

Netflix Data Analytics



*Date:-
04-03-2023*

*Prepared By:-
Darshankumar Patel*

Dataset Information: -

The dataset contains following information: -

- 1.) show_id :- Unique id for TV shows or Movies.
- 2.) type: - TV show or Movie.
- 3.) title:- title of TV show or Movie.
- 4.) director:- person who makes the TV show or Movie.
- 5.) actor(cast):- persons worked in TV show or Movie.
- 6.) country: - TV show or Movie released in.
- 7.) date_added: - date on which TV show or Movie is added on Netflix.
- 8.) release_year:- year on which TV show or Movie is released on Netflix.
- 9.) rating :-rating on TV show or Movie.
- 10.) duration:- how longest is TV show or Movie.
- 11.) listed_in:-type of TV show or Movie.
- 12.) description:- short overview of TV show or Movie.

Problem Statement :-

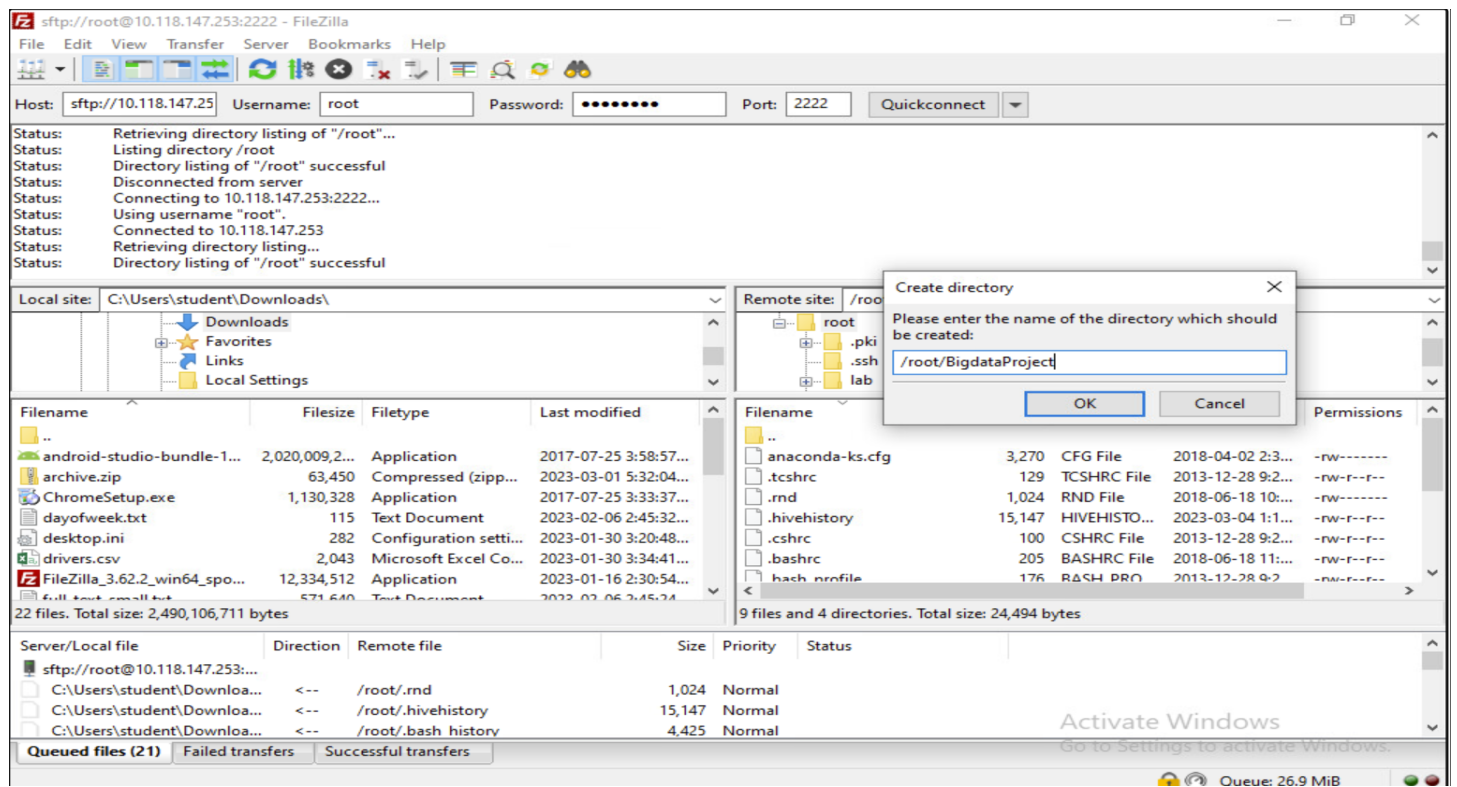
With so much content available on Netflix, users may find it challenging to discover new movies and shows that align with their interests. While Netflix has a vast database of content, the quality of the movies and shows may not be consistent across the platform. Netflix recommends content to users based on their viewing history. However, these recommendations may not always be accurate, leading to user dissatisfaction.

Some of Analysis we performed: -

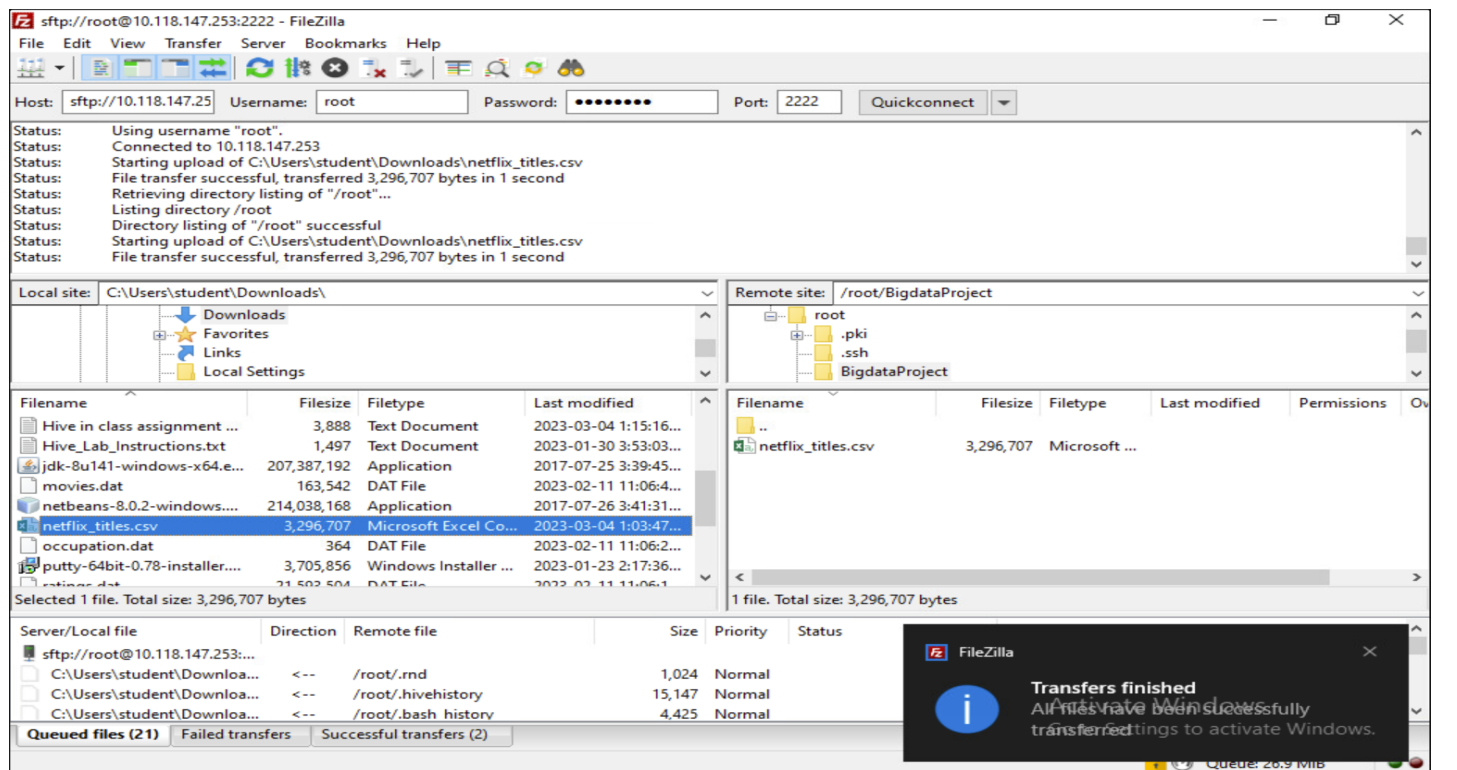
- 1.) To analysis which kind of show or movies is famous in Netflix dataset.
- 2.) To analysis which country is suitable for produce movies and TV shows.
- 3.) To analysis which actors have worked in more movies and TV shows.
- 4.) To analysis which rating movies or TV shows are more watched by audience.

And many more....

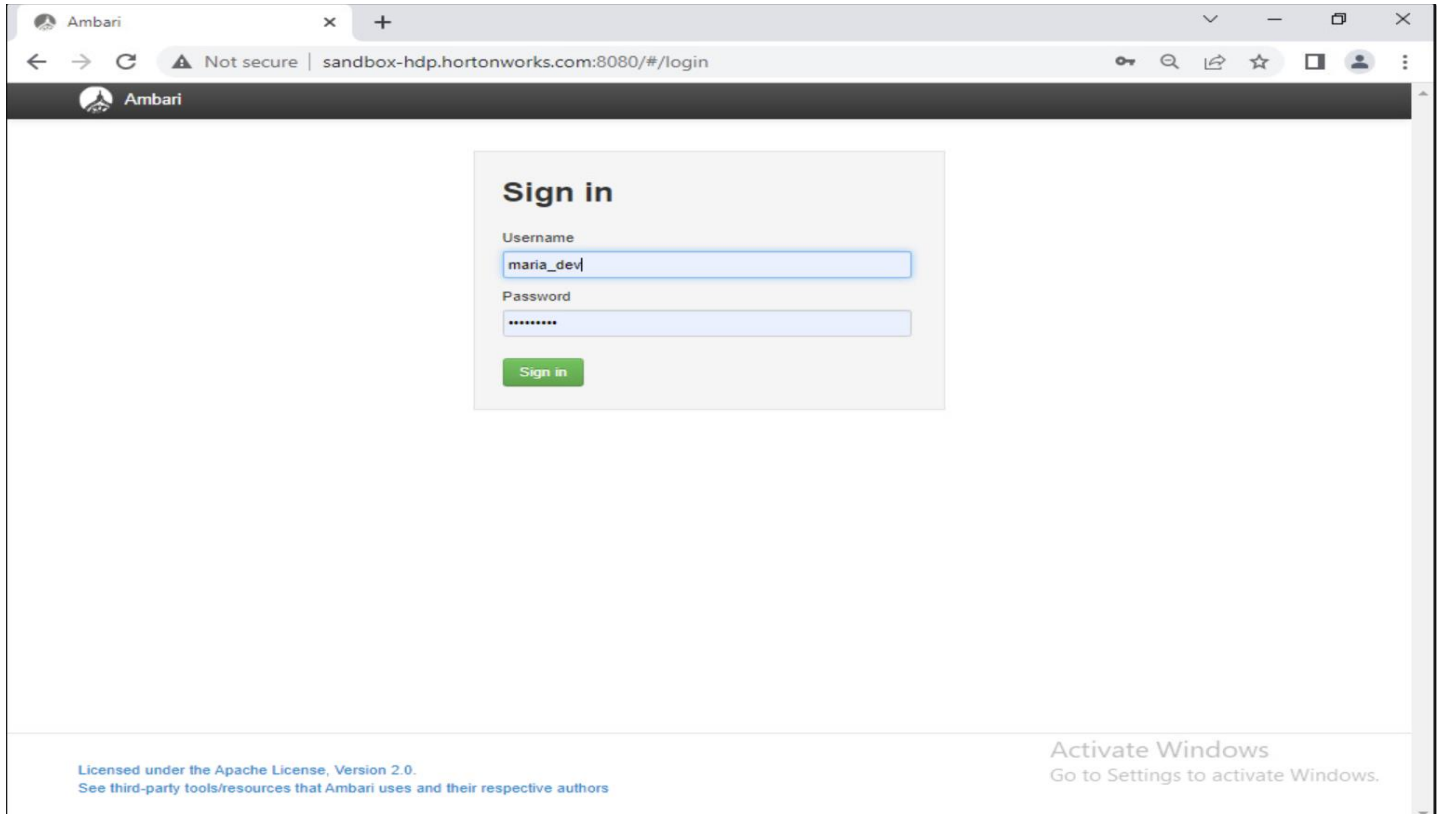
Step -1 – Login into FileZilla using the VMWare SSH and create a remote directory inside /root folder and named it BigdataProject.



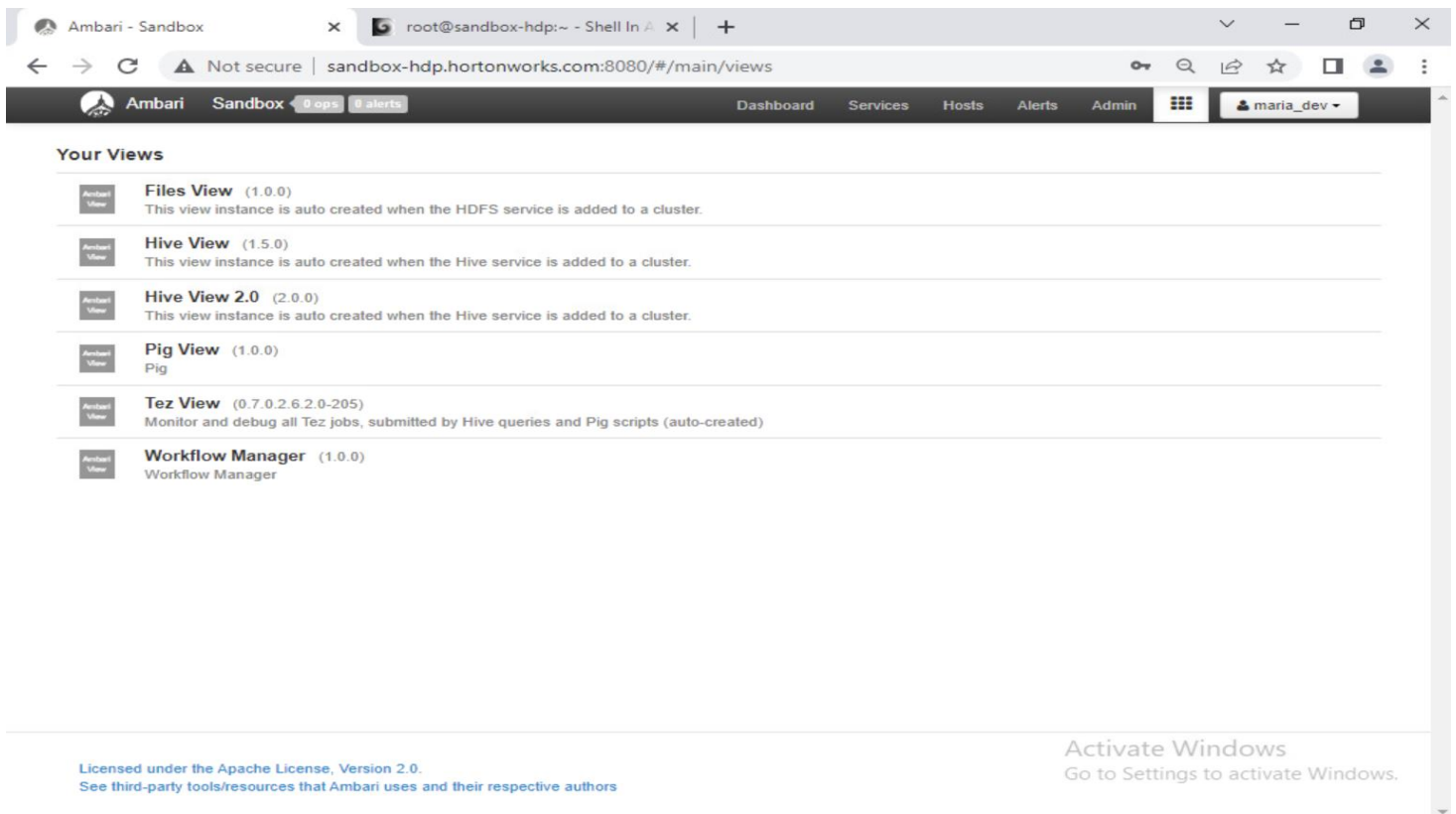
Step-2 -Now upload the dataset into remote directory(BigdataProject) just drag & drop from the downloads section folder seen on left-side(local site).



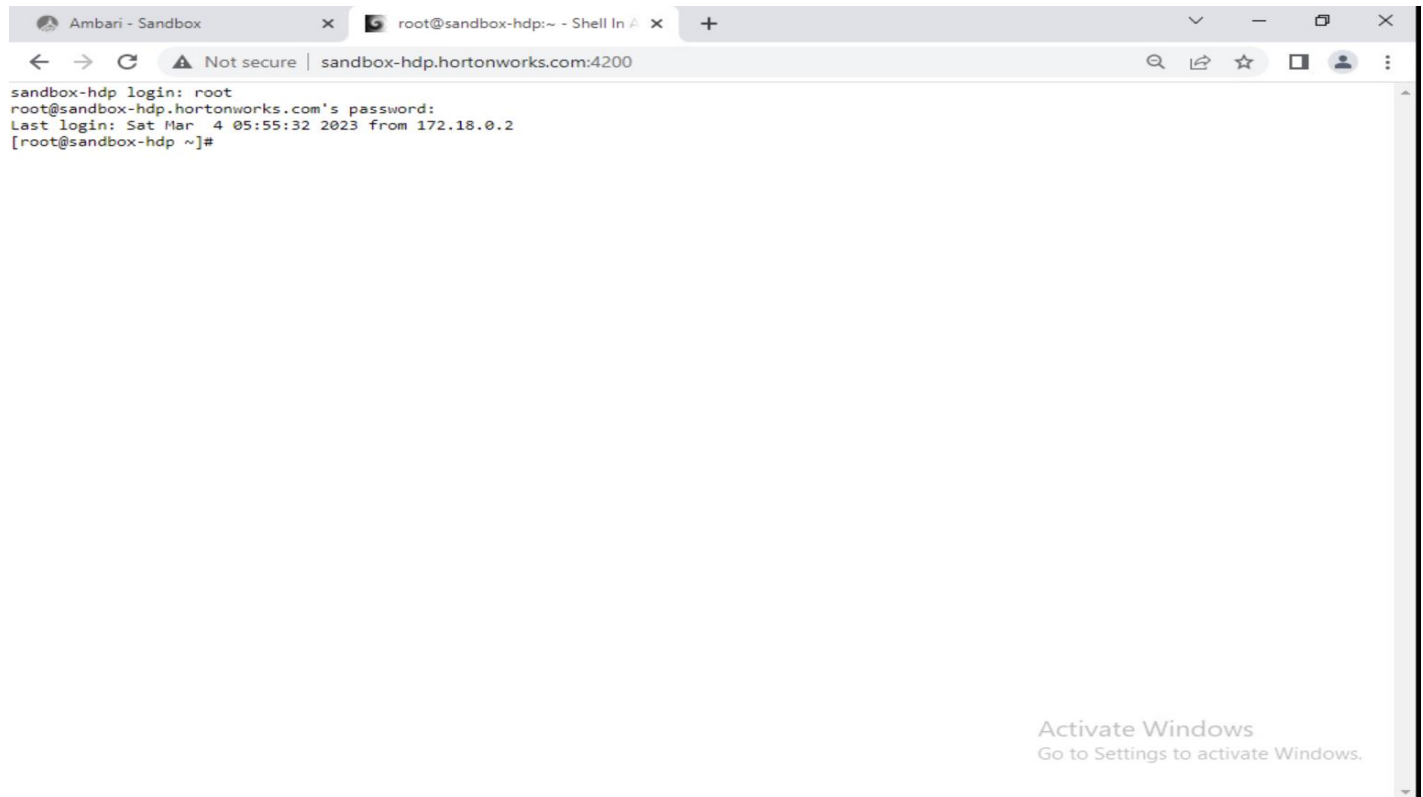
Step-3 – Now, Login into Ambari server using the Login credentials or other option is to use putty. Here, I am going to use Ambari server.



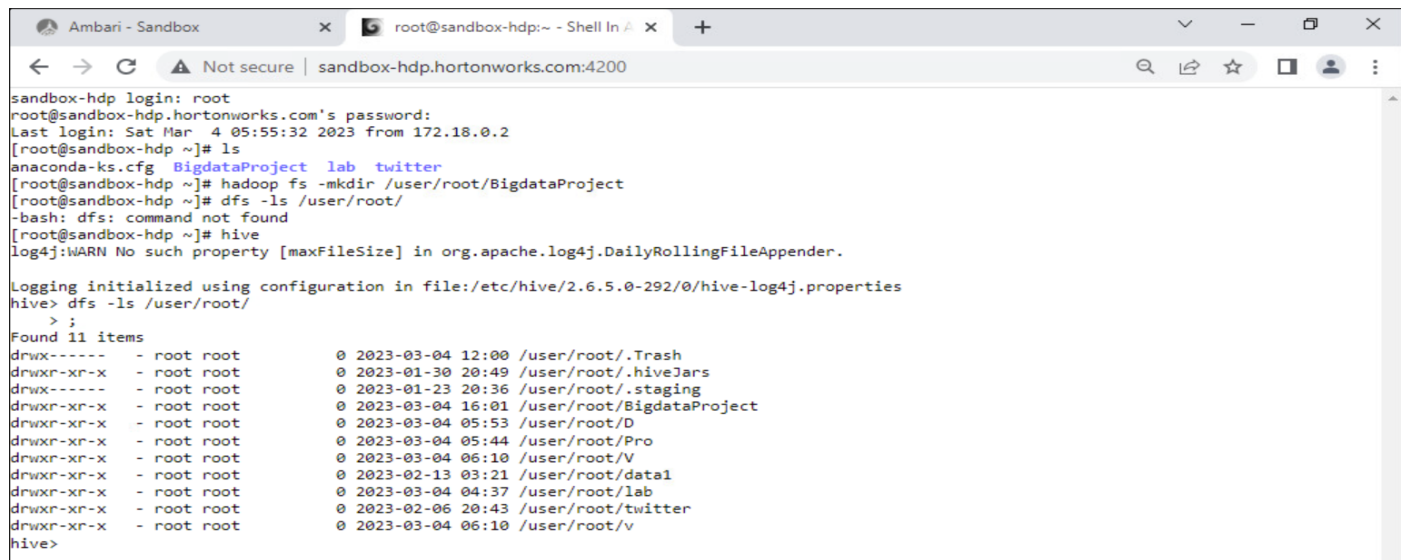
The Image shown Below is Ambari Dashboard.



Step-4- Now connect with HDP sandbox using the link <http://sandbox-hdp.hortonworks.com:4200> and enter Login Credentials.

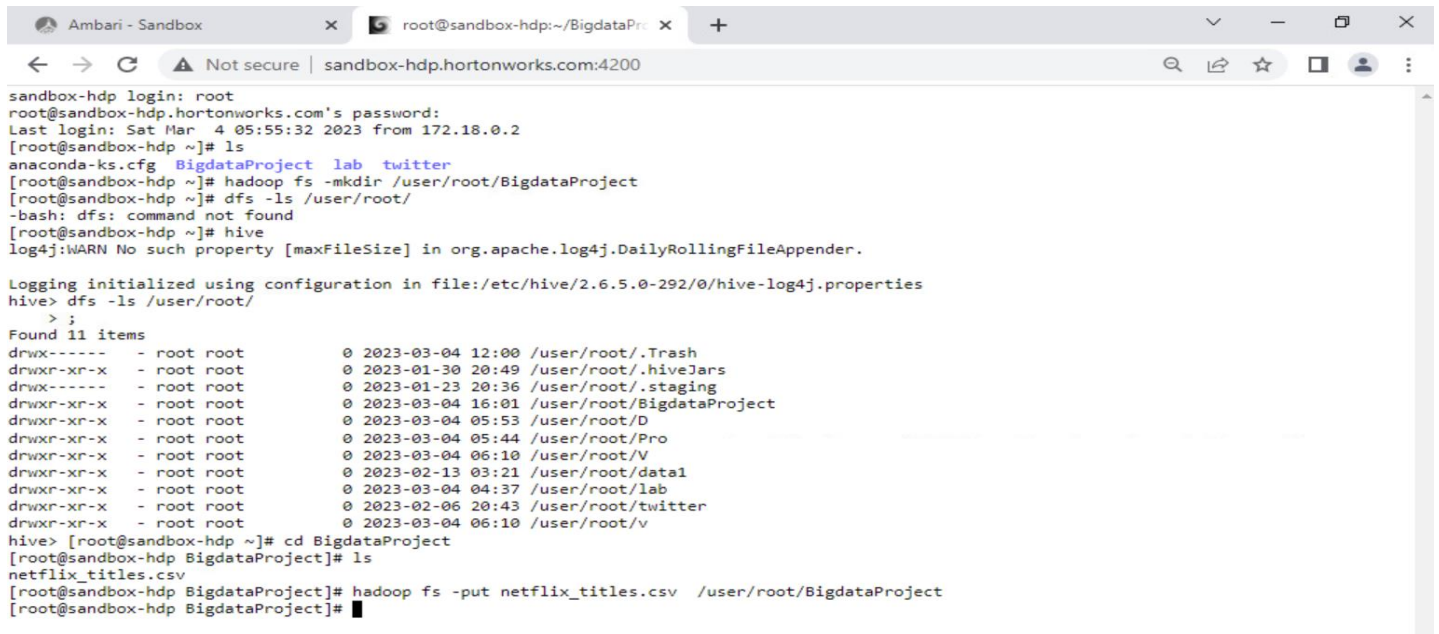


Step -5 – Now we will create a new directory in HDFS using the Hadoop Commands.



hadoop fs -mkdir /user/root/BigdataProject is used to create new directory in HDFS and then after to check it we can start hive and type **dfs -ls /user/root/** to see whether directory is created or not.(as this command list the directory inside the path.)

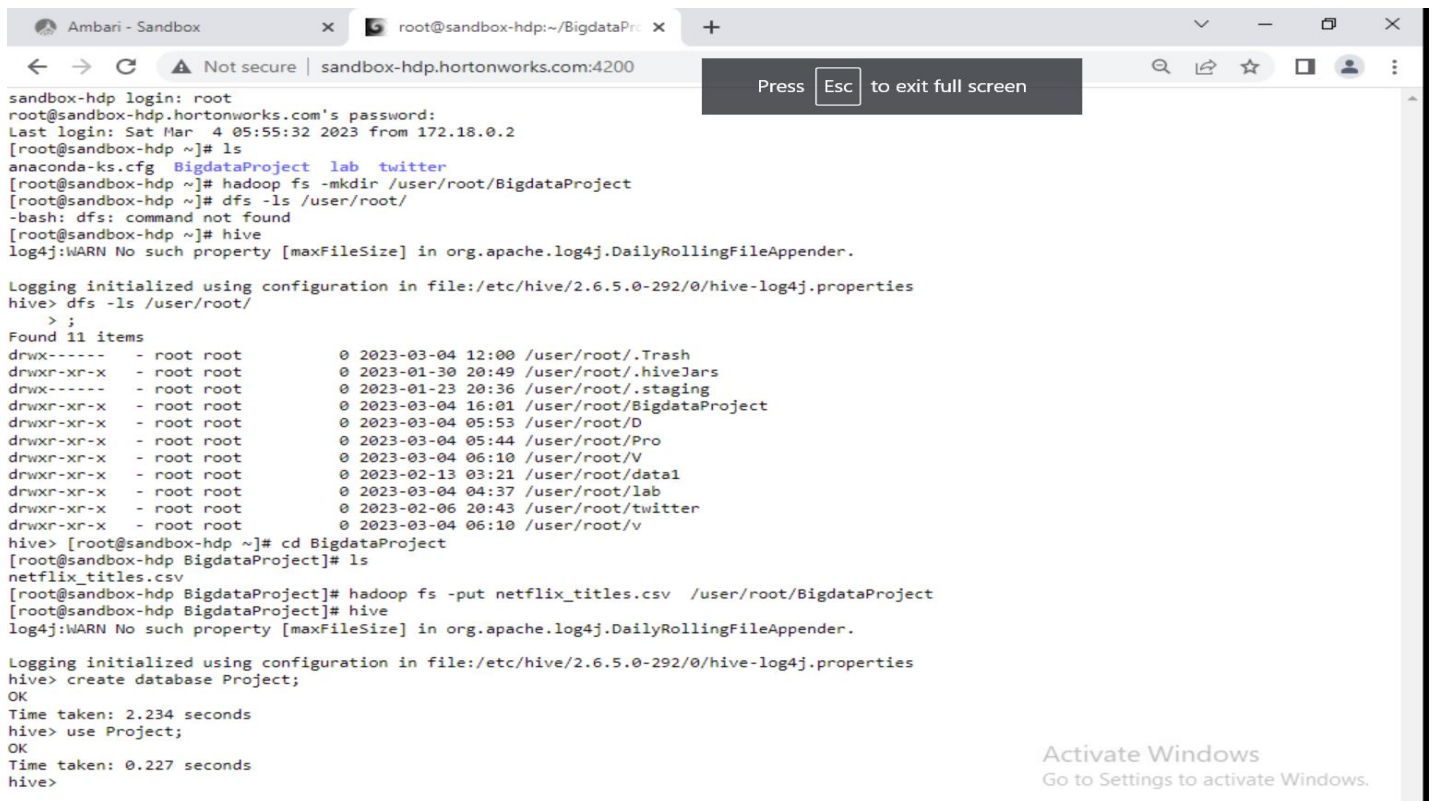
Step – 6- Now, we exit the hive using CTRL + C because I need to copy my dataset files to directory using the Hadoop command. Also, I went inside my directory where the dataset file i.e., netflix_titles.csv was there.



```
sandbox-hdp login: root
root@sandbox-hdp.hortonworks.com's password:
Last login: Sat Mar 4 05:55:32 2023 from 172.18.0.2
[root@sandbox-hdp ~]# ls
anaconda-ks.cfg BigdataProject lab twitter
[root@sandbox-hdp ~]# hadoop fs -mkdir /user/root/BigdataProject
[root@sandbox-hdp ~]# dfs -ls /user/root/
-bash: dfs: command not found
[root@sandbox-hdp ~]# hive
log4j:WARN No such property [maxFileSize] in org.apache.log4j.DailyRollingFileAppender.

Logging initialized using configuration in file:/etc/hive/2.6.5.0-292/0/hive-log4j.properties
hive> dfs -ls /user/root/
>
Found 11 items
drwx----- - root root      0 2023-03-04 12:00 /user/root/.Trash
drwxr-xr-x - root root      0 2023-01-30 20:49 /user/root/.hiveJars
drwx----- - root root      0 2023-01-23 20:36 /user/root/.staging
drwxr-xr-x - root root      0 2023-03-04 16:01 /user/root/BigdataProject
drwxr-xr-x - root root      0 2023-03-04 05:53 /user/root/D
drwxr-xr-x - root root      0 2023-03-04 05:44 /user/root/Pro
drwxr-xr-x - root root      0 2023-03-04 06:10 /user/root/V
drwxr-xr-x - root root      0 2023-02-13 03:21 /user/root/data1
drwxr-xr-x - root root      0 2023-03-04 04:37 /user/root/lab
drwxr-xr-x - root root      0 2023-02-06 20:43 /user/root/twitter
drwxr-xr-x - root root      0 2023-03-04 06:10 /user/root/v
hive> [root@sandbox-hdp ~]# cd BigdataProject
[root@sandbox-hdp BigdataProject]# ls
netflix_titles.csv
[root@sandbox-hdp BigdataProject]# hadoop fs -put netflix_titles.csv /user/root/BigdataProject
[root@sandbox-hdp BigdataProject]#
```

Step-7- Now, almost we are done with hdfs now it's time to start hive again and create new database for my project.



```
sandbox-hdp login: root
root@sandbox-hdp.hortonworks.com's password:
Last login: Sat Mar 4 05:55:32 2023 from 172.18.0.2
[root@sandbox-hdp ~]# ls
anaconda-ks.cfg BigdataProject lab twitter
[root@sandbox-hdp ~]# hadoop fs -mkdir /user/root/BigdataProject
[root@sandbox-hdp ~]# dfs -ls /user/root/
-bash: dfs: command not found
[root@sandbox-hdp ~]# hive
log4j:WARN No such property [maxFileSize] in org.apache.log4j.DailyRollingFileAppender.

Logging initialized using configuration in file:/etc/hive/2.6.5.0-292/0/hive-log4j.properties
hive> dfs -ls /user/root/
>
Found 11 items
drwx----- - root root      0 2023-03-04 12:00 /user/root/.Trash
drwxr-xr-x - root root      0 2023-01-30 20:49 /user/root/.hiveJars
drwx----- - root root      0 2023-01-23 20:36 /user/root/.staging
drwxr-xr-x - root root      0 2023-03-04 16:01 /user/root/BigdataProject
drwxr-xr-x - root root      0 2023-03-04 05:53 /user/root/D
drwxr-xr-x - root root      0 2023-03-04 05:44 /user/root/Pro
drwxr-xr-x - root root      0 2023-03-04 06:10 /user/root/V
drwxr-xr-x - root root      0 2023-02-13 03:21 /user/root/data1
drwxr-xr-x - root root      0 2023-03-04 04:37 /user/root/lab
drwxr-xr-x - root root      0 2023-02-06 20:43 /user/root/twitter
drwxr-xr-x - root root      0 2023-03-04 06:10 /user/root/v
hive> [root@sandbox-hdp ~]# cd BigdataProject
[root@sandbox-hdp BigdataProject]# ls
netflix_titles.csv
[root@sandbox-hdp BigdataProject]# hadoop fs -put netflix_titles.csv /user/root/BigdataProject
[root@sandbox-hdp BigdataProject]# hive
log4j:WARN No such property [maxFileSize] in org.apache.log4j.DailyRollingFileAppender.

Logging initialized using configuration in file:/etc/hive/2.6.5.0-292/0/hive-log4j.properties
hive> create database Project;
OK
Time taken: 2.234 seconds
hive> use Project;
OK
Time taken: 0.227 seconds
hive>
```


Step-8 – Creating Empty External Tables to store my dataset information into tables and loading the data into tables(it's GUI view for Hive).

The screenshot shows the Hive GUI interface. The 'QUERY' tab is active, and a SQL query is entered in the editor:

```
1 CREATE EXTERNAL TABLE IF NOT EXISTS netflix(  
2 show_id STRING,  
3 type STRING,  
4 title STRING,  
5 director STRING,  
6 actors STRING,  
7 country STRING,  
8 date_added STRING,  
9 release_year INT,  
10 rating STRING,  
11 duration STRING,  
12 listed_in STRING,  
13 description STRING)  
14 COMMENT 'Data from NETFLIX(Kaggle)'  
15 ROW FORMAT DELIMITED  
16 FIELDS TERMINATED BY ','  
17 STORED AS TEXTFILE  
18 location '/user/root/BigdataProject/'  
19 TBLPROPERTIES("skip.header.line.count"="1");  
20
```

Below the editor, the 'Execute' button is highlighted. On the right, the 'project' database is selected, and the 'Tables(0)' section shows 'No Table found.'

The screenshot shows the Hive GUI interface after executing the query. The 'RESULTS' tab is active, and the query results are displayed in a table:

```
1 select * from netflix;
```

netflix.show_id	netflix.type	netflix.title	netflix.director	netflix.actors	netflix.country	netflix.date_added	netflix.release_year
s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	"	United States	25-Sep-21	2020
s2	TV Show	Blood & Water	"	Ama Qamata; Khosi Ngema; Gail Mabalane; Thabang	South Africa	24-Sep-21	2021

On the right, the 'project' database is selected, and the 'Tables(1)' section shows the 'netflix' table.

Step-9 – Using the Zeppelin we are analyzing the dataset. Create a project DataFrame from CSV file.

The screenshot shows the Zeppelin Notebook interface. The top bar includes the Zeppelin logo, the word "Project", and various icons for running, saving, and viewing the notebook. The main area contains a code block with the following Scala code:

```
val project= (spark.read
  .option("header","true")
  .option("inferSchema","true")
  .csv("/user/root/BigdataProject/netflix_titles.csv"))
```

Below the code, the output is displayed: "project: org.apache.spark.sql.DataFrame = [show_id: string, type: string ... 10 more fields]". The status "FINISHED" is shown in the top right corner of the code block.

Step-10 – Printing the schema in a tree format.

The screenshot shows the Zeppelin Notebook interface. The top bar includes the Zeppelin logo, the word "Notebook", and a search bar. The main area contains a code block with the following Scala code:

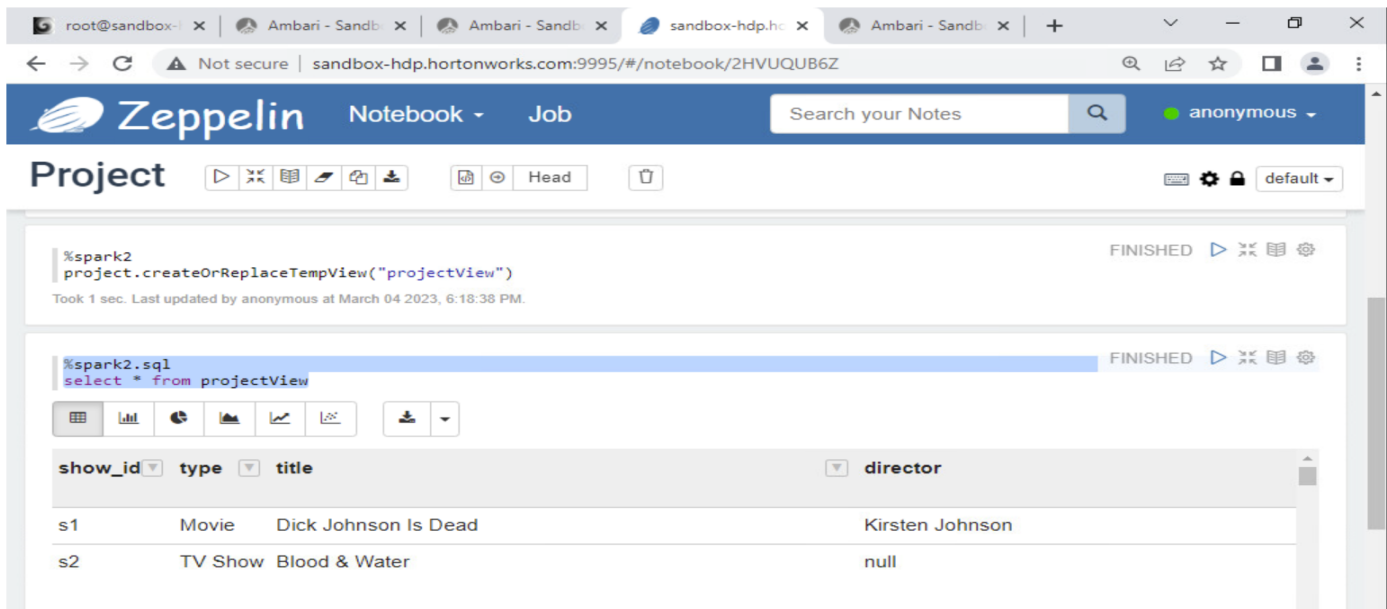
```
%spark2
project.printSchema()
```

Below the code, the output is displayed in a tree format:

```
root
|-- show_id: string (nullable = true)
|-- type: string (nullable = true)
|-- title: string (nullable = true)
|-- director: string (nullable = true)
|-- cast: string (nullable = true)
|-- country: string (nullable = true)
|-- date_added: string (nullable = true)
|-- release_year: string (nullable = true)
|-- rating: string (nullable = true)
|-- duration: string (nullable = true)
|-- listed_in: string (nullable = true)
|-- description: string (nullable = true)
```

The status "FINISHED" is shown in the top right corner of the code block.

Step-11 - we will now create a temporary view for access by SQL commands and display the data from view.



The screenshot shows a Zeppelin Notebook interface. The top bar includes the Zeppelin logo, 'Notebook' and 'Job' tabs, a search bar, and a user dropdown set to 'anonymous'. Below the top bar is a 'Project' header with various icons and a 'Head' button. The main area contains two code blocks. The first block, labeled '%spark2', contains the command 'project.createOrReplaceTempView("projectView")' and shows a status of 'FINISHED' with a timestamp. The second block, labeled '%spark2.sql', contains the query 'select * from projectView' and also shows a status of 'FINISHED'. Below the code blocks is a table view with columns 'show_id', 'type', 'title', and 'director'. The table contains two rows: 's1' (Movie) with title 'Dick Johnson Is Dead' and director 'Kirsten Johnson', and 's2' (TV Show) with title 'Blood & Water' and director 'null'.

show_id	type	title	director
s1	Movie	Dick Johnson Is Dead	Kirsten Johnson
s2	TV Show	Blood & Water	null

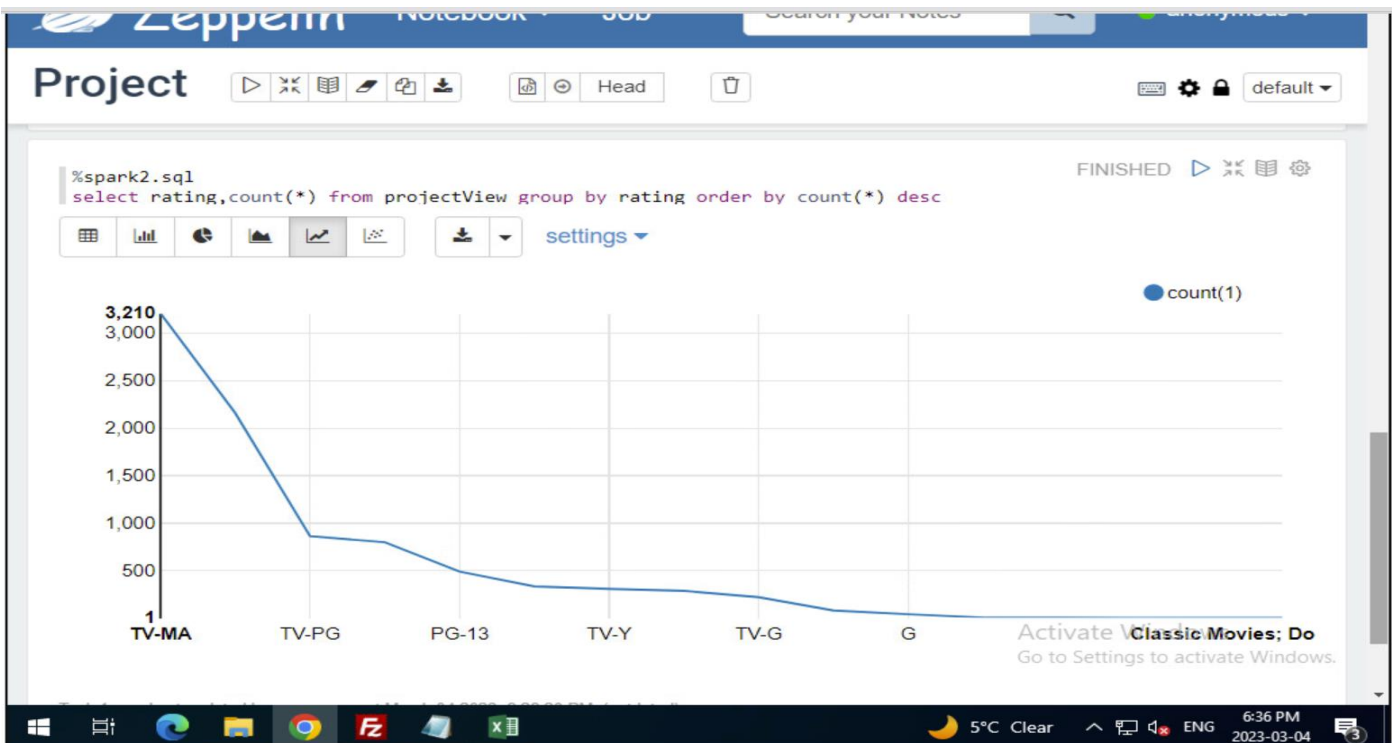
Step- 12 – Performing Various Analysis

1.) Based on rating, we can know which type of movies or TV shows are most popular.

Query: -

%spark2.sql

select rating,count(*) from projectView group by rating order by count(*) desc

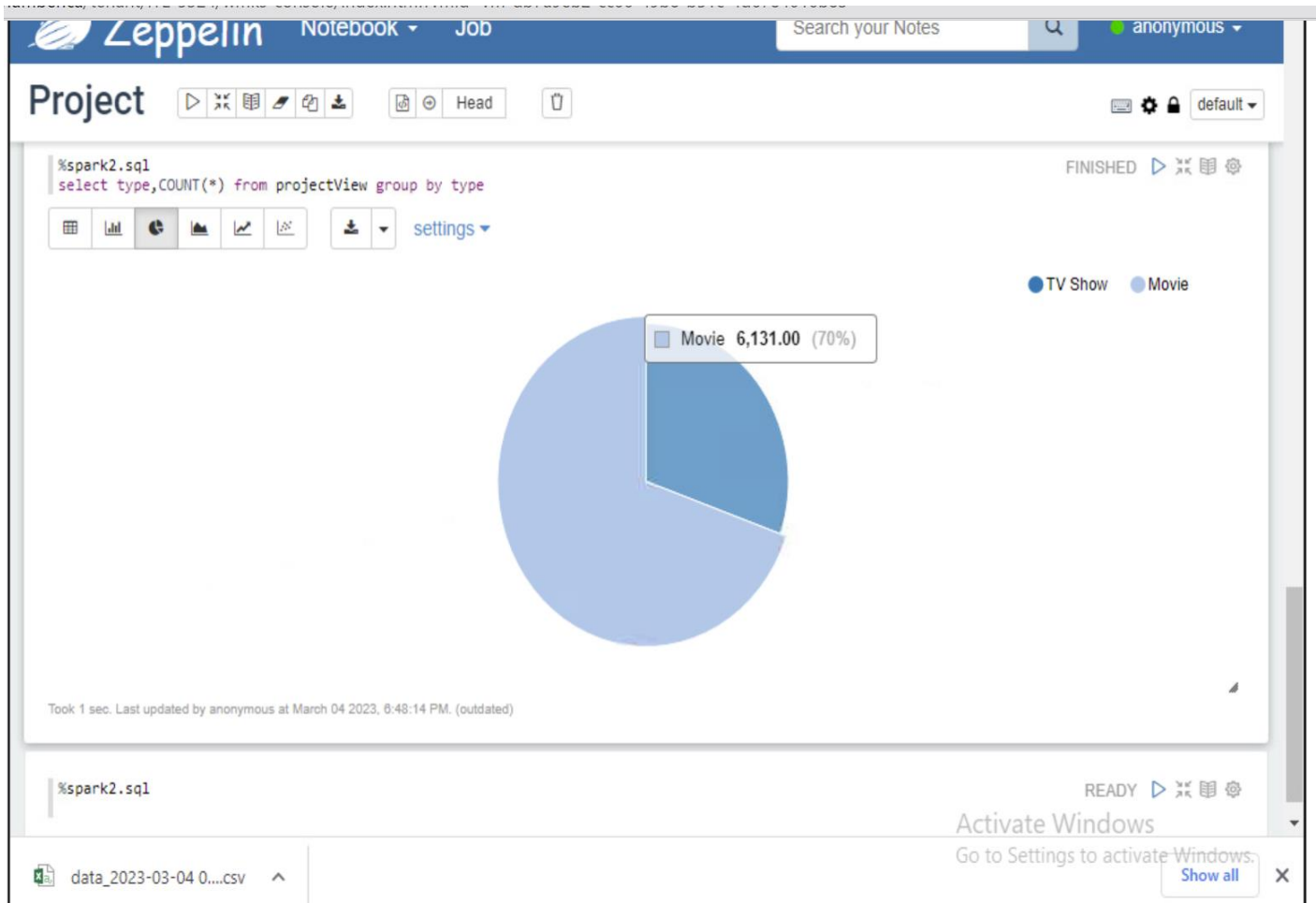


2.) Here we have analysis what interests more movies or TV shows.

Query:-

```
%spark2.sql
```

```
select type,COUNT(*) from projectView group by type
```

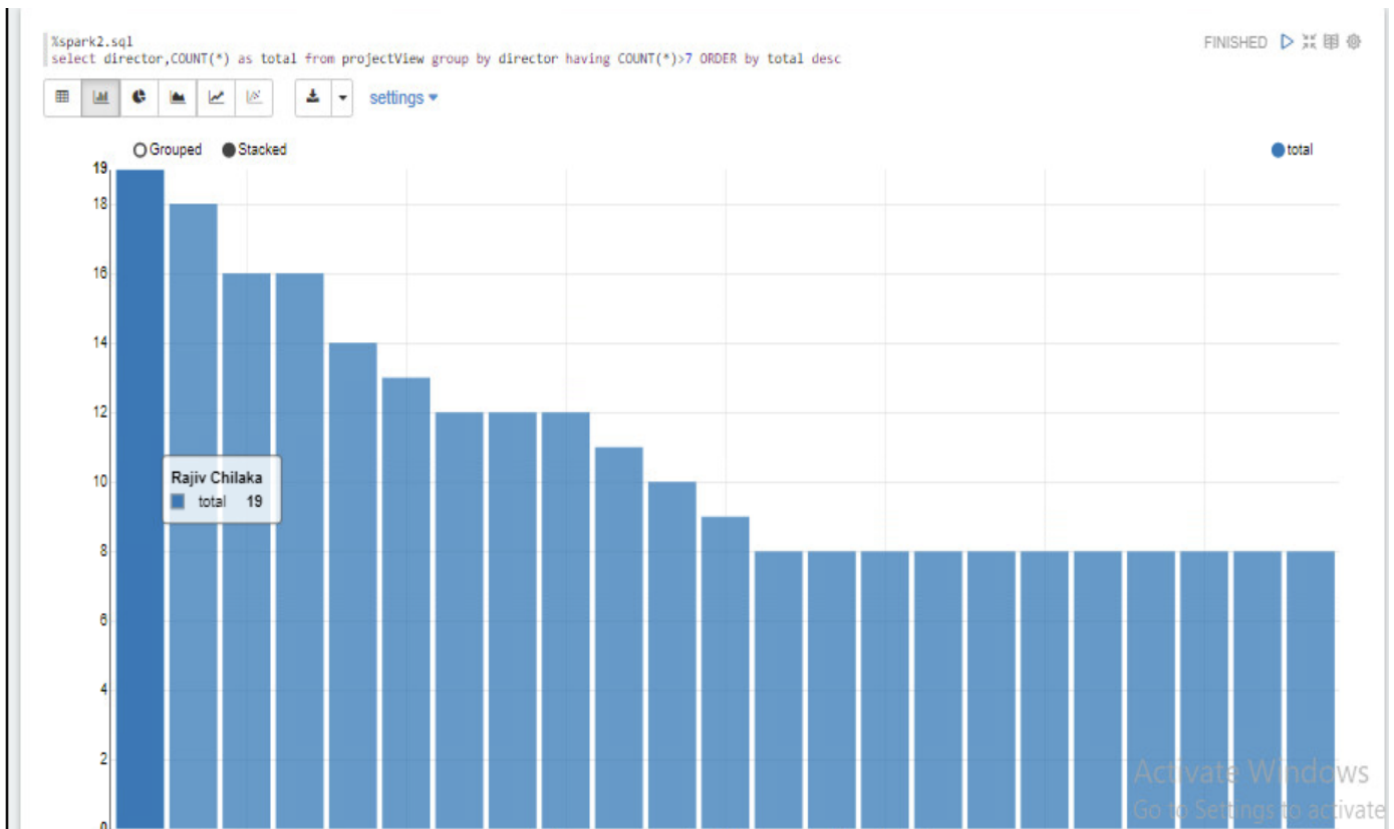


3.)Here we have analysis which actor have work more in movies and TV shows both(must have atleast worked in more than 7 movies result).

Query:-

```
%spark2.sql
```

```
select director,COUNT(*) as total from projectView group by director  
having COUNT(*)>7 ORDER by total desc
```



4.)Here,we have analyzed what kind of movies or TV shows are most popular(MAXIMUM 10 list).

Query:-

%spark2.sql

select listed_in,count(*) as total from projectView group by listed_in order by total desc limit 10

