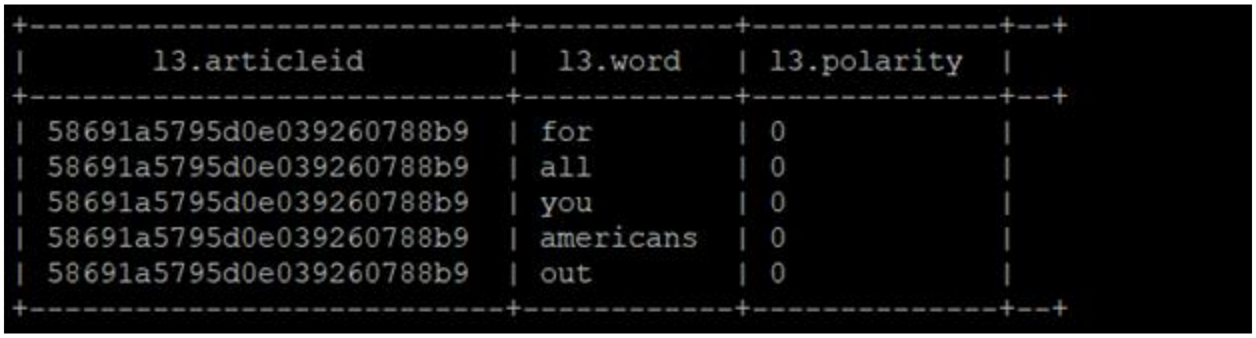
```
|       l1.articleid        |                                   l1.words
+---------------------------+----------------------------------------------------------------------------------------
| 58691a5795d0e039260788b9  | ["for","all","you","americans","out","there","still","rejoicing","over","the","majority","win","of","republicans","over","the","legislature","of","this","land"
| 58691a5795d0e039260788b9  | ["br","beware"]
| 58691a5795d0e039260788b9  | ["br","just","like","you","would","have","been","if","there","were","any","other","kind","of","majority"]
| 58691a5795d0e039260788b9  | ["br","the","founding","fathers","had","something","like","this","in","mind","when","they","formed","our","great","nation"]
| 58691a5795d0e039260788b9  | ["br","it's","part","of","the","natural","checks","amp","balances","system","that","keeps","this","country","on","an","even","keel"]
+---------------------------+----------------------------------------------------------------------------------------
```

```
create view IF NOT EXISTS l2 as
select articleid, word
from l1
lateral view explode(words) dummy as word;
```



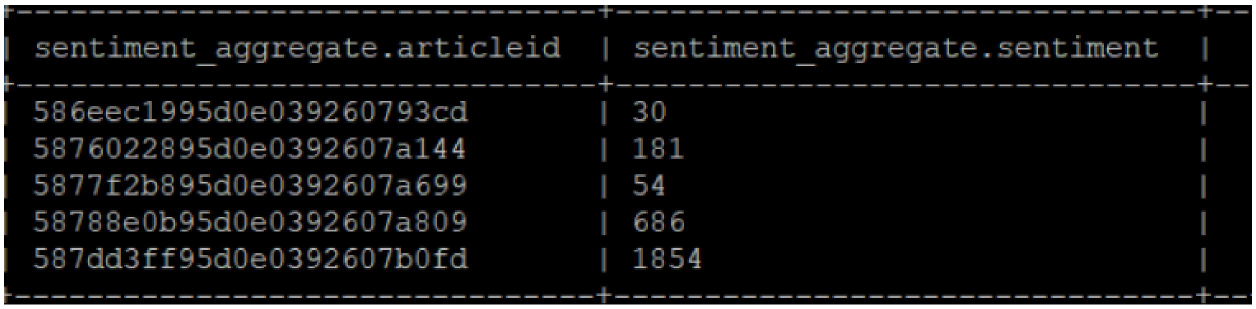
```
+---------------------------+------------+--+
|       l2.articleid        |  l2.word   |
+---------------------------+------------+--+
| 58691a5795d0e039260788b9  | for        |
| 58691a5795d0e039260788b9  | all        |
| 58691a5795d0e039260788b9  | you        |
| 58691a5795d0e039260788b9  | americans  |
| 58691a5795d0e039260788b9  | out        |
+---------------------------+------------+--+
```

```
create view IF NOT EXISTS l3 as select
articleid,
l2.word,
case d.polarity
when 'negative' then -1
when 'positive' then 1
else 0 end as polarity
from l2 left outer join dictionary d on l2.word = d.word;
```



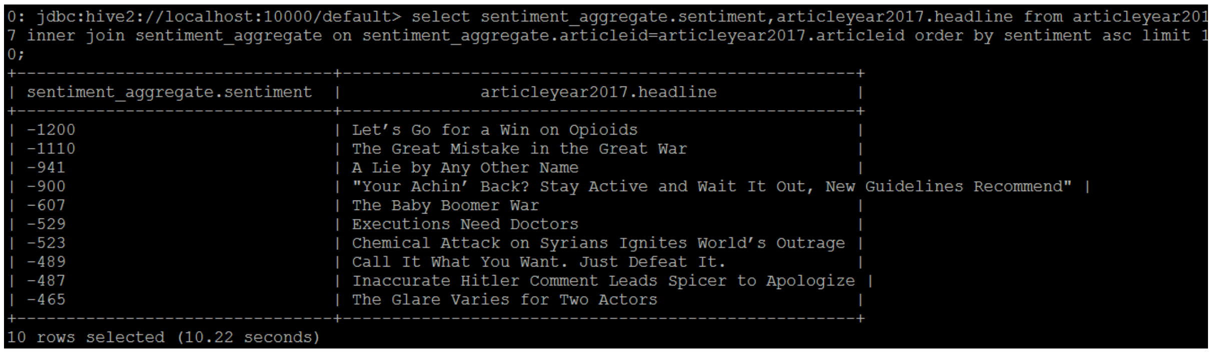
```
+---------------------------+------------+--------------+--+
|       l3.articleid        |  l3.word   | l3.polarity  |
+---------------------------+------------+--------------+--+
| 58691a5795d0e039260788b9  | for        | 0            |
| 58691a5795d0e039260788b9  | all        | 0            |
| 58691a5795d0e039260788b9  | you        | 0            |
| 58691a5795d0e039260788b9  | americans  | 0            |
| 58691a5795d0e039260788b9  | out        | 0            |
+---------------------------+------------+--------------+--+
```

```
create table IF NOT EXISTS sentiment_aggregate
stored as orc as select
articleid,sum( polarity ) sentiment
from l3 group by articleid;
```



```
+-------------------------------+-------------------------------+--
| sentiment_aggregate.articleid  | sentiment_aggregate.sentiment  |
+-------------------------------+-------------------------------+--
| 586eec1995d0e039260793cd       | 30                             |
| 5876022895d0e0392607a144       | 181                            |
| 5877f2b895d0e0392607a699       | 54                             |
| 58788e0b95d0e0392607a809       | 686                            |
| 587dd3ff95d0e0392607b0fd       | 1854                           |
+-------------------------------+-------------------------------+--
```

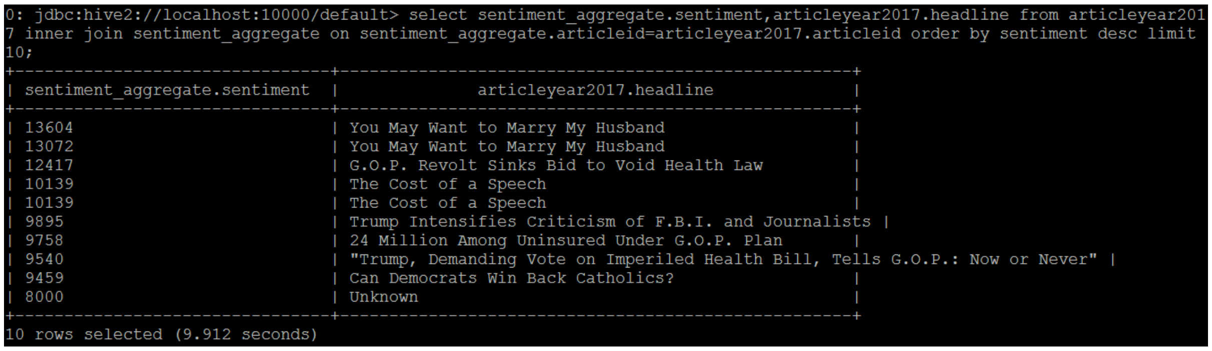
select sentiment_aggregate.sentiment,articleyear2017.headline from articleyear2017 inner join sentiment_aggregate on sentiment_aggregate.articleid=articleyear2017.articleid order by sentiment asc limit 10;



```
0: jdbc:hive2://localhost:10000/default> select sentiment_aggregate.sentiment,articleyear2017.headline from articleyear201
7 inner join sentiment_aggregate on sentiment_aggregate.articleid=articleyear2017.articleid order by sentiment asc limit 1
0;
+-------------------------------+---------------------------------------------------+
| sentiment_aggregate.sentiment  |              articleyear2017.headline              |
+-------------------------------+---------------------------------------------------+
| -1200                          | Let’s Go for a Win on Opioids                     |
| -1110                          | The Great Mistake in the Great War                |
| -941                           | A Lie by Any Other Name                           |
| -900                           | "Your Achin’ Back? Stay Active and Wait It Out, New Guidelines Recommend" |
| -607                           | The Baby Boomer War                               |
| -529                           | Executions Need Doctors                           |
| -523                           | Chemical Attack on Syrians Ignites World’s Outrage |
| -489                           | Call It What You Want. Just Defeat It.            |
| -487                           | Inaccurate Hitler Comment Leads Spicer to Apologize |
| -465                           | The Glare Varies for Two Actors                   |
+-------------------------------+---------------------------------------------------+
10 rows selected (10.22 seconds)
```

Query 7: What is the degree of polarity by most negative headlines?

select sentiment_aggregate.sentiment,articleyear2017.headline from articleyear2017 inner join sentiment_aggregate on sentiment_aggregate.articleid=articleyear2017.articleid order by sentiment desc limit 10;



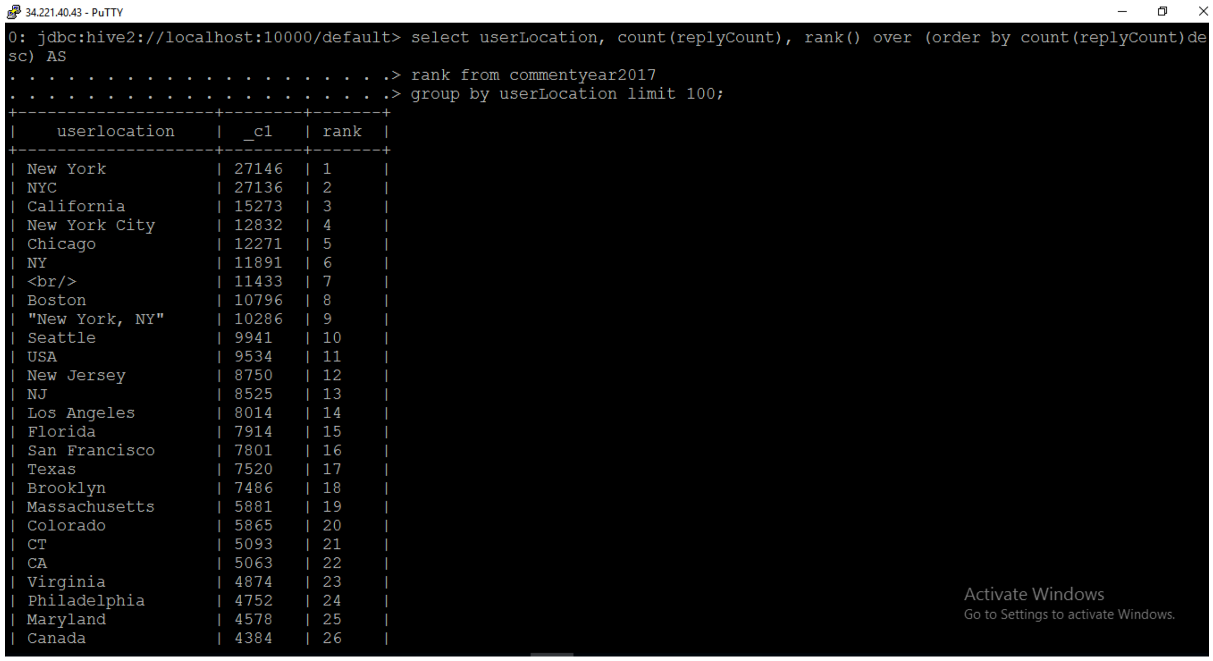
```
0: jdbc:hive2://localhost:10000/default> select sentiment_aggregate.sentiment,articleyear2017.headline from articleyear201
7 inner join sentiment_aggregate on sentiment_aggregate.articleid=articleyear2017.articleid order by sentiment desc limit
10;
+--------------------------------+---------------------------------------------------+
| sentiment_aggregate.sentiment  |             articleyear2017.headline              |
+--------------------------------+---------------------------------------------------+
| 13604                          | You May Want to Marry My Husband                  |
| 13072                          | You May Want to Marry My Husband                  |
| 12417                          | G.O.P. Revolt Sinks Bid to Void Health Law        |
| 10139                          | The Cost of a Speech                              |
| 10139                          | The Cost of a Speech                              |
| 9895                           | Trump Intensifies Criticism of F.B.I. and Journalists |
| 9758                           | 24 Million Among Uninsured Under G.O.P. Plan      |
| 9540                           | "Trump, Demanding Vote on Imperiled Health Bill, Tells G.O.P.: Now or Never" |
| 9459                           | Can Democrats Win Back Catholics?                 |
| 8000                           | Unknown                                           |
+--------------------------------+---------------------------------------------------+
10 rows selected (9.912 seconds)
```

Query 9: Where is most active user's from based on replycount for the year 2017?

Reply counts are received for people and various users of NYT. These people and users could be present at different locations and so we have tried to get a count of reply count that we receive from different locations across the U.S for both the years.

For year 2017:

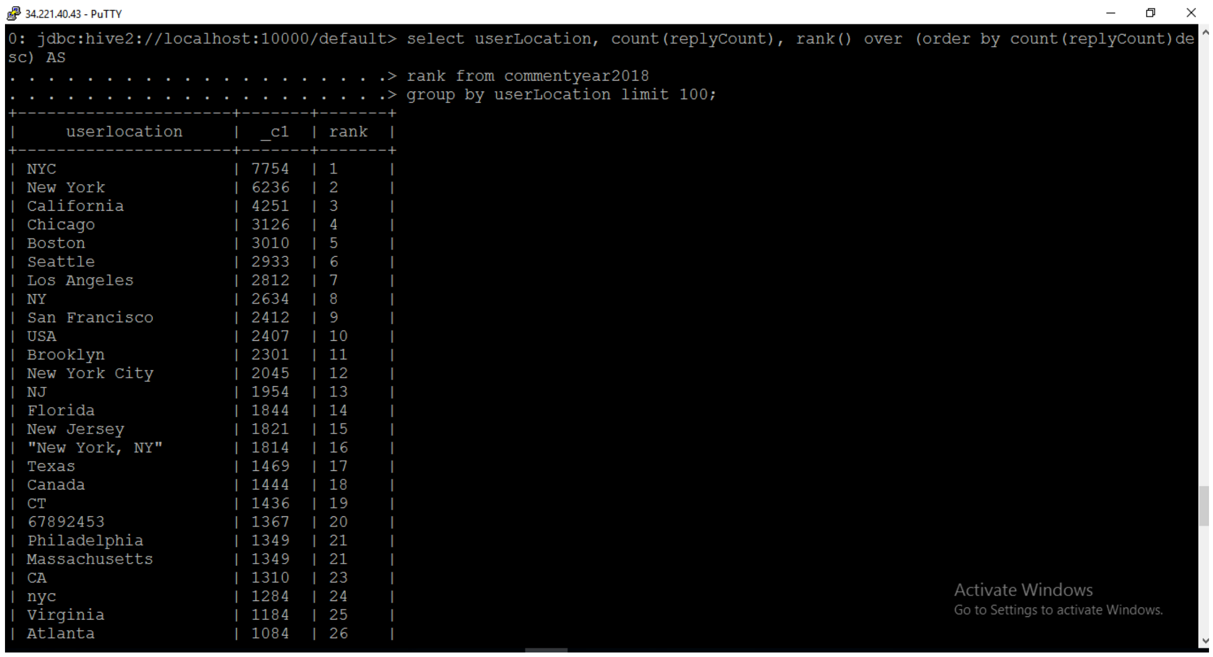
```
select userLocation, count(replycount), rank() over (order by count(replycount)desc) AS
rank from commentyear2017
group by userLocation limit 100;
```



```
0: jdbc:hive2://localhost:10000/default> select userLocation, count(replyCount), rank() over (order by count(replyCount)de
sc) AS
. . . . . . . . . . . . . . . . . . . .> rank from commentyear2017
. . . . . . . . . . . . . . . . . . . .> group by userLocation limit 100;
+---------------------+--------+-------+
|    userlocation     |  _c1   | rank  |
+---------------------+--------+-------+
| New York            | 27146  | 1     |
| NYC                 | 27136  | 2     |
| California          | 15273  | 3     |
| New York City       | 12832  | 4     |
| Chicago             | 12271  | 5     |
| NY                  | 11891  | 6     |
| <br/>               | 11433  | 7     |
| Boston              | 10796  | 8     |
| "New York, NY"      | 10286  | 9     |
| Seattle             | 9941   | 10    |
| USA                 | 9534   | 11    |
| New Jersey          | 8750   | 12    |
| NJ                  | 8525   | 13    |
| Los Angeles         | 8014   | 14    |
| Florida             | 7914   | 15    |
| San Francisco       | 7801   | 16    |
| Texas               | 7520   | 17    |
| Brooklyn            | 7486   | 18    |
| Massachusetts       | 5881   | 19    |
| Colorado            | 5865   | 20    |
| CT                  | 5093   | 21    |
| CA                  | 5063   | 22    |
| Virginia            | 4874   | 23    |
| Philadelphia        | 4752   | 24    |
| Maryland            | 4578   | 25    |
| Canada              | 4384   | 26    |
```

Query 10: Where is most active user's from based on replycount for the year 2018?

```
select userLocation, count(replycount), rank() over (order by count(replycount)desc) AS
rank from commentyear2018
group by userLocation limit 100;
```



```
0: jdbc:hive2://localhost:10000/default> select userLocation, count(replyCount), rank() over (order by count(replyCount)de
sc) AS
. . . . . . . . . . . . . . . . . . . . .> rank from commentyear2018
. . . . . . . . . . . . . . . . . . . . .> group by userLocation limit 100;
+----------------------+-------+-------+
|     userlocation     |  _c1  | rank  |
+----------------------+-------+-------+
| NYC                  | 7754  | 1     |
| New York             | 6236  | 2     |
| California           | 4251  | 3     |
| Chicago              | 3126  | 4     |
| Boston               | 3010  | 5     |
| Seattle              | 2933  | 6     |
| Los Angeles          | 2812  | 7     |
| NY                   | 2634  | 8     |
| San Francisco        | 2412  | 9     |
| USA                  | 2407  | 10    |
| Brooklyn             | 2301  | 11    |
| New York City        | 2045  | 12    |
| NJ                   | 1954  | 13    |
| Florida              | 1844  | 14    |
| New Jersey           | 1821  | 15    |
| "New York, NY"       | 1814  | 16    |
| Texas                | 1469  | 17    |
| Canada               | 1444  | 18    |
| CT                   | 1436  | 19    |
| 67892453             | 1367  | 20    |
| Philadelphia         | 1349  | 21    |
| Massachusetts        | 1349  | 21    |
| CA                   | 1310  | 23    |
| nyc                  | 1284  | 24    |
| Virginia             | 1184  | 25    |
| Atlanta              | 1084  | 26    |
```

Query 12: Most Popular Author(byline) with respect to recommendations of public for the year 2017 ?

First, we find out the sum of recommendations as per each unique articleid with help of following query

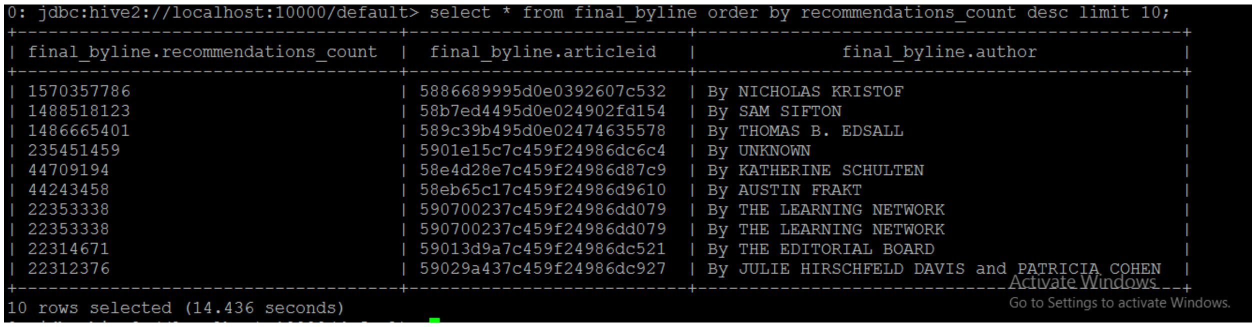
```
create table if not exists shradha_byline2 as select sum(recommendations) as
recommendations,articleid from commentyear2017 group by articleid;
```

We create a new table which will store the results of the above output as well map the author with the help of inner join.

```
create table final_byline as select shradha_byline2.recommendations
recommendations_count,shradha_byline2.articleid articleid,articleyear2017.byline author from
shradha_byline2 inner join articleyear2017 on shradha_byline2.articleid =
articleyear2017.articleid;
```

Lastly, we find out the most popular author - byline with help of below query

```
select * from final_byline order by recommendations_count desc limit 10;
```



```
0: jdbc:hive2://localhost:10000/default> select * from final_byline order by recommendations_count desc limit 10;
+-------------------------------------+---------------------------+---------------------------------------------+
| final_byline.recommendations_count  |  final_byline.articleid   |             final_byline.author             |
+-------------------------------------+---------------------------+---------------------------------------------+
| 1570357786                          | 5886689995d0e0392607c532  | By NICHOLAS KRISTOF                         |
| 1488518123                          | 58b7ed4495d0e024902fd154  | By SAM SIFTON                               |
| 1486665401                          | 589c39b495d0e02474635578  | By THOMAS B. EDSALL                         |
| 235451459                           | 5901e15c7c459f24986dc6c4  | By UNKNOWN                                  |
| 44709194                            | 58e4d28e7c459f24986d87c9  | By KATHERINE SCHULTEN                       |
| 44243458                            | 58eb65c17c459f24986d9610  | By AUSTIN FRAKT                             |
| 22353338                            | 590700237c459f24986dd079  | By THE LEARNING NETWORK                     |
| 22353338                            | 590700237c459f24986dd079  | By THE LEARNING NETWORK                     |
| 22314671                            | 59013d9a7c459f24986dc521  | By THE EDITORIAL BOARD                      |
| 22312376                            | 59029a437c459f24986dc927  | By JULIE HIRSCHFELD DAVIS and PATRICIA COHEN |
+-------------------------------------+---------------------------+---------------------------------------------+
10 rows selected (14.436 seconds)
```

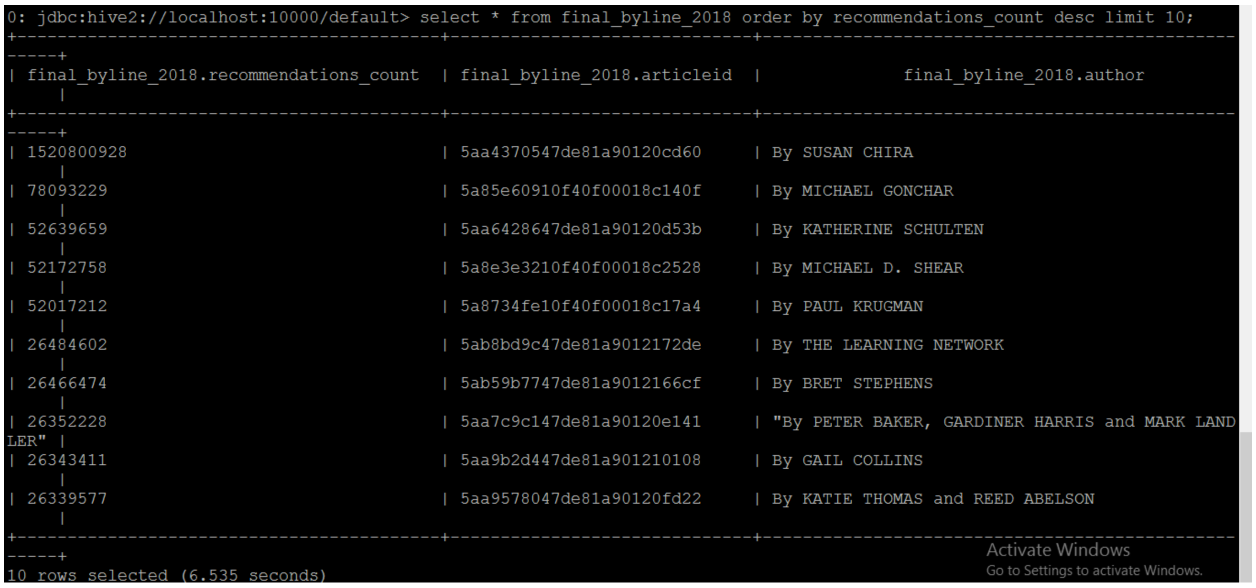
Query 13: Most Popular Author(byline) with respect to recommendations of public for the year 2018?

Similarly follow the above commands to find out the most popular author - byline with respect to the recommendations of public in the year 2018.

```
create table if not exists shradha_byline2_2018 as select sum(recommendations) as
recommendations,articleid from commentyear2018 group by articleid;
```

```
create table final_byline_2018 as select shradha_byline2_2018.recommendations
recommendations_count,shradha_byline2_2018.articleid articleid,articleyear2018.byline author
from shradha_byline2_2018 inner join articleyear2018 on shradha_byline2_2018.articleid =
articleyear2018.articleid;
```

```
select * from final_byline_2018 order by recommendations_count desc limit 10;
```



```
0: jdbc:hive2://localhost:10000/default> select * from final_byline_2018 order by recommendations_count desc limit 10;
+-------------------------------------------+-----------------------------+----------------------------------------------------
-----+
| final_byline_2018.recommendations_count  | final_byline_2018.articleid  |              final_byline_2018.author
     |
+-------------------------------------------+-----------------------------+----------------------------------------------------
-----+
| 1520800928                                | 5aa4370547de81a90120cd60     | By SUSAN CHIRA
     |
| 78093229                                  | 5a85e60910f40f00018c140f     | By MICHAEL GONCHAR
     |
| 52639659                                  | 5aa6428647de81a90120d53b     | By KATHERINE SCHULTEN
     |
| 52172758                                  | 5a8e3e3210f40f00018c2528     | By MICHAEL D. SHEAR
     |
| 52017212                                  | 5a8734fe10f40f00018c17a4     | By PAUL KRUGMAN
     |
| 26484602                                  | 5ab8bd9c47de81a9012172de     | By THE LEARNING NETWORK
     |
| 26466474                                  | 5ab59b7747de81a9012166cf     | By BRET STEPHENS
     |
| 26352228                                  | 5aa7c9c147de81a90120e141     | "By PETER BAKER, GARDINER HARRIS and MARK LAND
LER" |
| 26343411                                  | 5aa9b2d447de81a901210108     | By GAIL COLLINS
     |
| 26339577                                  | 5aa9578047de81a90120fd22     | By KATIE THOMAS and REED ABELSON
     |
+-------------------------------------------+-----------------------------+----------------------------------------------------
-----+
10 rows selected (6.535 seconds)
```

Hence the results for the most popular author - byline with respect to recommendations of public as given above for the year 2018.

DOWNLOADING DATA (OUTPUT FILES) INTO YOUR PC

After the Hive tables are created, we can download it to our personal PC/laptop as follows:

(The following is an example to download the output file for one query, similarly all the output files for all the queries have been downloaded in the same manner)

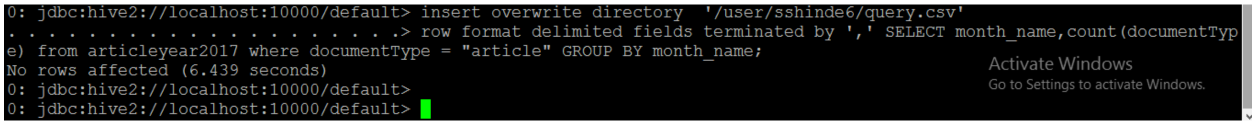
Step 1: Open another terminal Bash and connect it to Beeline which is connected to the Oracle cloud in order to download the output files and type in the following command at beeline:

```
insert overwrite directory  '/user/sshinde6/query.csv'

row format delimited fields terminated by ',' SELECT month_name,count(documentType) from
articleyear2017 where documentType = "article" GROUP BY month_name;
```

For the field marked in Green: Note: query1.csv here is just a sample file name. you can name it anything and accordingly file with that name will be created)

For the field marked in Red: Here, which ever query you wish to run, copy and paste it here, in the field marked red above)

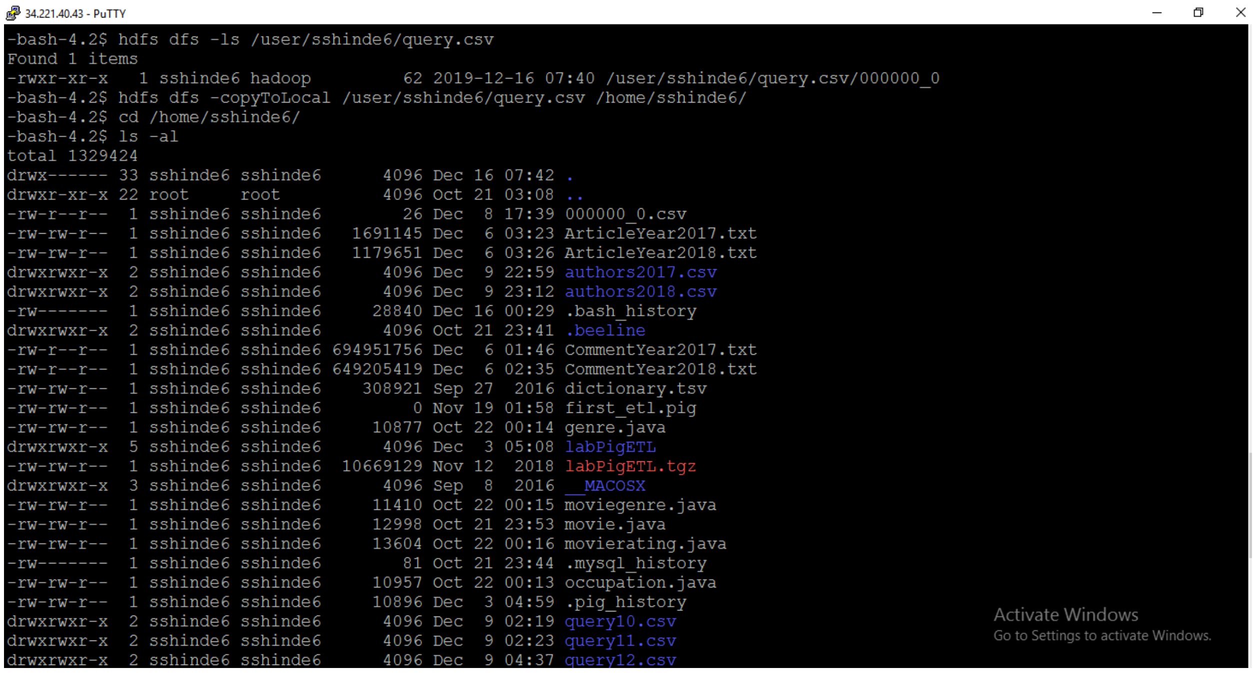
The following will be displayed an output on your screen:



```
0: jdbc:hive2://localhost:10000/default> insert overwrite directory  '/user/sshinde6/query.csv'
. . . . . . . . . . . . . . . . . . . .> row format delimited fields terminated by ',' SELECT month_name,count(documentTyp
e) from articleyear2017 where documentType = "article" GROUP BY month_name;
No rows affected (6.439 seconds)
0: jdbc:hive2://localhost:10000/default>
0: jdbc:hive2://localhost:10000/default>
```

Follow the rest steps as given below:

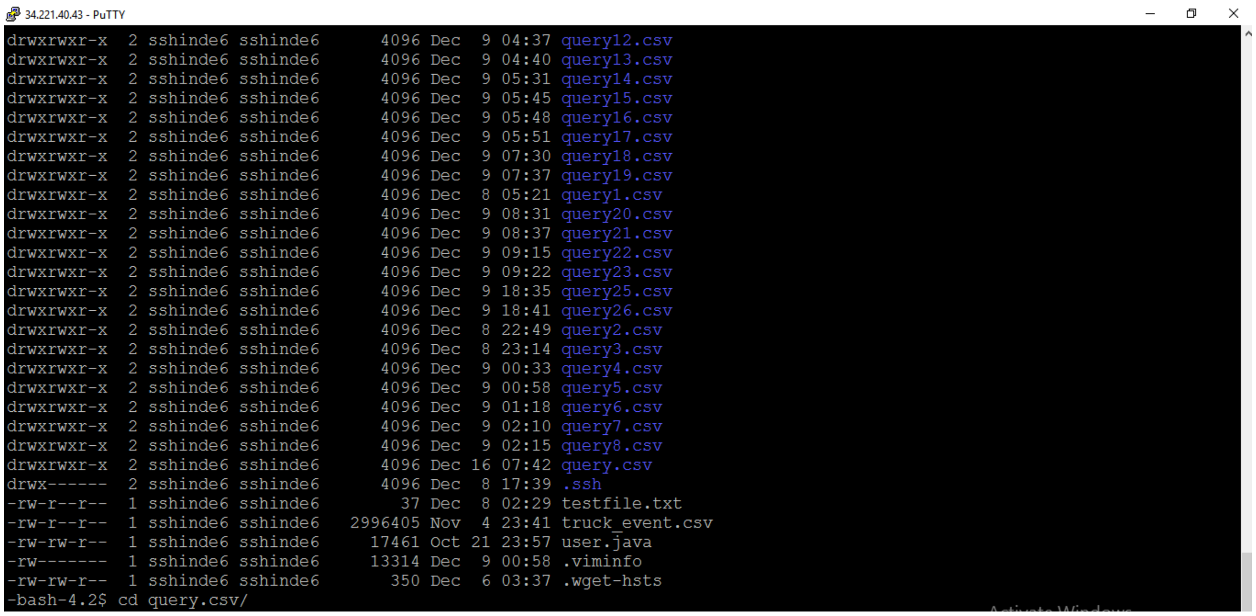
```
Step 2 :  -bash-4.1$ hdfs dfs -ls /user/sshinde6/query.csv

          :  -bash-4.1$ hdfs dfs -copyToLocal /user/sshinde6/query.csv /home/sshinde6/

          :  -bash-4.1$ cd /home/sshinde6/

          :  -bash-4.1$ ls -al

          :  -bash-4.1$ cd query.csv/

          :  -bash-4.1$ vi 000000_0
```



34.221.40.43 - PuTTY

```
-bash-4.2$ hdfs dfs -ls /user/sshinde6/query.csv
Found 1 items
-rwxr-xr-x   1 sshinde6 hadoop         62 2019-12-16 07:40 /user/sshinde6/query.csv/000000_0
-bash-4.2$ hdfs dfs -copyToLocal /user/sshinde6/query.csv /home/sshinde6/
-bash-4.2$ cd /home/sshinde6/
-bash-4.2$ ls -al
total 1329424
drwx------ 33 sshinde6 sshinde6      4096 Dec 16 07:42 .
drwxr-xr-x 22 root     root          4096 Oct 21 03:08 ..
-rw-r--r--  1 sshinde6 sshinde6        26 Dec  8 17:39 000000_0.csv
-rw-rw-r--  1 sshinde6 sshinde6   1691145 Dec  6 03:23 ArticleYear2017.txt
-rw-rw-r--  1 sshinde6 sshinde6   1179651 Dec  6 03:26 ArticleYear2018.txt
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 22:59 authors2017.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 23:12 authors2018.csv
-rw-------  1 sshinde6 sshinde6     28840 Dec 16 00:29 .bash_history
drwxrwxr-x  2 sshinde6 sshinde6      4096 Oct 21 23:41 .beeline
-rw-r--r--  1 sshinde6 sshinde6 694951756 Dec  6 01:46 CommentYear2017.txt
-rw-r--r--  1 sshinde6 sshinde6 649205419 Dec  6 02:35 CommentYear2018.txt
-rw-rw-r--  1 sshinde6 sshinde6    308921 Sep 27  2016 dictionary.tsv
-rw-rw-r--  1 sshinde6 sshinde6         0 Nov 19 01:58 first_etl.pig
-rw-rw-r--  1 sshinde6 sshinde6     10877 Oct 22 00:14 genre.java
drwxrwxr-x  5 sshinde6 sshinde6      4096 Dec  3 05:08 labPigETL
-rw-rw-r--  1 sshinde6 sshinde6  10669129 Nov 12  2018 labPigETL.tgz
drwxrwxr-x  3 sshinde6 sshinde6      4096 Sep  8  2016 __MACOSX
-rw-rw-r--  1 sshinde6 sshinde6     11410 Oct 22 00:15 moviegenre.java
-rw-rw-r--  1 sshinde6 sshinde6     12998 Oct 21 23:53 movie.java
-rw-rw-r--  1 sshinde6 sshinde6     13604 Oct 22 00:16 movierating.java
-rw-------  1 sshinde6 sshinde6        81 Oct 21 23:44 .mysql_history
-rw-rw-r--  1 sshinde6 sshinde6     10957 Oct 22 00:13 occupation.java
-rw-rw-r--  1 sshinde6 sshinde6     10896 Dec  3 04:59 .pig_history
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 02:19 query10.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 02:23 query11.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 04:37 query12.csv
```

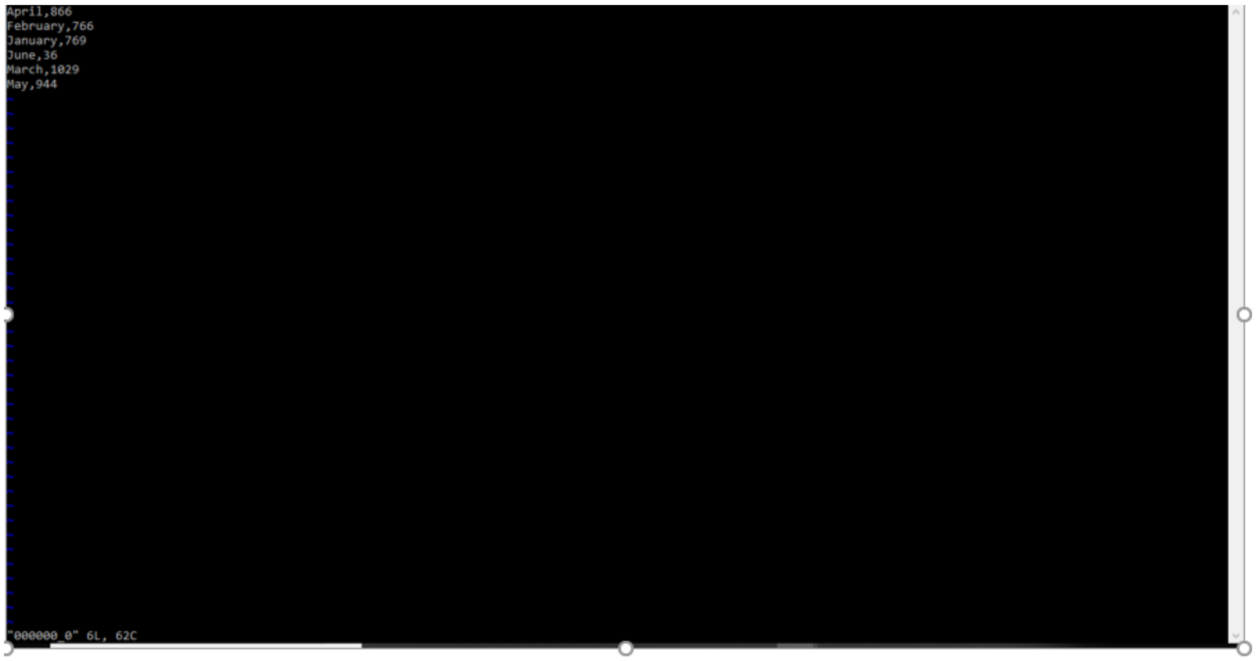
Activate Windows
Go to Settings to activate Windows.



```
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 04:37 query12.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 04:40 query13.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 05:31 query14.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 05:45 query15.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 05:48 query16.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 05:51 query17.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 07:30 query18.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 07:37 query19.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  8 05:21 query1.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 08:31 query20.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 08:37 query21.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 09:15 query22.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 09:22 query23.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 18:35 query25.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 18:41 query26.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  8 22:49 query2.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  8 23:14 query3.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 00:33 query4.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 00:58 query5.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 01:18 query6.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 02:10 query7.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec  9 02:15 query8.csv
drwxrwxr-x  2 sshinde6 sshinde6      4096 Dec 16 07:42 query.csv
drwx------  2 sshinde6 sshinde6      4096 Dec  8 17:39 .ssh
-rw-r--r--  1 sshinde6 sshinde6        37 Dec  8 02:29 testfile.txt
-rw-r--r--  1 sshinde6 sshinde6   2996405 Nov  4 23:41 truck_event.csv
-rw-rw-r--  1 sshinde6 sshinde6     17461 Oct 21 23:57 user.java
-rw-------  1 sshinde6 sshinde6     13314 Dec  9 00:58 .viminfo
-rw-rw-r--  1 sshinde6 sshinde6       350 Dec  6 03:37 .wget-hsts
-bash-4.2$ cd query.csv/
```

When you run the last command of the screenshot, that is, " -bash-4.1$ vi 000000_0" , the output should be the result of your query :

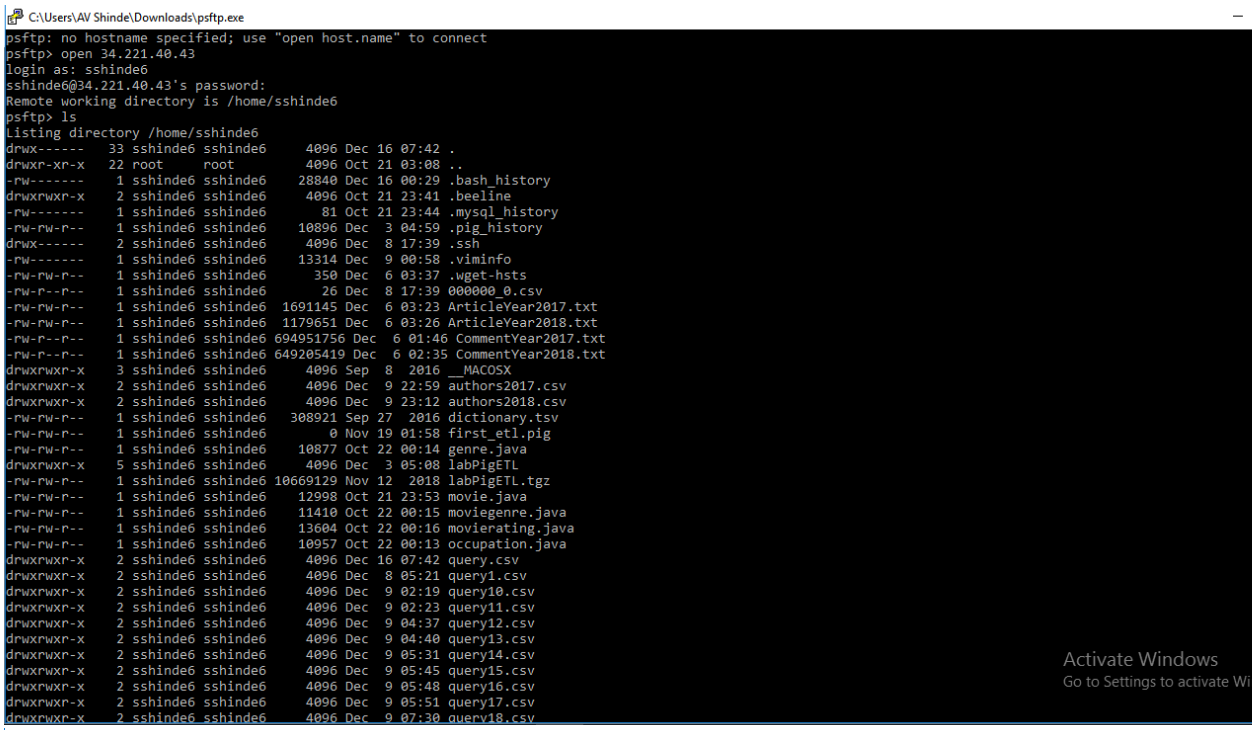
Example: For the query that I have run above, the output is as follows:



```
April,866
February,766
January,769
June,36
March,1029
May,944
"000000_0" 6L, 62C
```

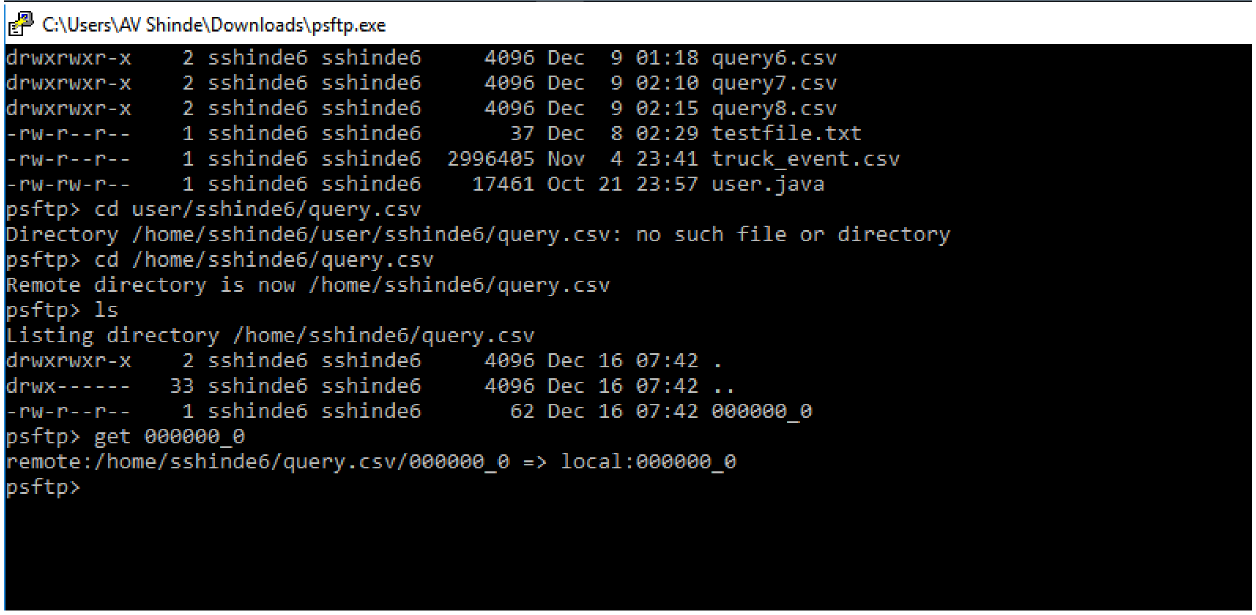
Step 3: Now open pfstp and run the following steps.

(We have done this using pfstp)



C:\Users\AV Shinde\Downloads\psftp.exe

```
psftp: no hostname specified; use "open host.name" to connect
psftp> open 34.221.40.43
login as: sshinde6
sshinde6@34.221.40.43's password:
Remote working directory is /home/sshinde6
psftp> ls
Listing directory /home/sshinde6
drwx------   33 sshinde6 sshinde6     4096 Dec 16 07:42 .
drwxr-xr-x   22 root     root         4096 Oct 21 03:08 ..
-rw-------    1 sshinde6 sshinde6    28840 Dec 16 00:29 .bash_history
drwxrwxr-x    2 sshinde6 sshinde6     4096 Oct 21 23:41 .beeline
-rw-------    1 sshinde6 sshinde6       81 Oct 21 23:44 .mysql_history
-rw-rw-r--    1 sshinde6 sshinde6    10896 Dec  3 04:59 .pig_history
drwx------    2 sshinde6 sshinde6     4096 Dec  8 17:39 .ssh
-rw-------    1 sshinde6 sshinde6    13314 Dec  9 00:58 .viminfo
-rw-rw-r--    1 sshinde6 sshinde6      350 Dec  6 03:37 .wget-hsts
-rw-r--r--    1 sshinde6 sshinde6       26 Dec  8 17:39 000000_0.csv
-rw-rw-r--    1 sshinde6 sshinde6  1691145 Dec  6 03:23 ArticleYear2017.txt
-rw-rw-r--    1 sshinde6 sshinde6  1179651 Dec  6 03:26 ArticleYear2018.txt
-rw-r--r--    1 sshinde6 sshinde6 694951756 Dec  6 01:46 CommentYear2017.txt
-rw-r--r--    1 sshinde6 sshinde6 649205419 Dec  6 02:35 CommentYear2018.txt
drwxrwxr-x    3 sshinde6 sshinde6     4096 Sep  8  2016 __MACOSX
drwxrwxr-x    2 sshinde6 sshinde6     4096 Dec  9 22:59 authors2017.csv
drwxrwxr-x    2 sshinde6 sshinde6     4096 Dec  9 23:12 authors2018.csv
-rw-rw-r--    1 sshinde6 sshinde6   308921 Sep 27  2016 dictionary.tsv
-rw-rw-r--    1 sshinde6 sshinde6        0 Nov 19 01:58 first_etl.pig
-rw-rw-r--    1 sshinde6 sshinde6    10877 Oct 22 00:14 genre.java
drwxrwxr-x    5 sshinde6 sshinde6     4096 Dec  3 05:08 labPigETL
-rw-rw-r--    1 sshinde6 sshinde6 10669129 Nov 12  2018 labPigETL.tgz
-rw-rw-r--    1 sshinde6 sshinde6    12998 Oct 21 23:53 movie.java
-rw-rw-r--    1 sshinde6 sshinde6    11410 Oct 22 00:15 moviegenre.java
-rw-rw-r--    1 sshinde6 sshinde6    13604 Oct 22 00:16 movierating.java
-rw-rw-r--    1 sshinde6 sshinde6    10957 Oct 22 00:13 occupation.java
drwxrwxr-x    2 sshinde6 sshinde6     4096 Dec 16 07:42 query.csv
drwxrwxr-x    2 sshinde6 sshinde6     4096 Dec  8 05:21 query1.csv
drwxrwxr-x    2 sshinde6 sshinde6     4096 Dec  9 02:19 query10.csv
drwxrwxr-x    2 sshinde6 sshinde6     4096 Dec  9 02:23 query11.csv
drwxrwxr-x    2 sshinde6 sshinde6     4096 Dec  9 04:37 query12.csv
drwxrwxr-x    2 sshinde6 sshinde6     4096 Dec  9 04:40 query13.csv
drwxrwxr-x    2 sshinde6 sshinde6     4096 Dec  9 05:31 query14.csv
drwxrwxr-x    2 sshinde6 sshinde6     4096 Dec  9 05:45 query15.csv
drwxrwxr-x    2 sshinde6 sshinde6     4096 Dec  9 05:48 query16.csv
drwxrwxr-x    2 sshinde6 sshinde6     4096 Dec  9 05:51 query17.csv
drwxrwxr-x    2 sshinde6 sshinde6     4096 Dec  9 07:30 query18.csv
```



C:\Users\AV Shinde\Downloads\psftp.exe

```
drwxrwxr-x    2 sshinde6 sshinde6     4096 Dec  9 01:18 query6.csv
drwxrwxr-x    2 sshinde6 sshinde6     4096 Dec  9 02:10 query7.csv
drwxrwxr-x    2 sshinde6 sshinde6     4096 Dec  9 02:15 query8.csv
-rw-r--r--    1 sshinde6 sshinde6       37 Dec  8 02:29 testfile.txt
-rw-r--r--    1 sshinde6 sshinde6  2996405 Nov  4 23:41 truck_event.csv
-rw-rw-r--    1 sshinde6 sshinde6    17461 Oct 21 23:57 user.java
psftp> cd user/sshinde6/query.csv
Directory /home/sshinde6/user/sshinde6/query.csv: no such file or directory
psftp> cd /home/sshinde6/query.csv
Remote directory is now /home/sshinde6/query.csv
psftp> ls
Listing directory /home/sshinde6/query.csv
drwxrwxr-x    2 sshinde6 sshinde6     4096 Dec 16 07:42 .
drwx------   33 sshinde6 sshinde6     4096 Dec 16 07:42 ..
-rw-r--r--    1 sshinde6 sshinde6       62 Dec 16 07:42 000000_0
psftp> get 000000_0
remote:/home/sshinde6/query.csv/000000_0 => local:000000_0
psftp>
```

Step 4: After this go to your local machine and you will find the output file there.

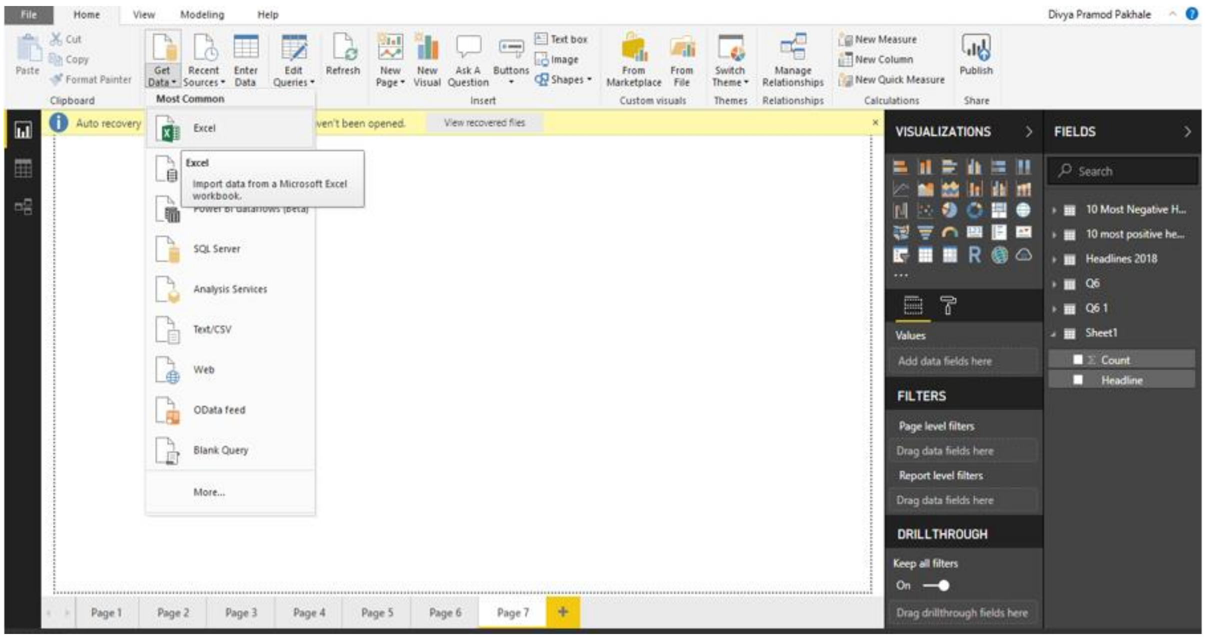
For Example, in your downloads folder

VISUALIZATION OF DATA

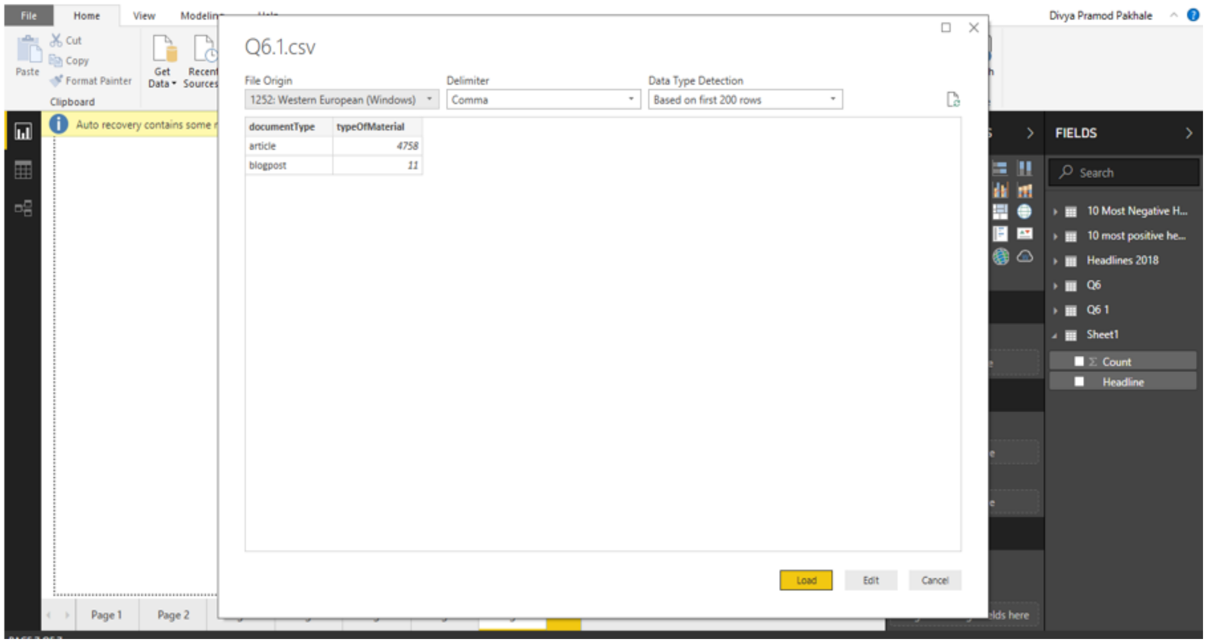
Visualizing Data: (In order to visualize the data, we have used tableau, power BI as well as Excel 3D Maps)

Query 1: Count of document type by type of material for the year 2017 and 2018.

Open the power bi tool. Click on Get data -> Excel -> Select your file (Q6.1)



Click on Load. Once the data is loaded, go to one sheet


Q6.1.csv

File Origin: 1252: Western European (Windows)

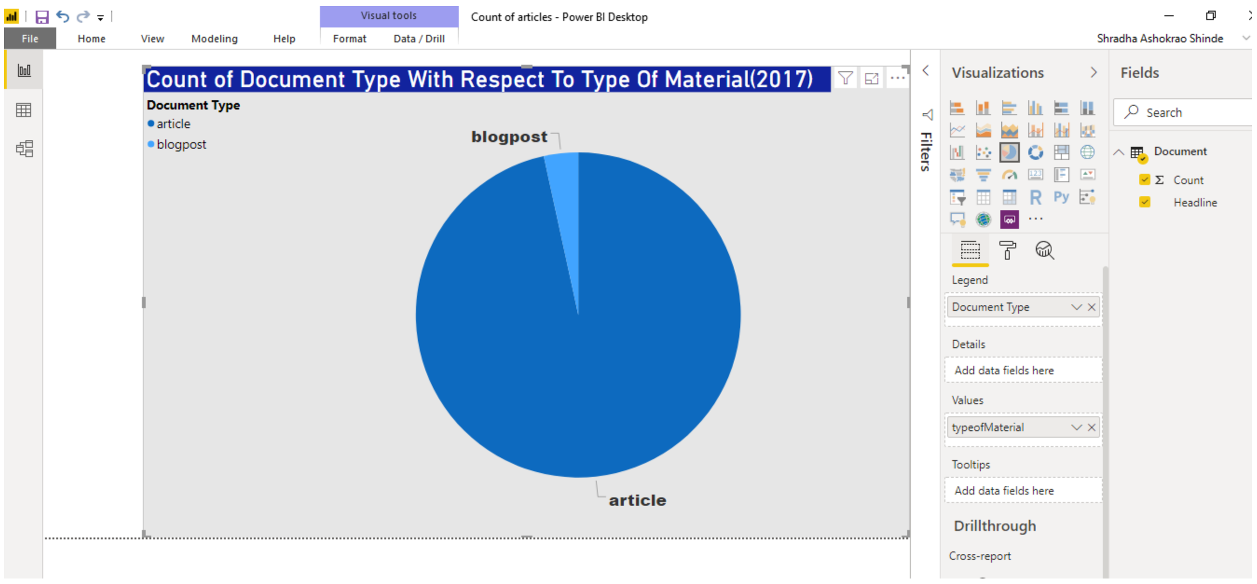
Delimiter: Comma

Data Type Detection: Based on first 200 rows

documentType	typeOfMaterial
article	4758
blogpost	11

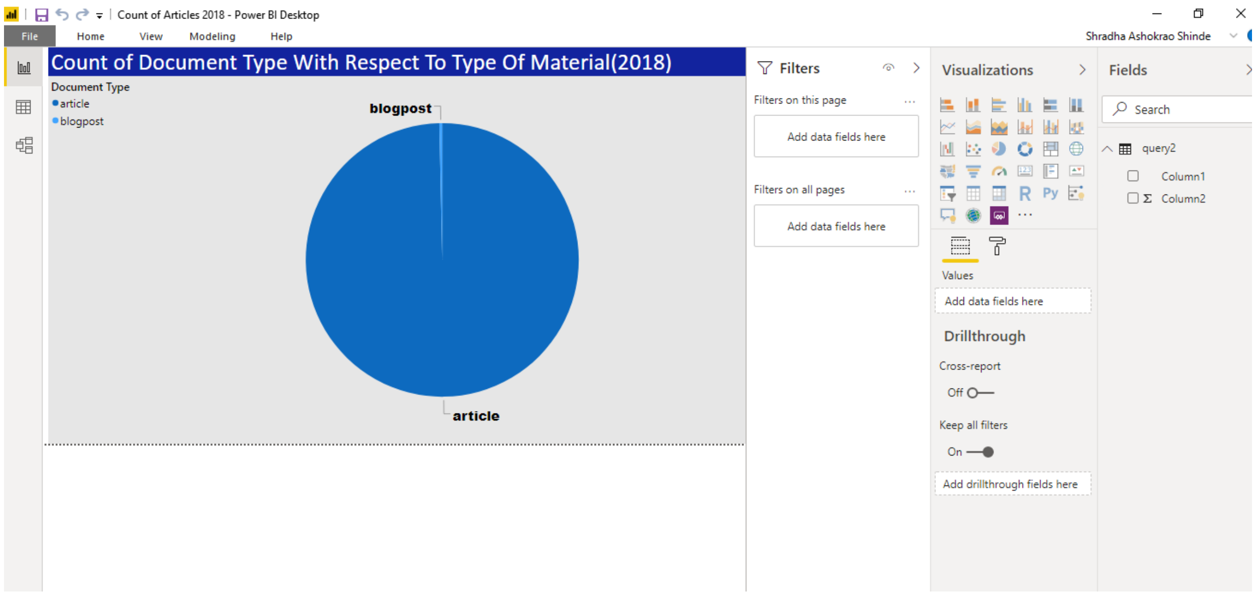
For the Year 2017

Drag document type and type of material as shown in the picture below and select pie chart visualization from the visualization field

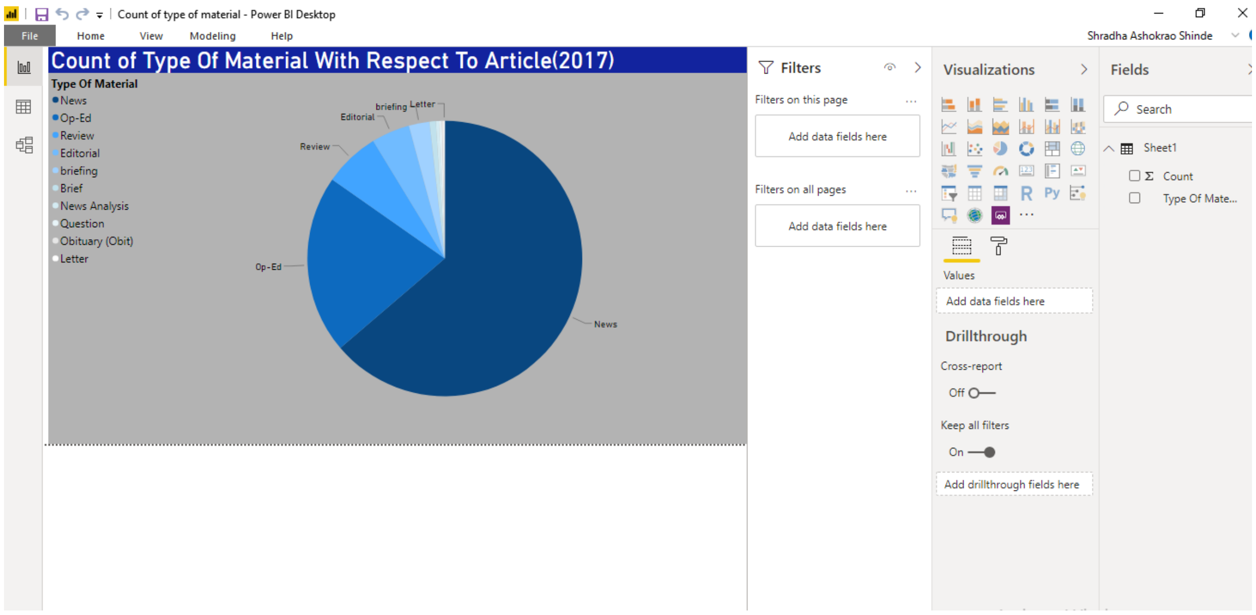


For the year 2018

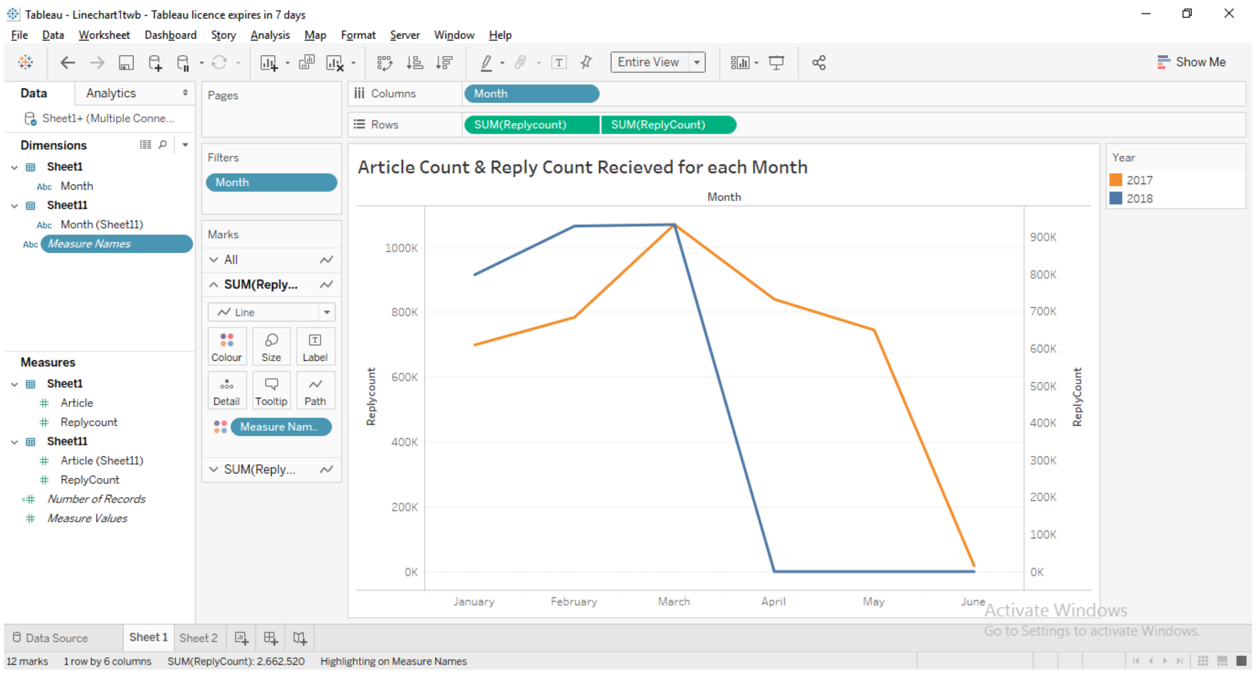
Drag document type and type of material as shown in the picture below and select pie chart visualization from the visualization field

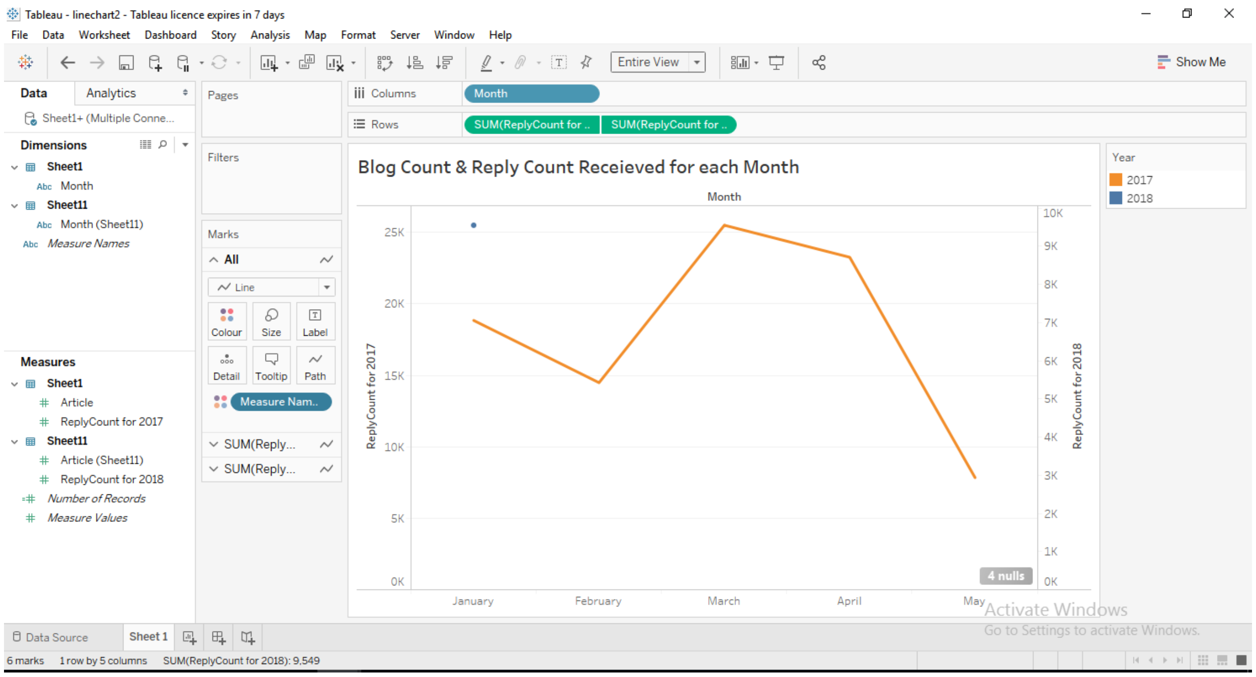


Follow same steps for type of material visualization:



Query 2: What is the reply count for the document type month wise for Year 2017 & 2018?





Above visual shows Line Chart. Open Tableau Tool -> Upload the output file -> Drag month from dimensions to rows and reply count from dimensions to rows.

X Axis represents Month from January to May. Y Axis represents reply cunt for 2017 (Left) and 2018 (Right). Orange color represents year 2017 and blue color shows trend in year 2018

Query 3: What is the reply count for each comment type.

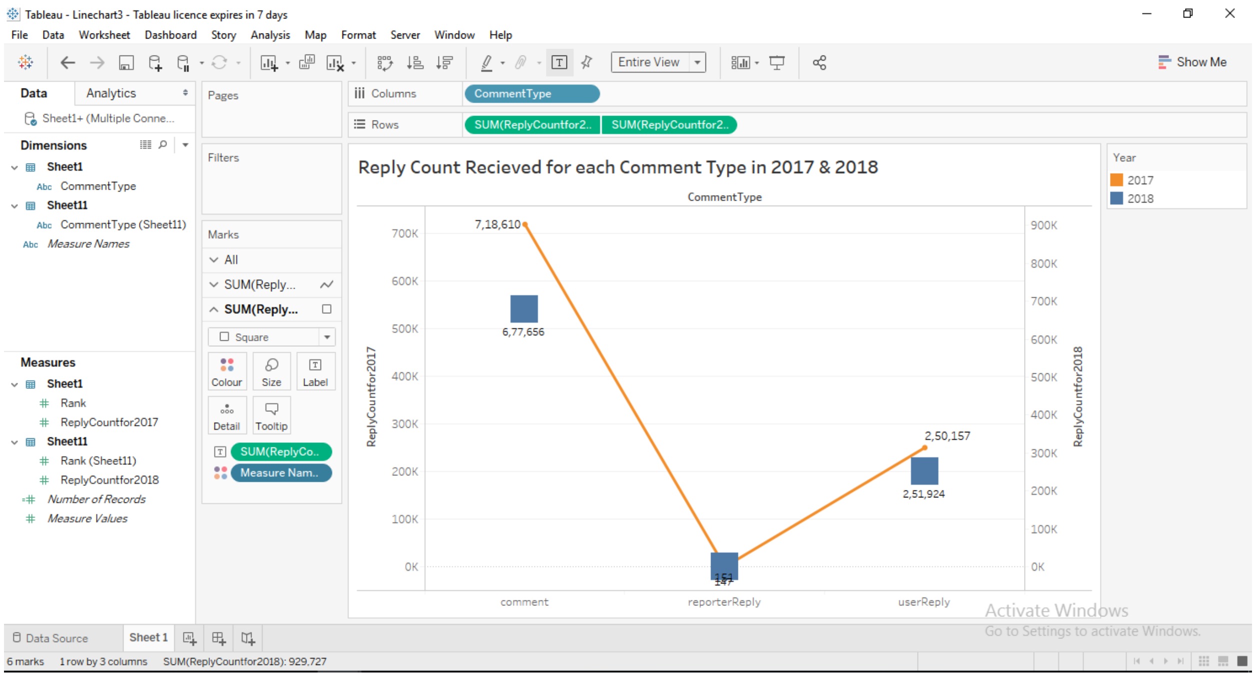


Tableau - Linechart3 - Tableau licence expires in 7 days
Columns: CommentType
Rows: SUM(ReplyCountfor2.. SUM(ReplyCountfor2..
Reply Count Recieved for each Comment Type in 2017 & 2018
CommentType
7,18,610
6,77,656
147
151
2,50,157
2,51,924
comment
reporterReply
userReply
ReplyCountfor2017
ReplyCountfor2018
Year
🟧 2017
🟦 2018
6 marks 1 row by 3 columns SUM(ReplyCountfor2018): 929,727

Above visual shows comparative analysis using dual axis. Open Tableau Tool -> Upload the output file -> Drag comment type from dimensions to rows and reply count for 2017 and 2018 from dimensions to rows.

X Axis represents comment type. Y Axis represents reply count for 2017 (Left) and 2018 (Right). Blue color represents year 2017 and grey color shows trend in year 2018.

Query 4: What is the count of new desk month wise.

Open the output file from “query14” folder in Tableau. After loading the data in Tableau, we will click of sheet 1 as shown in the picture below to create our visualization.

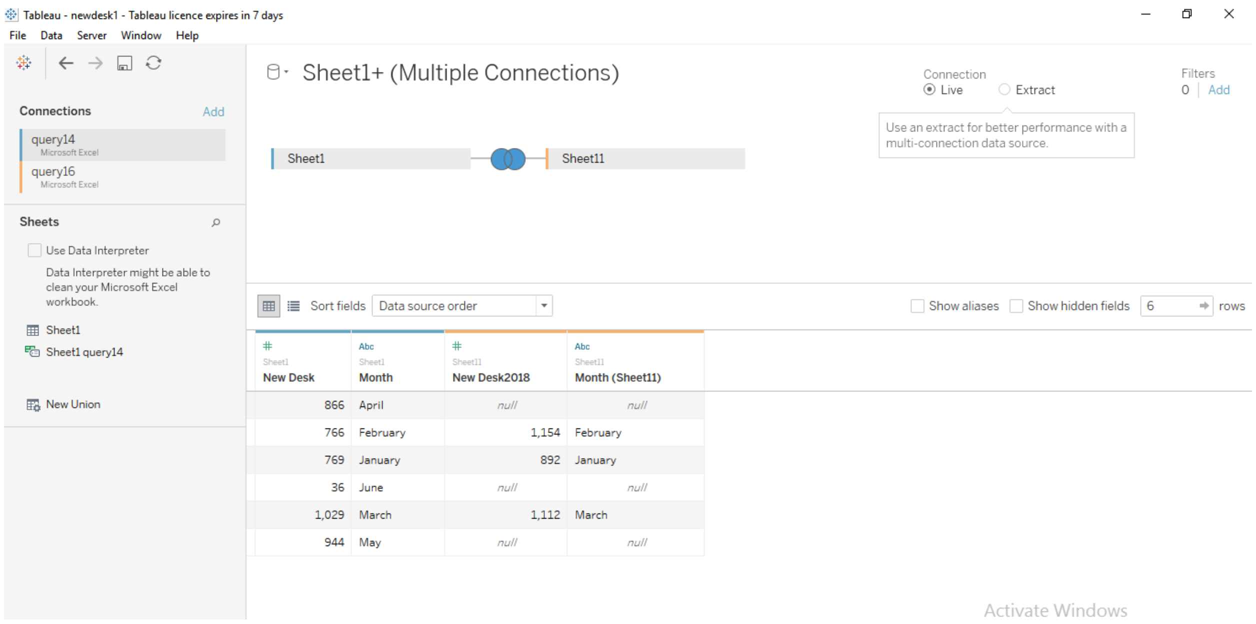
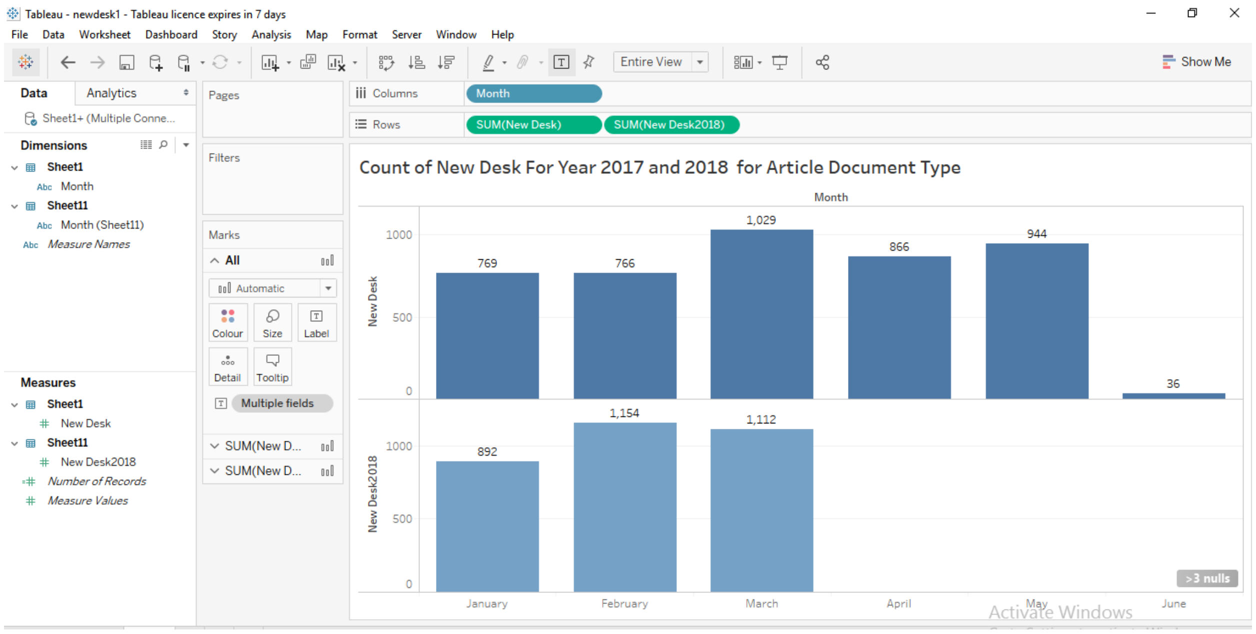


Tableau - newdesk1 - Tableau licence expires in 7 days
Sheet1+ (Multiple Connections)
Connections
query14
query16
Sheets
Sheet1
Sheet1 query14
New Union
Sheet1
Sheet11
Connection ● Live ○ Extract
Use an extract for better performance with a multi-connection data source.

New Desk	Month	New Desk2018	Month (Sheet11)
866	April	null	null
766	February	1,154	February
769	January	892	January
36	June	null	null
1,029	March	1,112	March
944	May	null	null

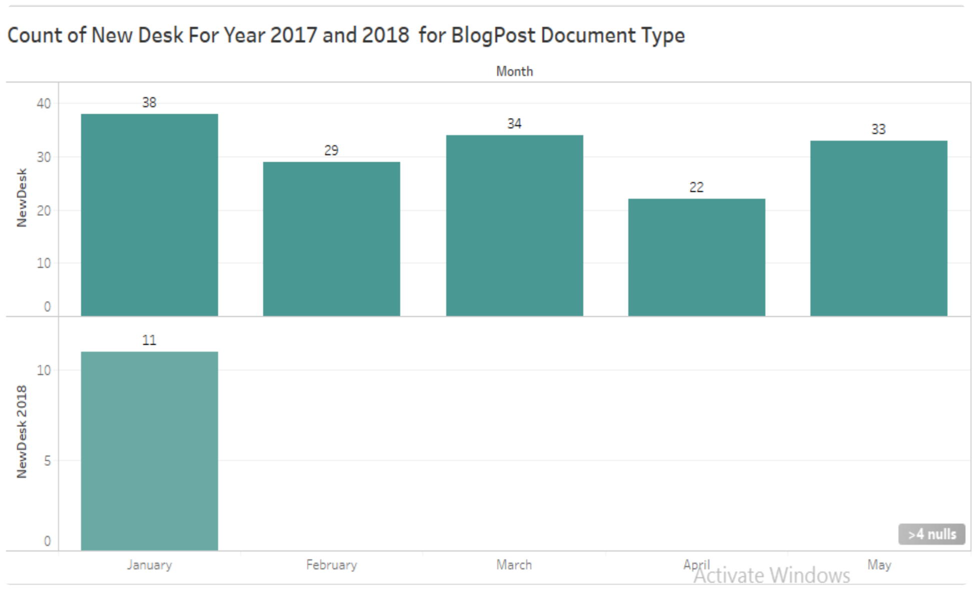
After clicking on sheet 1 worksheet, we drag the month name from dimensions to the rows field and number of desks filed from measures to the columns field.

For the year article : 2017 & 2018 year



After clicking on sheet 2 worksheet, we drag the month name from dimensions to the rows field and number of desks filed from measures to the columns field.

For the Blog post year 2017 & 2018

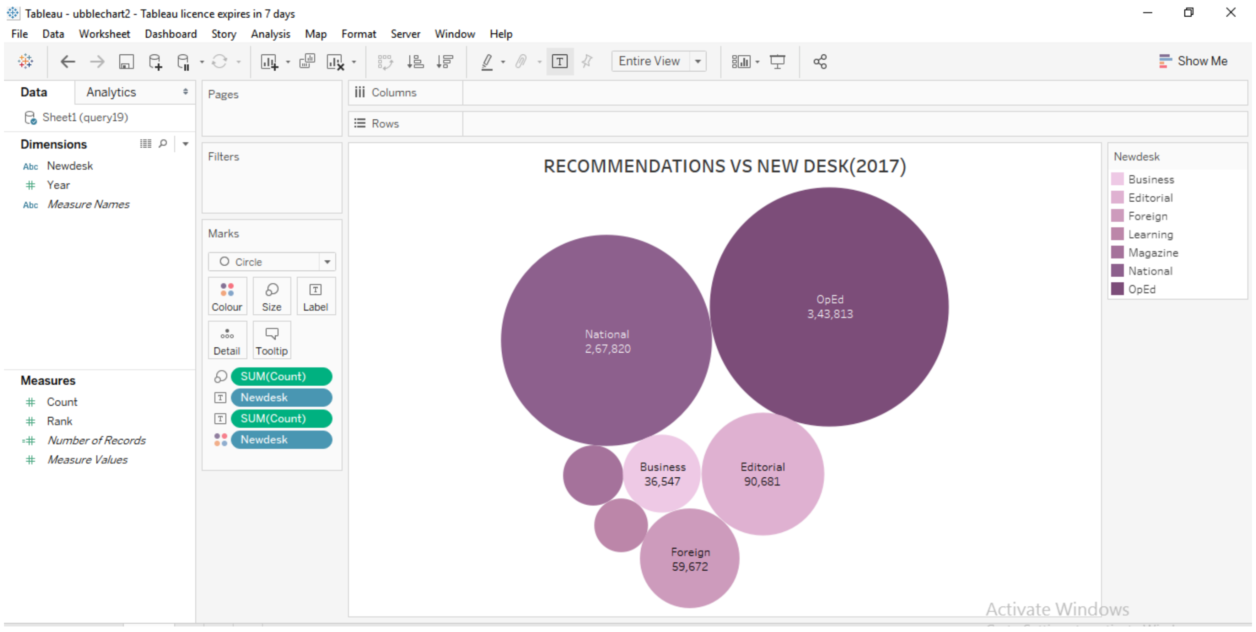


Query 5: What is the count of new desk based on recommendations.

Open the output file from "q5.1(2017)" folder in Tableau. After loading the data in Tableau, as shown in the picture below to create our visualization.

Drag New Desk from the dimensions field into rows and drag recommendations from measures into text in the marks field and then clicked on packed bubbles from the show me tab to get the visualization.

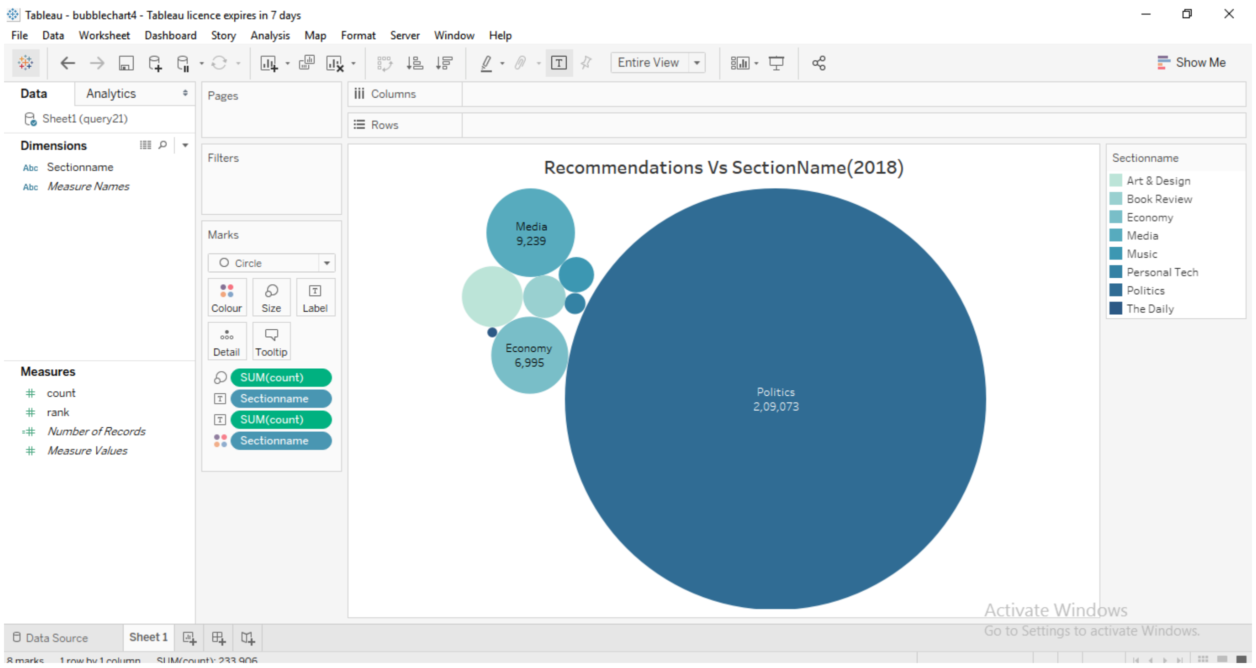
For the year 2017



For the year 2018

Open the output file from "q5.1(2018)" folder in Tableau. After loading the data in Tableau, as shown in the picture below to create our visualization.

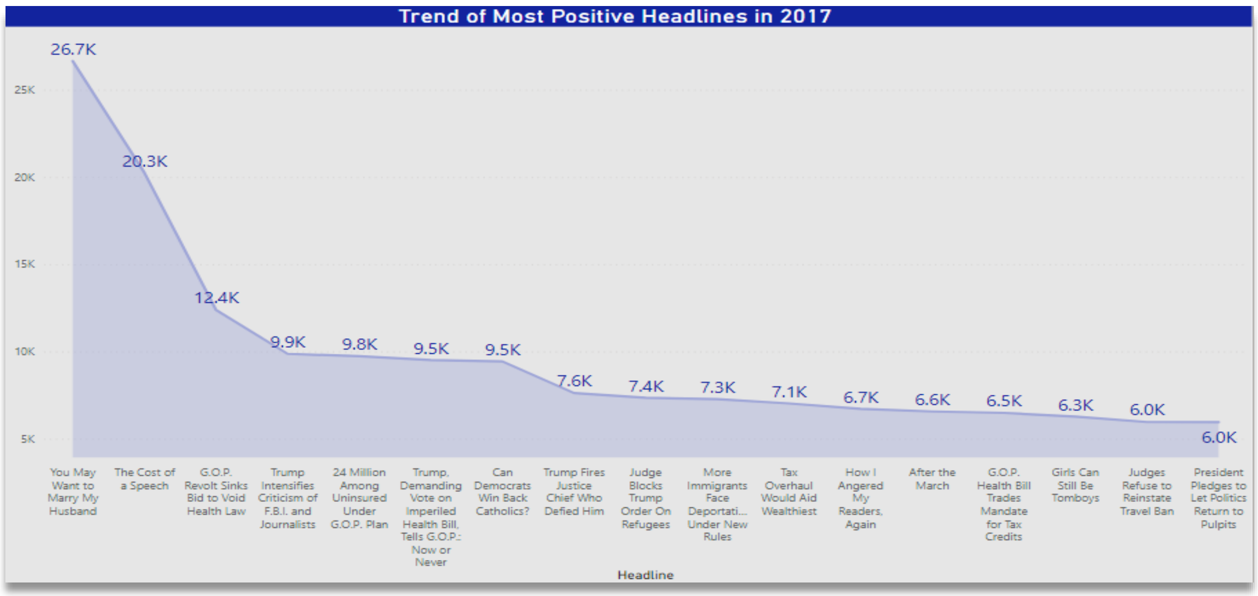
Drag New Desk from the dimensions field into rows and drag recommendations from measures into text in the marks field and then clicked on packed bubbles from the show me tab to get the visualization.



Query 6: What is the degree of polarity by most positive headlines.

Open the power bi tool. Click on Get data -> Excel -> Select your file (Ten most positive headlines)

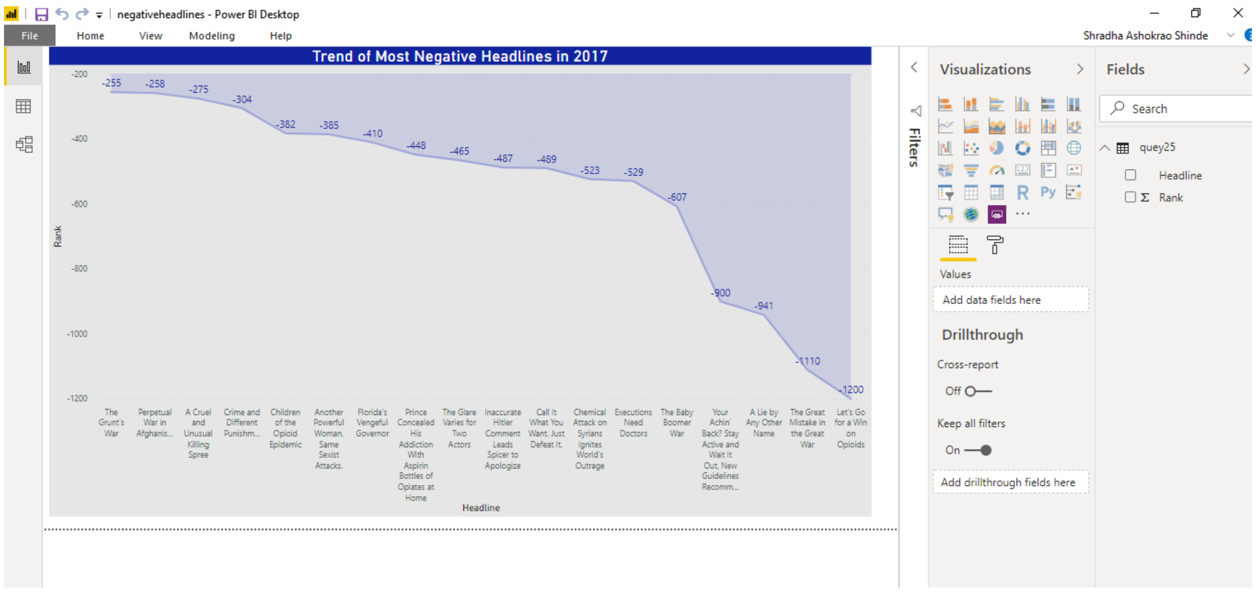
Drag column 1- Degree of Polarity and column 2 - 10 Most Positive Headlines based on the public comments as shown in the picture below and select area chart visualization from the visualization field.



Query 7: What is the degree of polarity by most negative headlines.

Open the power bi tool. Click on Get data -> Excel -> Select your file (Ten most negative headlines)

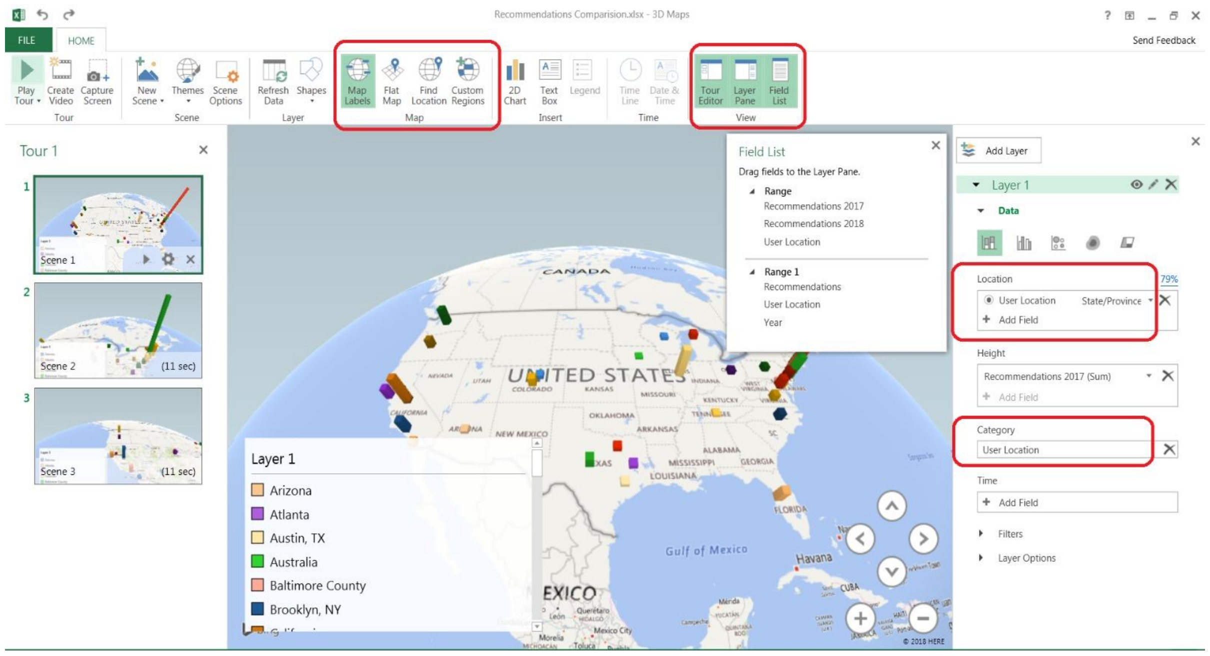
Drag most negative headlines and degree of polarity columns based on the public comments as shown in the picture below and select area chart visualization from the visualization field.


negativeheadlines - Power BI Desktop
File
Home
View
Modeling
Help
Shradha Ashokrao Shinde
Trend of Most Negative Headlines in 2017
Rank
-200
-400
-600
-800
-1000
-1200
-255
-258
-275
-304
-382
-385
-410
-448
-465
-487
-489
-523
-529
-607
-900
-941
-1110
-1200
The Grunt's War
Perpetual War in Afghanis...
A Cruel and Unusual Killing Spree
Crime and Different Punishm...
Children of the Opioid Epidemic
Another Powerful Woman. Same Sexist Attacks.
Florida's Vengeful Governor
Prince Concealed His Addiction With Aspirin Bottles of Opiates at Home
The Glare Varies for Two Actors
Inaccurate Hitler Comment Leads Spicer to Apologize
Call It What You Want. Just Defeat It.
Chemical Attack on Syrians Ignites World's Outrage
Executions Need Doctors
The Baby Boomer War
Your Achin' Back? Stay Active and Wait It Out, New Guidelines Recomm...
A Lie by Any Other Name
The Great Mistake in the Great War
Let's Go for a Win on Opioids
Headline
Filters
Visualizations
Values
Add data fields here
Drillthrough
Cross-report
Off
Keep all filters
On
Add drillthrough fields here
Fields
Search
quey25
Headline
Rank

Query 10: What are the recommendations by the user's location for the year 2017.

To create the 3D-Map, follow these steps:

1. Load the output file for the respective query (which had been extracted from Hive) into Microsoft Excel.
2. Select the data, including the column headers in the table format.
3. Click Insert | 3D Maps | Open 3D Maps.
4. Drag fields (column header names) to the Layer panel as shown in the screenshot.
5. Reveal the 3D-Map.



Field List

Drag fields to the Layer Pane.

- Range
  - Recommendations 2017
  - Recommendations 2018
  - User Location
- Range 1
  - Recommendations
  - User Location
  - Year

Layer 1

- 🟧 Arizona
- 🟪 Atlanta
- 🟨 Austin, TX
- 🟩 Australia
- 🟥 Baltimore County
- 🟦 Brooklyn, NY

Layer 1

Data

Location 79%
User Location State/Province
+ Add Field

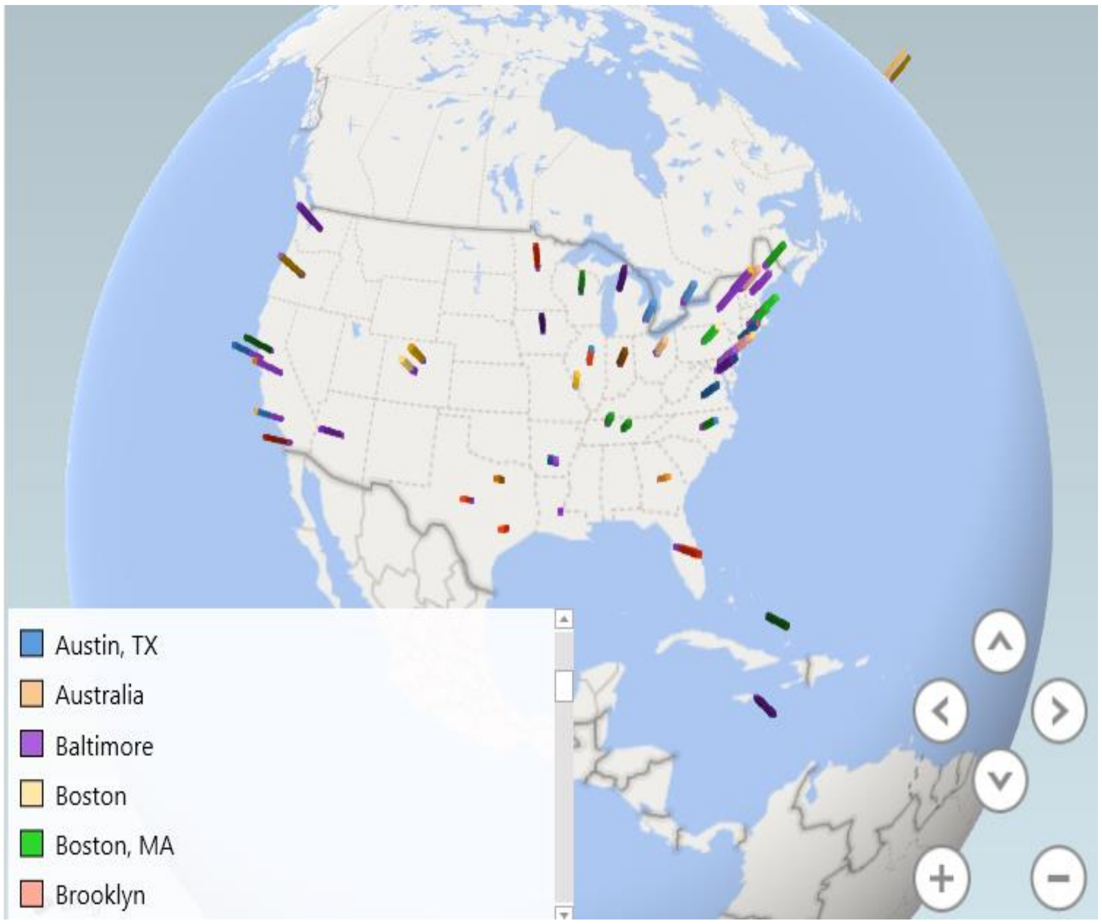
Height
Recommendations 2017 (Sum)
+ Add Field

Category
User Location

Time
+ Add Field

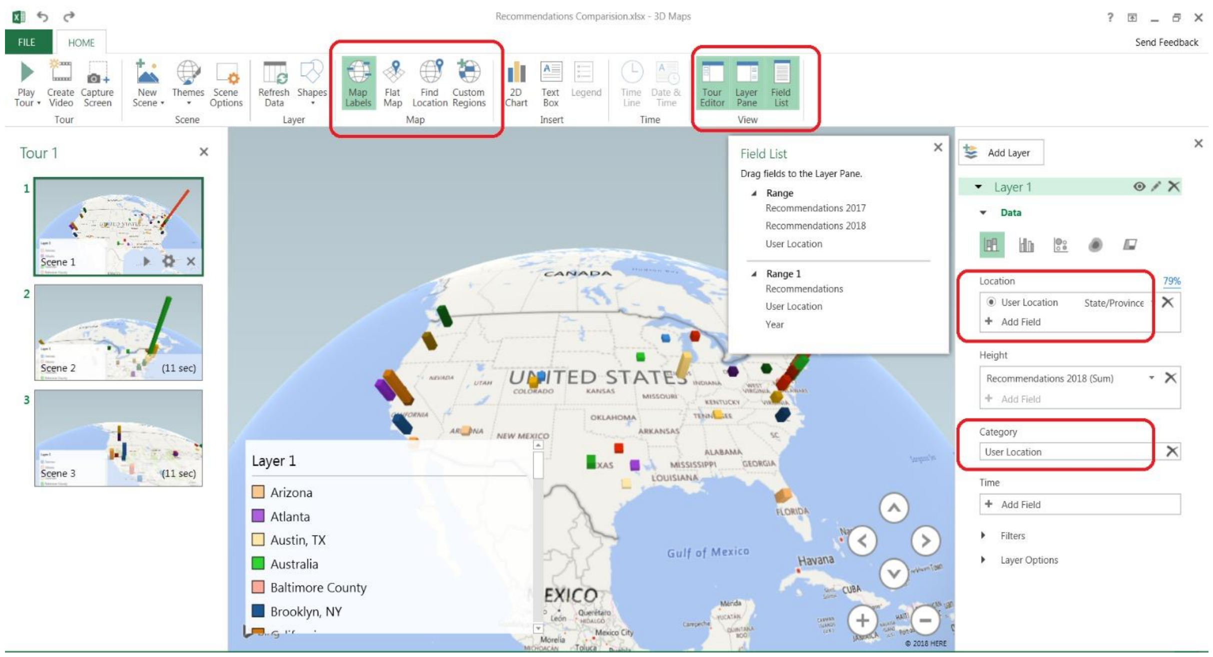
Filters

Layer Options



Query 11: What are the recommendations by the user's location for the year 2018.

(Follow the same steps as mentioned above)



Field List

Drag fields to the Layer Pane.

- Range
  - Recommendations 2017
  - Recommendations 2018
  - User Location
- Range 1
  - Recommendations
  - User Location
  - Year

Add Layer

Layer 1

Data

Location: User Location — State/Province (79%)

Height: Recommendations 2018 (Sum)

Category: User Location

Time: Add Field

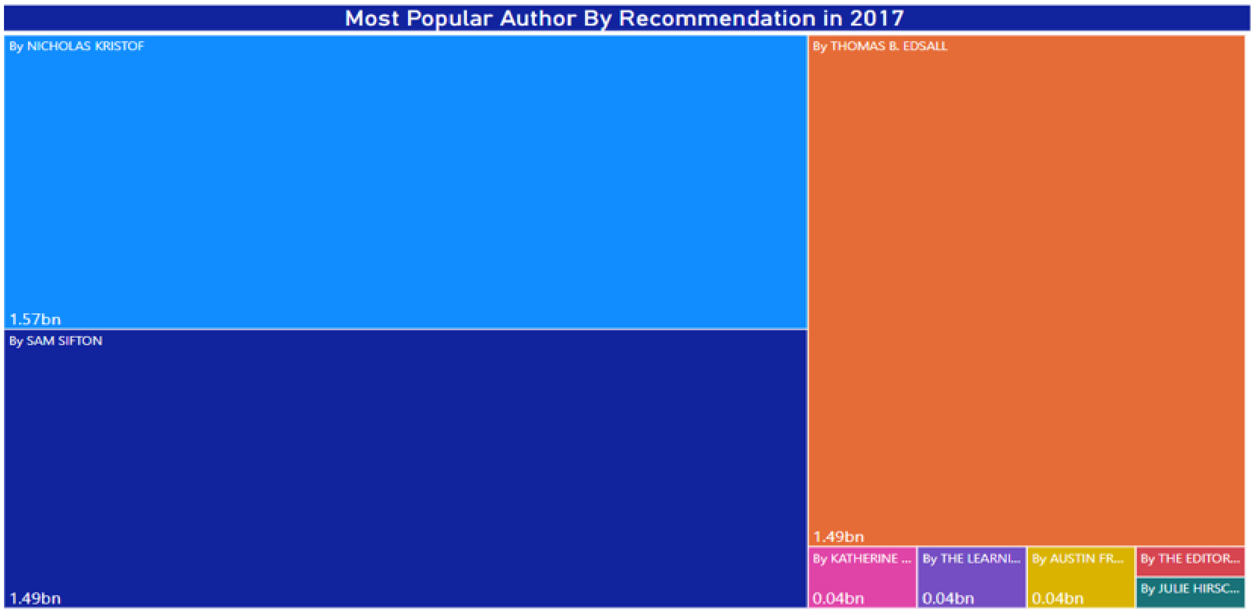
Filters

Layer Options

Query 12: Most Popular Author(byline) with respect to recommendations of public for the year 2017?

Open the power bi tool. Click on Get data -> Excel -> Select your file (byline2017.csv)

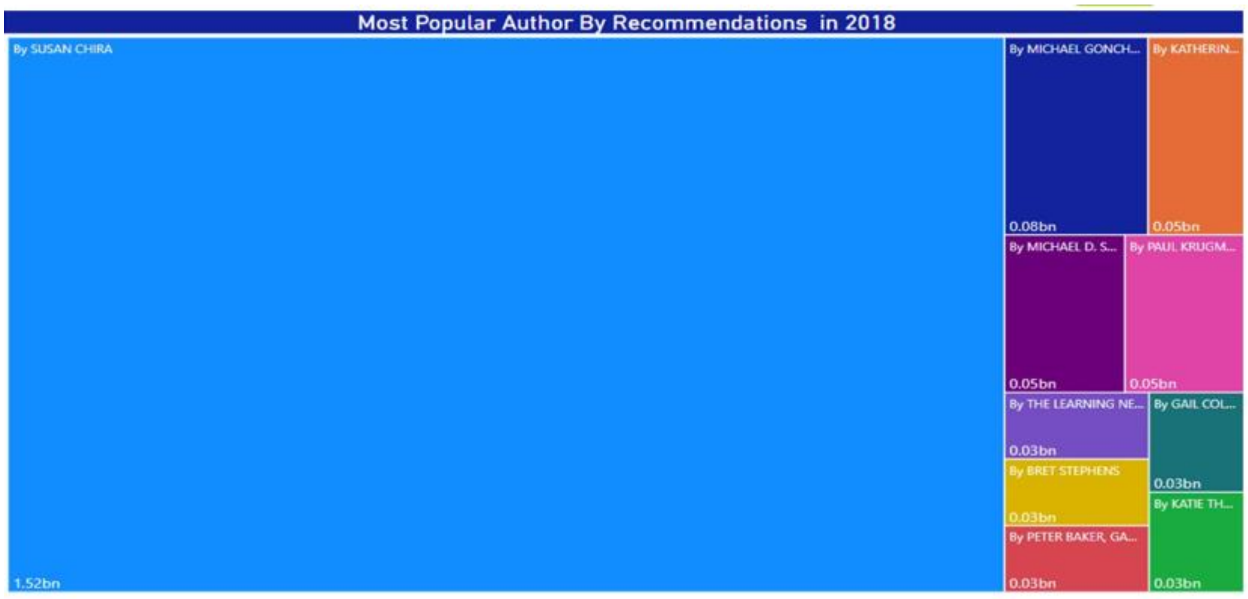
Drag Author and Recommendations column as shown in the picture below and select tree map chart visualization from the visualization field.



Query 13: Most Popular Author(byline) with respect to recommendations of public for the year 2018?

Open the power bi tool. Click on Get data -> Excel -> Select your file (byline2018.csv)

Drag Author and Recommendations column as shown in the picture below and select tree map chart visualization from the visualization field.



REFERENCES AND GITHUB LINK

1. https://github.com/shradhacsula/5200-project
2. https://towardsdatascience.com/predicting-popularity-of-the-new-york-times-comments-part-1-d32f26261f6f
3. https://www.kaggle.com/aashita/exploratory-data-analysis-of-comments-on-nyt/notebook

This is the end of the lab