# PRINCIPLES OF BIG DATA MANAGEMENT

## PHASE 2 REPORT

Team Member:

HARUN SAI KUMAR GENTE(16294902)

RAHUL BABU BICHINEPALLY (16300459)

SARIKA REDDY KOTA (16302630)

**SOFTWARES PLANNING TO USE:**

- Queries: Apache Spark with Scala.
- Visualization: pycharm with pyspark and python.
- Web application: HTML, CSS and Java script.

**IMPLEMENTATION:**

- First, we need to creatre a developer account to access the data from Twitter.

  Twitter developer account can be created from the link below.

  https://developer.twitter.com/en

- For analyzing the twitter data we need to extract the data from Twitter.
- So, we used python code for the extraction of tweets and save them into JSON format, the output of the code contains the tweets in JSON format .
- The JSON tweets are added to Apache SparkSQL in the form of Views.
- Python IDE is used to write queries and visualize the outputs with tables and charts.
- Then by using HTML, CSS and Java script we develop a web application.

**Libraries used:**

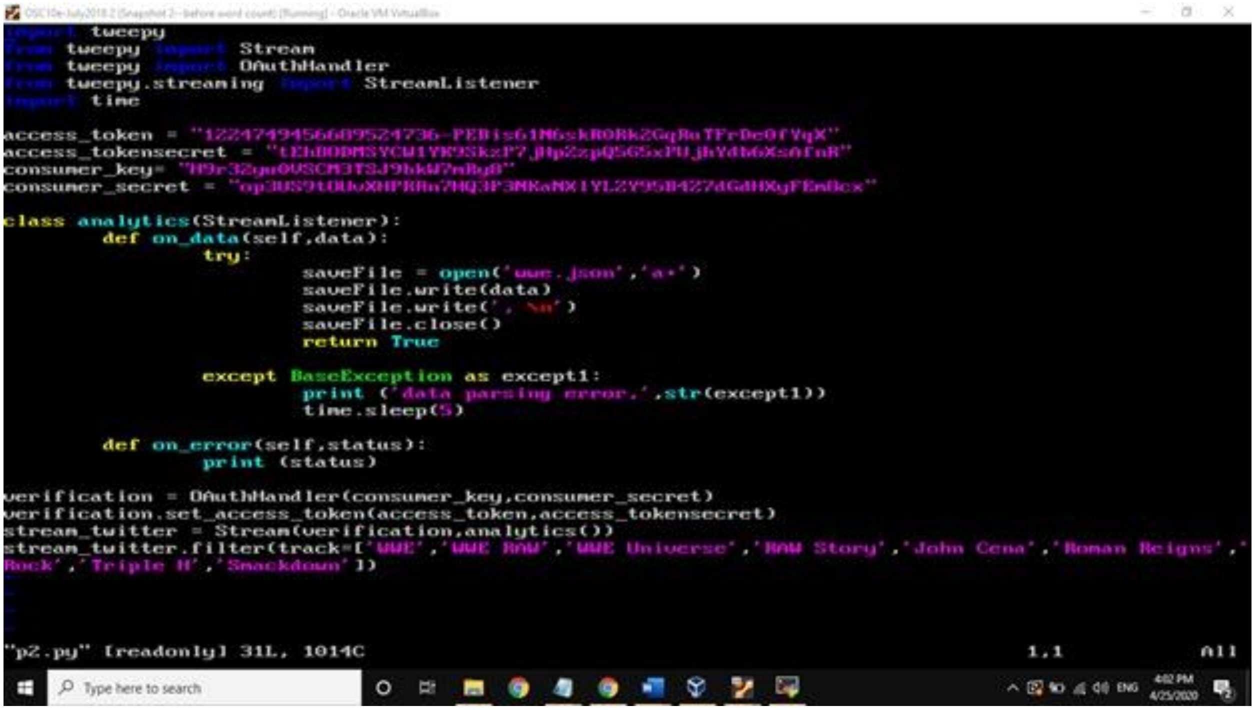
- Matplot for visualizations
- Flask for web application

**Code for Extraction of tweets:**

With the help of generated tokens and tweepy library we have extracted the tweets from twitter database.

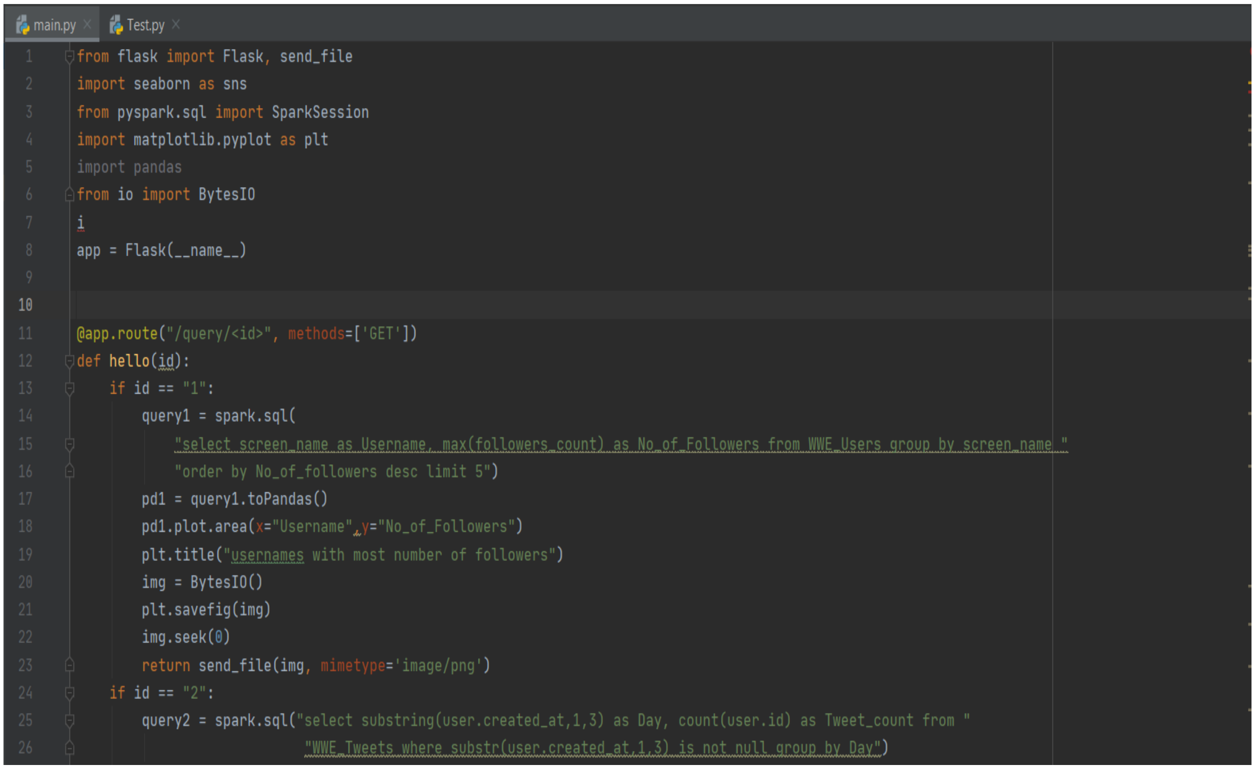
Source code: tweet_extraction.py

## Python Code for Tweet Extraction :

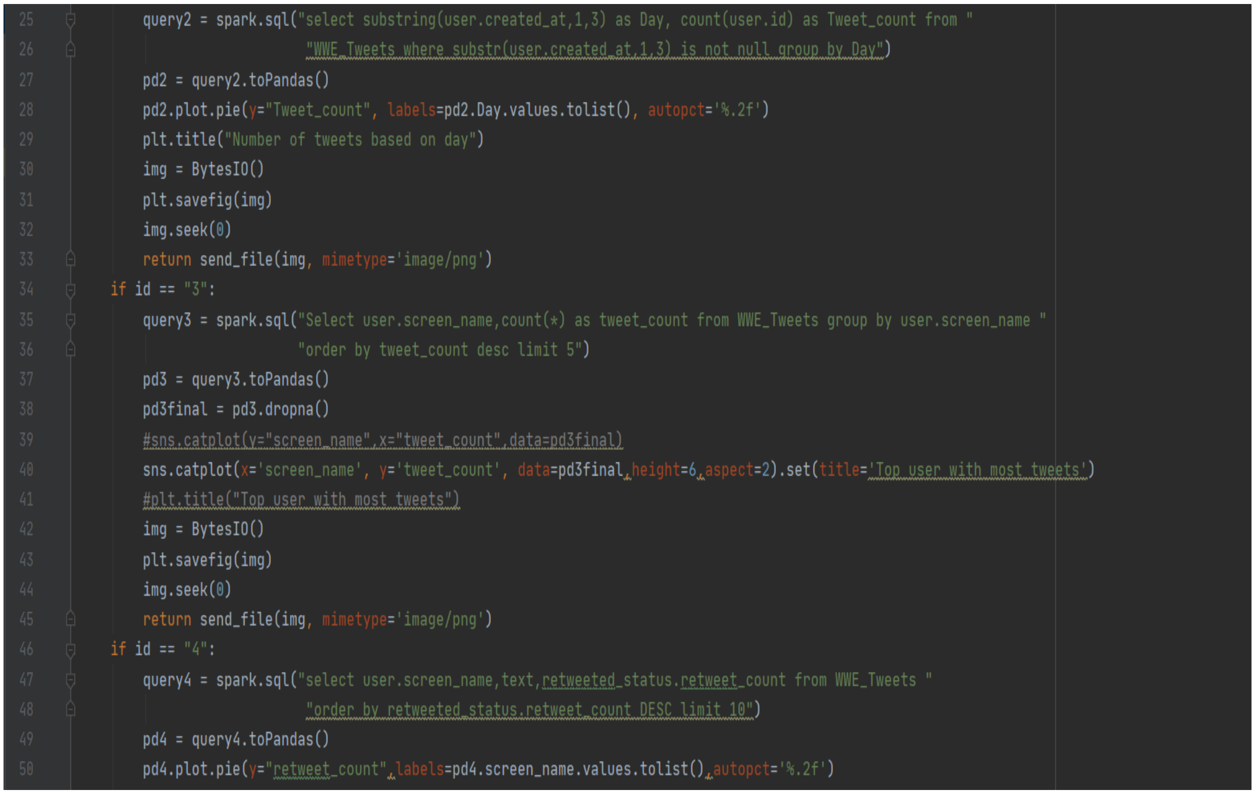

OSC10e-July2018.2 (Snapshot 2 - before word count) [Running] - Oracle VM VirtualBox
Type here to search
ENG
4:02 PM
4/25/2020

```
import tweepy
from tweepy import Stream
from tweepy import OAuthHandler
from tweepy.streaming import StreamListener
import time

access_token = "1224749456609524736-PEBis61N6skR0Rk2GqRuTFrDe0fYqX"
access_tokensecret = "tEhDODMSYCW1YK9SkzP7JHp2zpQ5G5xPUjhYdb6XsAfnR"
consumer_key= "H9r32yu0VSCM3TSJ9bkW7mByB"
consumer_secret = "op3US9tOUvXHPRRn7HQ3P3NKaNX1YLZY95B4Z7dGdHXyFEmBcx"

class analytics(StreamListener):
        def on_data(self,data):
                try:
                        saveFile = open('wwe.json','a+')
                        saveFile.write(data)
                        saveFile.write(', \n')
                        saveFile.close()
                        return True

                except BaseException as except1:
                        print ('data parsing error.',str(except1))
                        time.sleep(5)

        def on_error(self,status):
                print (status)

verification = OAuthHandler(consumer_key,consumer_secret)
verification.set_access_token(access_token,access_tokensecret)
stream_twitter = Stream(verification,analytics())
stream_twitter.filter(track=['WWE','WWE RAW','WWE Universe','RAW Story','John Cena','Roman Reigns','
Rock','Triple H','Smackdown'])
~
~
~
"p2.py" [readonly] 31L, 1014C                                                    1,1           All
```

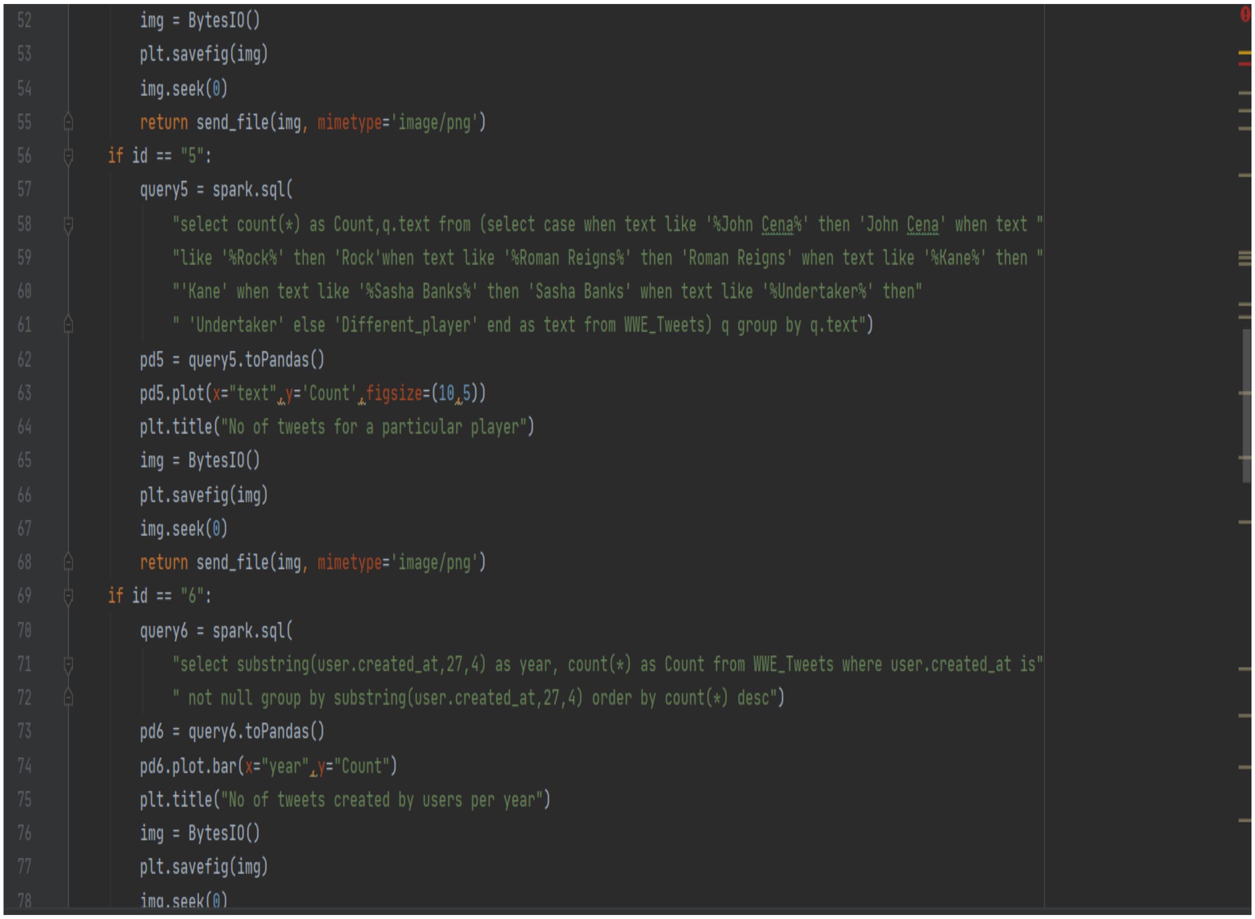
## Python Code for Queries:



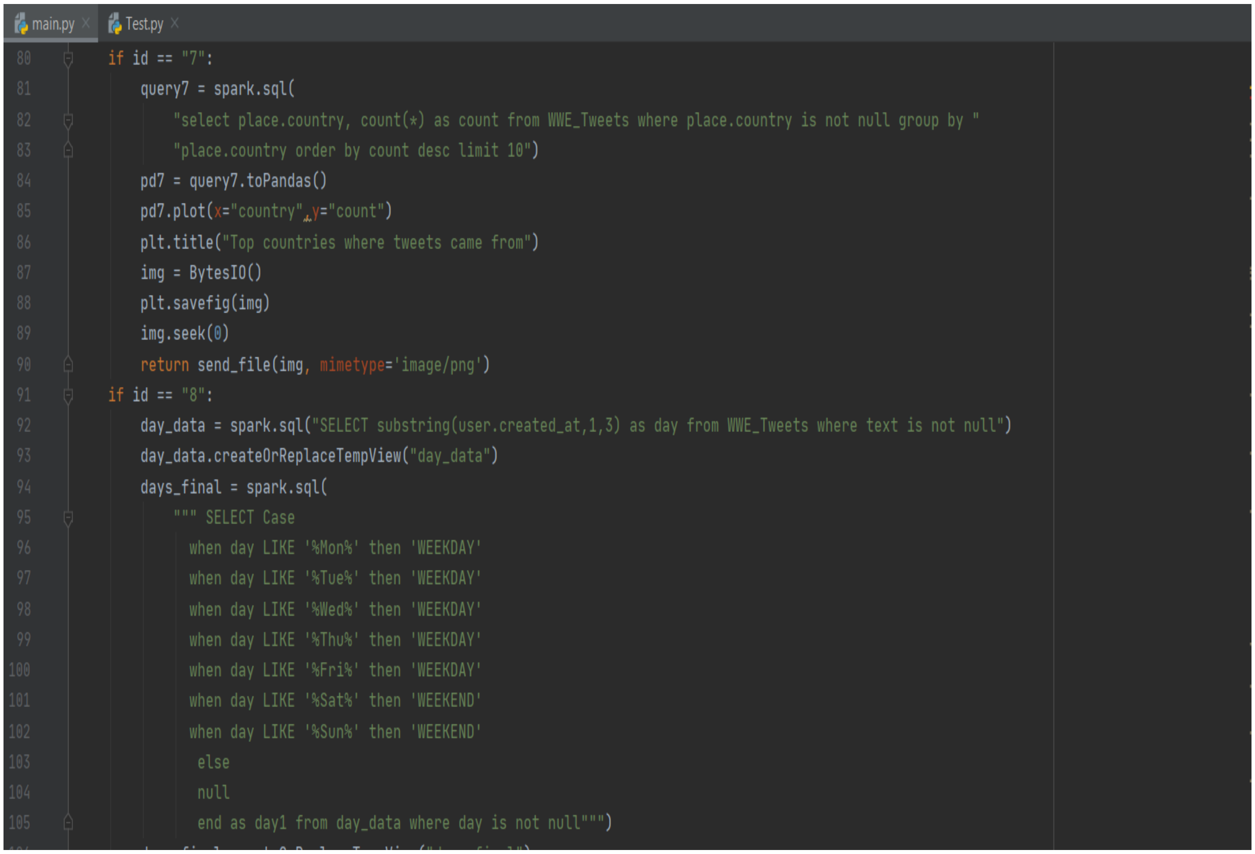
```python
from flask import Flask, send_file
import seaborn as sns
from pyspark.sql import SparkSession
import matplotlib.pyplot as plt
import pandas
from io import BytesIO
i
app = Flask(__name__)


@app.route("/query/<id>", methods=['GET'])
def hello(id):
    if id == "1":
        query1 = spark.sql(
            "select screen_name as Username, max(followers_count) as No_of_Followers from WWE_Users group by screen_name "
            "order by No_of_followers desc limit 5")
        pd1 = query1.toPandas()
        pd1.plot.area(x="Username",y="No_of_Followers")
        plt.title("usernames with most number of followers")
        img = BytesIO()
        plt.savefig(img)
        img.seek(0)
        return send_file(img, mimetype='image/png')
    if id == "2":
        query2 = spark.sql("select substring(user.created_at,1,3) as Day, count(user.id) as Tweet_count from "
                           "WWE_Tweets where substr(user.created_at,1,3) is not null group by Day")
```



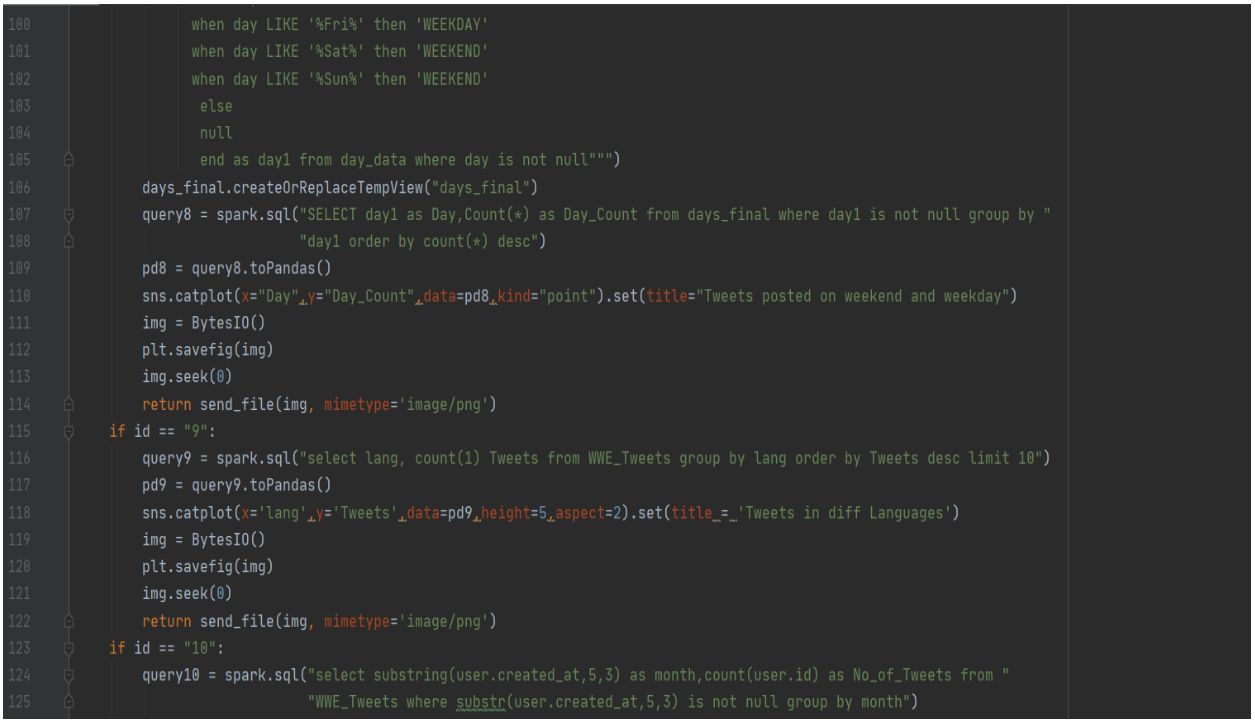
```
        query2 = spark.sql("select substring(user.created_at,1,3) as Day, count(user.id) as Tweet_count from "
                           "WWE_Tweets where substr(user.created_at,1,3) is not null group by Day")
        pd2 = query2.toPandas()
        pd2.plot.pie(y="Tweet_count", labels=pd2.Day.values.tolist(), autopct='%.2f')
        plt.title("Number of tweets based on day")
        img = BytesIO()
        plt.savefig(img)
        img.seek(0)
        return send_file(img, mimetype='image/png')
    if id == "3":
        query3 = spark.sql("Select user.screen_name,count(*) as tweet_count from WWE_Tweets group by user.screen_name "
                          "order by tweet_count desc limit 5")
        pd3 = query3.toPandas()
        pd3final = pd3.dropna()
        #sns.catplot(y="screen_name",x="tweet_count",data=pd3final)
        sns.catplot(x='screen_name', y='tweet_count', data=pd3final,height=6,aspect=2).set(title='Top user with most tweets')
        #plt.title("Top user with most tweets")
        img = BytesIO()
        plt.savefig(img)
        img.seek(0)
        return send_file(img, mimetype='image/png')
    if id == "4":
        query4 = spark.sql("select user.screen_name,text,retweeted_status.retweet_count from WWE_Tweets "
                           "order by retweeted_status.retweet_count DESC limit 10")
        pd4 = query4.toPandas()
        pd4.plot.pie(y="retweet_count",labels=pd4.screen_name.values.tolist(),autopct='%.2f')
```



```
        img = BytesIO()
        plt.savefig(img)
        img.seek(0)
        return send_file(img, mimetype='image/png')
    if id == "5":
        query5 = spark.sql(
            "select count(*) as Count,q.text from (select case when text like '%John Cena%' then 'John Cena' when text "
            "like '%Rock%' then 'Rock'when text like '%Roman Reigns%' then 'Roman Reigns' when text like '%Kane%' then "
            "'Kane' when text like '%Sasha Banks%' then 'Sasha Banks' when text like '%Undertaker%' then"
            " 'Undertaker' else 'Different_player' end as text from WWE_Tweets) q group by q.text")
        pd5 = query5.toPandas()
        pd5.plot(x="text",y='Count',figsize=(10,5))
        plt.title("No of tweets for a particular player")
        img = BytesIO()
        plt.savefig(img)
        img.seek(0)
        return send_file(img, mimetype='image/png')
    if id == "6":
        query6 = spark.sql(
            "select substring(user.created_at,27,4) as year, count(*) as Count from WWE_Tweets where user.created_at is"
            " not null group by substring(user.created_at,27,4) order by count(*) desc")
        pd6 = query6.toPandas()
        pd6.plot.bar(x="year",y="Count")
        plt.title("No of tweets created by users per year")
        img = BytesIO()
        plt.savefig(img)
        img.seek(0)
```

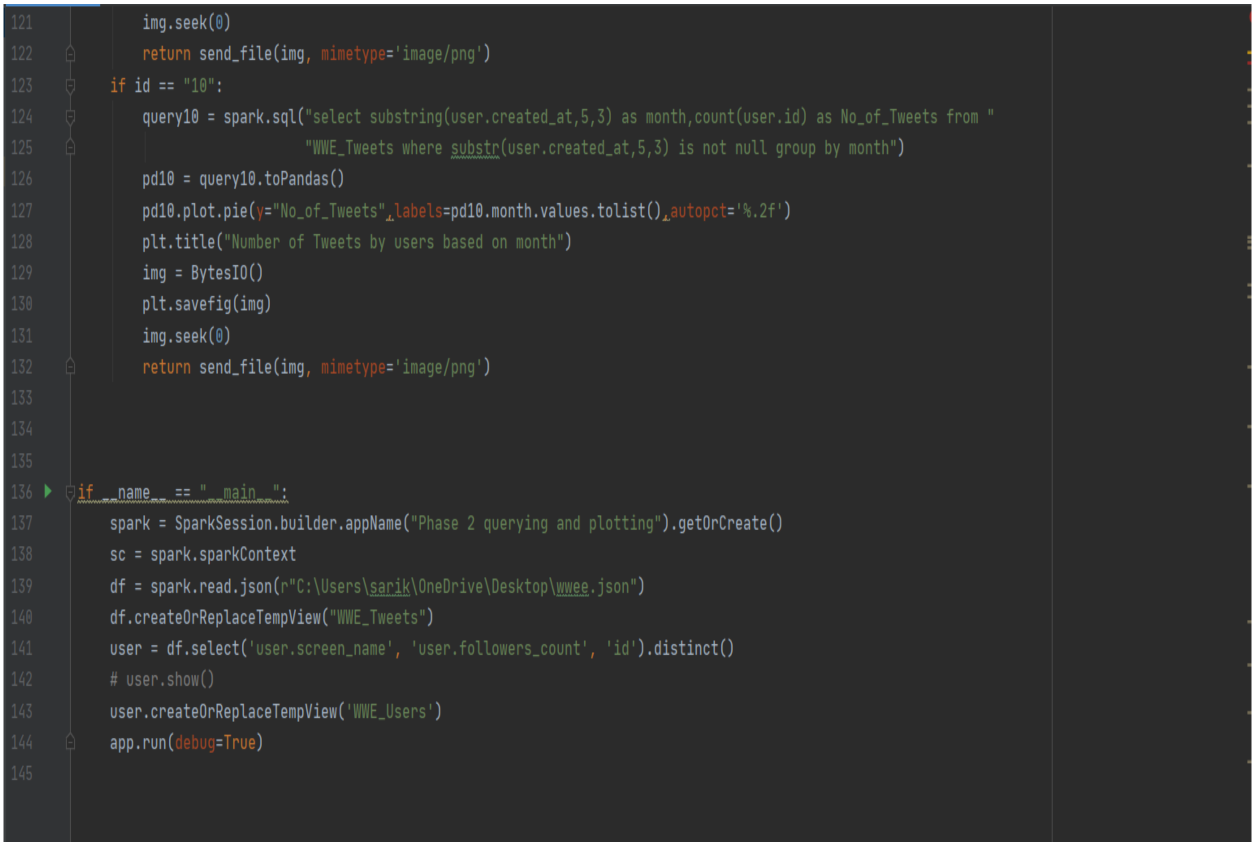


```
    if id == "7":
        query7 = spark.sql(
            "select place.country, count(*) as count from WWE_Tweets where place.country is not null group by "
            "place.country order by count desc limit 10")
        pd7 = query7.toPandas()
        pd7.plot(x="country",y="count")
        plt.title("Top countries where tweets came from")
        img = BytesIO()
        plt.savefig(img)
        img.seek(0)
        return send_file(img, mimetype='image/png')
    if id == "8":
        day_data = spark.sql("SELECT substring(user.created_at,1,3) as day from WWE_Tweets where text is not null")
        day_data.createOrReplaceTempView("day_data")
        days_final = spark.sql(
            """ SELECT Case
              when day LIKE '%Mon%' then 'WEEKDAY'
              when day LIKE '%Tue%' then 'WEEKDAY'
              when day LIKE '%Wed%' then 'WEEKDAY'
              when day LIKE '%Thu%' then 'WEEKDAY'
              when day LIKE '%Fri%' then 'WEEKDAY'
              when day LIKE '%Sat%' then 'WEEKEND'
              when day LIKE '%Sun%' then 'WEEKEND'
               else
               null
               end as day1 from day_data where day is not null""")
```



```
              when day LIKE '%Fri%' then 'WEEKDAY'
              when day LIKE '%Sat%' then 'WEEKEND'
              when day LIKE '%Sun%' then 'WEEKEND'
               else
               null
               end as day1 from day_data where day is not null""")
```

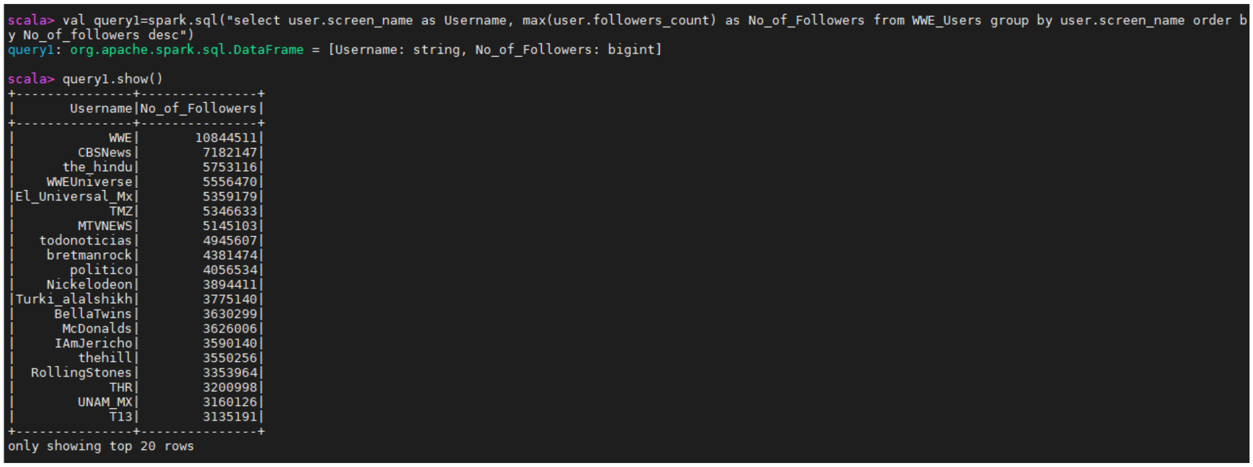
```
        days_final.createOrReplaceTempView("days_final")
        query8 = spark.sql("SELECT day1 as Day,Count(*) as Day_Count from days_final where day1 is not null group by "
                           "day1 order by count(*) desc")
        pd8 = query8.toPandas()
        sns.catplot(x="Day",y="Day_Count",data=pd8,kind="point").set(title="Tweets posted on weekend and weekday")
        img = BytesIO()
        plt.savefig(img)
        img.seek(0)
        return send_file(img, mimetype='image/png')
    if id == "9":
        query9 = spark.sql("select lang, count(1) Tweets from WWE_Tweets group by lang order by Tweets desc limit 10")
        pd9 = query9.toPandas()
        sns.catplot(x='lang',y='Tweets',data=pd9,height=5,aspect=2).set(title = 'Tweets in diff Languages')
        img = BytesIO()
        plt.savefig(img)
        img.seek(0)
        return send_file(img, mimetype='image/png')
    if id == "10":
        query10 = spark.sql("select substring(user.created_at,5,3) as month,count(user.id) as No_of_Tweets from "
                            "WWE_Tweets where substr(user.created_at,5,3) is not null group by month")
```



```
        img.seek(0)
        return send_file(img, mimetype='image/png')
    if id == "10":
        query10 = spark.sql("select substring(user.created_at,5,3) as month,count(user.id) as No_of_Tweets from "
                            "WWE_Tweets where substr(user.created_at,5,3) is not null group by month")
        pd10 = query10.toPandas()
        pd10.plot.pie(y="No_of_Tweets",labels=pd10.month.values.tolist(),autopct='%.2f')
        plt.title("Number of Tweets by users based on month")
        img = BytesIO()
        plt.savefig(img)
        img.seek(0)
        return send_file(img, mimetype='image/png')



if __name__ == "__main__":
    spark = SparkSession.builder.appName("Phase 2 querying and plotting").getOrCreate()
    sc = spark.sparkContext
    df = spark.read.json(r"C:\Users\sarik\OneDrive\Desktop\wwee.json")
    df.createOrReplaceTempView("WWE_Tweets")
    user = df.select('user.screen_name', 'user.followers_count', 'id').distinct()
    # user.show()
    user.createOrReplaceTempView('WWE_Users')
    app.run(debug=True)

```

**Query1:**

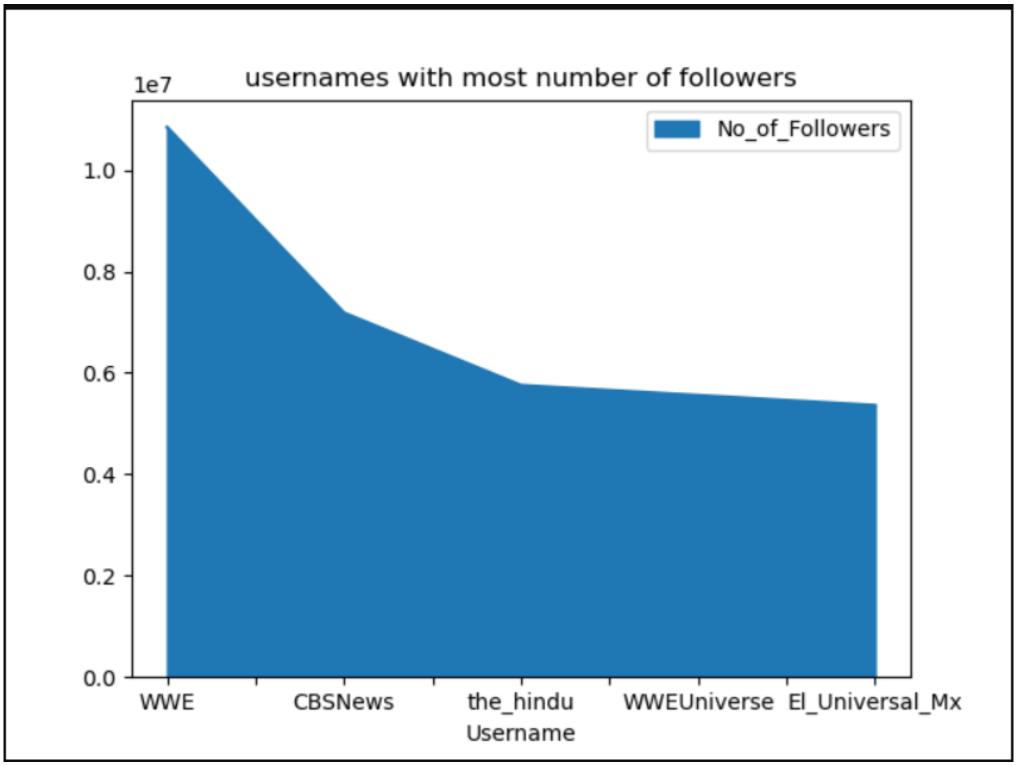
Usernames with most number of followers



```
scala> val query1=spark.sql("select user.screen_name as Username, max(user.followers_count) as No_of_Followers from WWE_Users group by user.screen_name order b
y No_of_followers desc")
query1: org.apache.spark.sql.DataFrame = [Username: string, No_of_Followers: bigint]

scala> query1.show()
+--------------+---------------+
|      Username|No_of_Followers|
+--------------+---------------+
|           WWE|       10844511|
|       CBSNews|        7182147|
|     the_hindu|        5753116|
|   WWEUniverse|        5556470|
|El_Universal_Mx|       5359179|
|           TMZ|        5346633|
|       MTVNEWS|        5145103|
|  todonoticias|        4945607|
|   bretmanrock|        4381474|
|      politico|        4056534|
|   Nickelodeon|        3894411|
|Turki_alalshikh|       3775140|
|    BellaTwins|        3630299|
|     McDonalds|        3626006|
|    IAmJericho|        3590140|
|       thehill|        3550256|
|  RollingStones|       3353964|
|           THR|        3200998|
|       UNAM_MX|        3160126|
|           T13|        3135191|
+--------------+---------------+
only showing top 20 rows
```

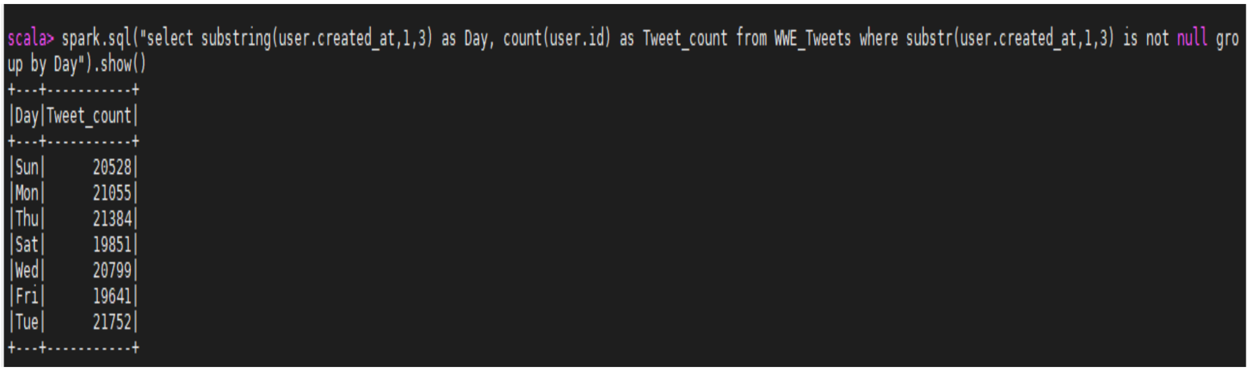
query1 = spark.sql("select screen_name as Username, max(followers_count) as No_of_Followers from WWE_Users group by screen_name order by No_of_followers desc limit 5")

**Visualization output:**



**Query2:**

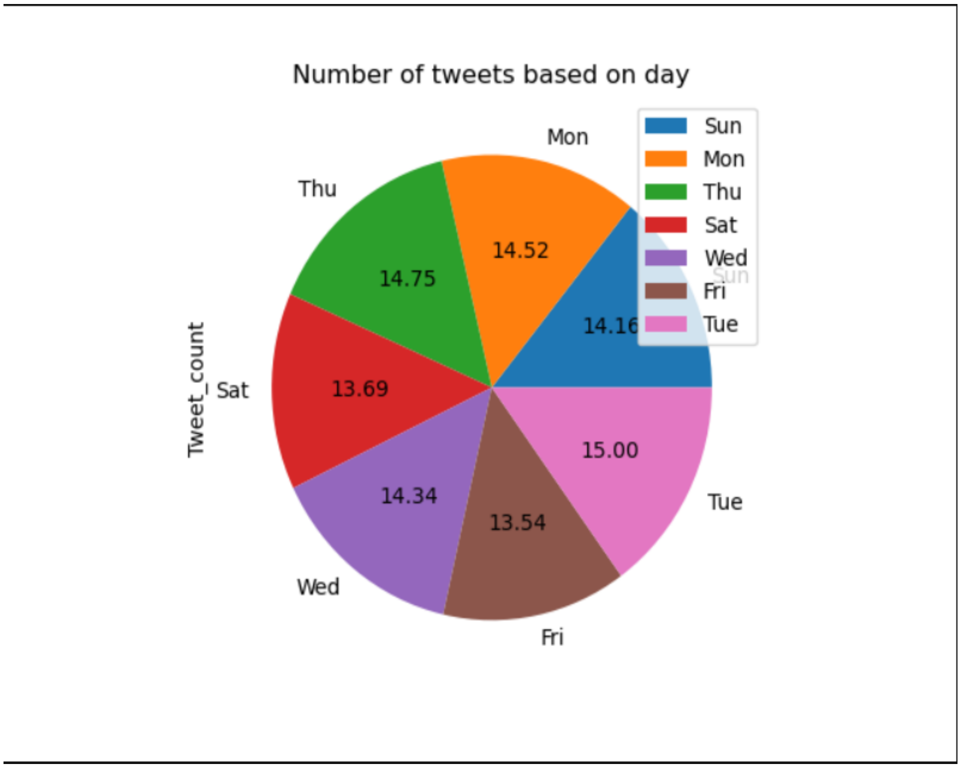
Number of tweets based on day



```
scala> spark.sql("select substring(user.created_at,1,3) as Day, count(user.id) as Tweet_count from WWE_Tweets where substr(user.created_at,1,3) is not null gro
up by Day").show()
+---+-----------+
|Day|Tweet_count|
+---+-----------+
|Sun|      20528|
|Mon|      21055|
|Thu|      21384|
|Sat|      19851|
|Wed|      20799|
|Fri|      19641|
|Tue|      21752|
+---+-----------+
```

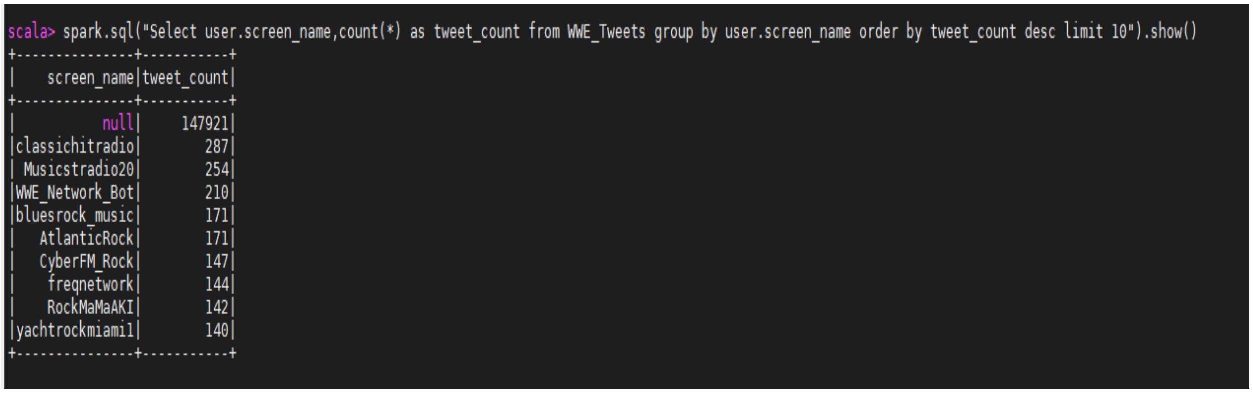
query2 = spark.sql("select substring(user.created_at,1,3) as Day, count(user.id) as Tweet_count from WWE_Tweets where substr(user.created_at,1,3) is not null group by Day")

**Visualization output:**



**Query3:**

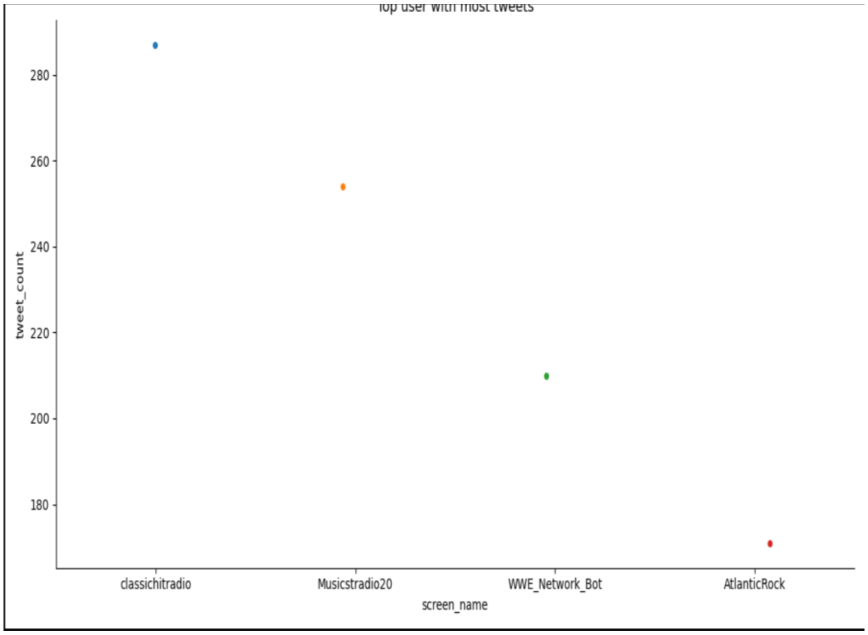
users with most number of  tweets



```
scala> spark.sql("Select user.screen_name,count(*) as tweet_count from WWE_Tweets group by user.screen_name order by tweet_count desc limit 10").show()
+---------------+-----------+
|    screen_name|tweet_count|
+---------------+-----------+
|           null|     147921|
|classichitradio|        287|
| Musicstradio20|        254|
|WWE_Network_Bot|        210|
|bluesrock_music|        171|
|   AtlanticRock|        171|
|   CyberFM_Rock|        147|
|    freqnetwork|        144|
|    RockMaMaAKI|        142|
|yachtrockmiami1|        140|
+---------------+-----------+
```

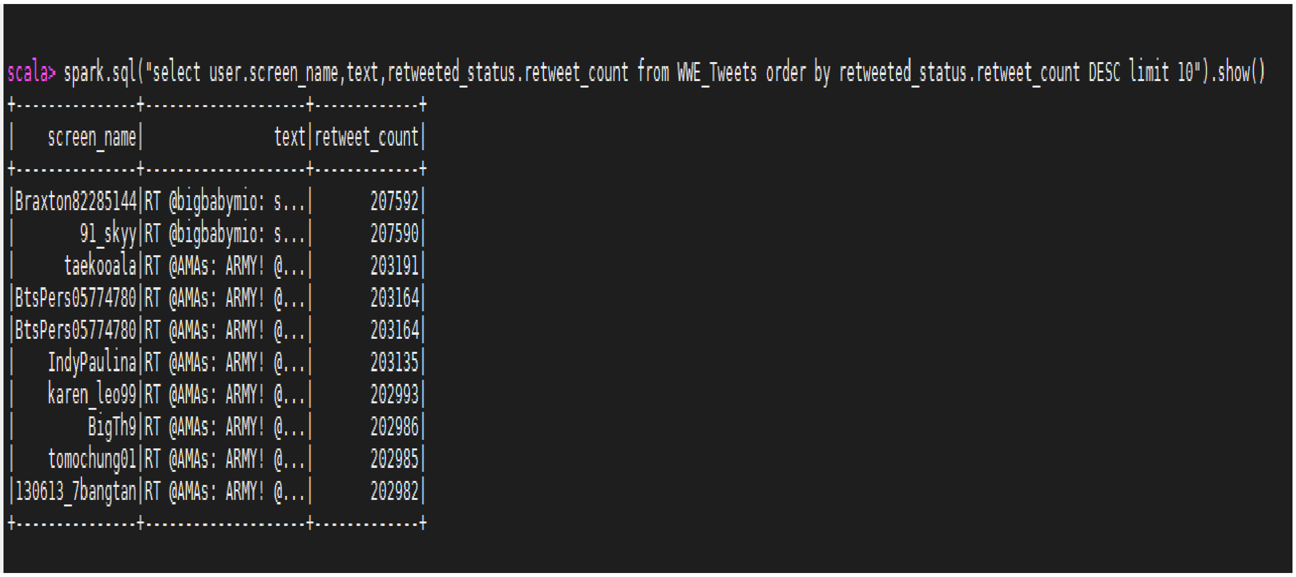
query3=spark.sql("Select user.screen_name,count(*) as tweet_count from WWE_Tweets group by user.screen_name order by tweet_count desc limit 10")

**Visualization output:**



**Query4:**

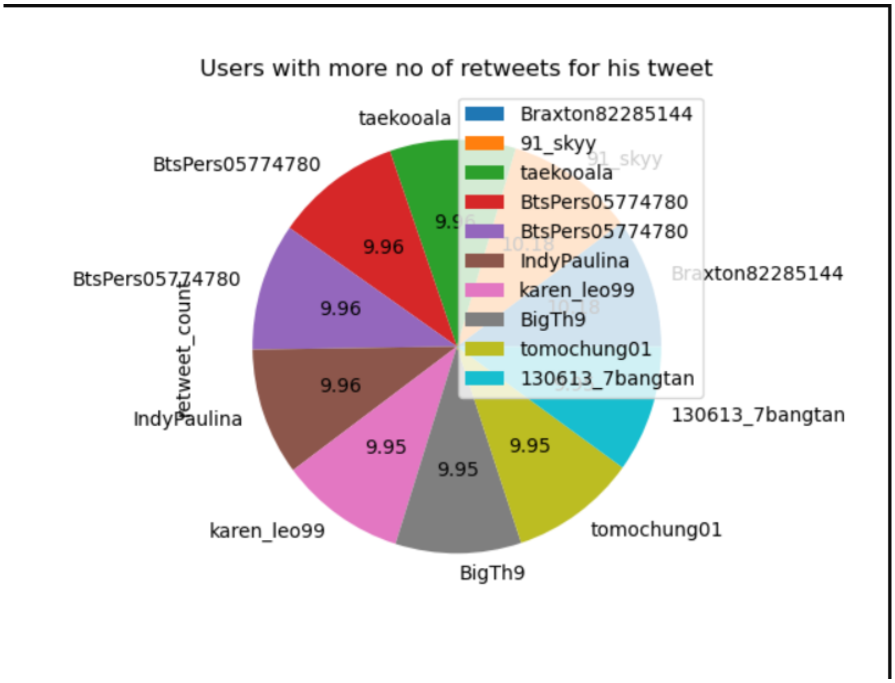
Users with more no of retweets for his tweet



```
scala> spark.sql("select user.screen_name,text,retweeted_status.retweet_count from WWE_Tweets order by retweeted_status.retweet_count DESC limit 10").show()
+-------------+------------------+-------------+
|  screen_name|              text|retweet_count|
+-------------+------------------+-------------+
|Braxton82285144|RT @bigbabymio: s...|       207592|
|      91_skyy|RT @bigbabymio: s...|       207590|
|    taekooala|RT @AMAs: ARMY! @...|       203191|
|BtsPers05774780|RT @AMAs: ARMY! @...|       203164|
|BtsPers05774780|RT @AMAs: ARMY! @...|       203164|
|   IndyPaulina|RT @AMAs: ARMY! @...|       203135|
|   karen_leo99|RT @AMAs: ARMY! @...|       202993|
|        BigTh9|RT @AMAs: ARMY! @...|       202986|
|   tomochung01|RT @AMAs: ARMY! @...|       202985|
|130613_7bangtan|RT @AMAs: ARMY! @...|       202982|
+-------------+------------------+-------------+
```

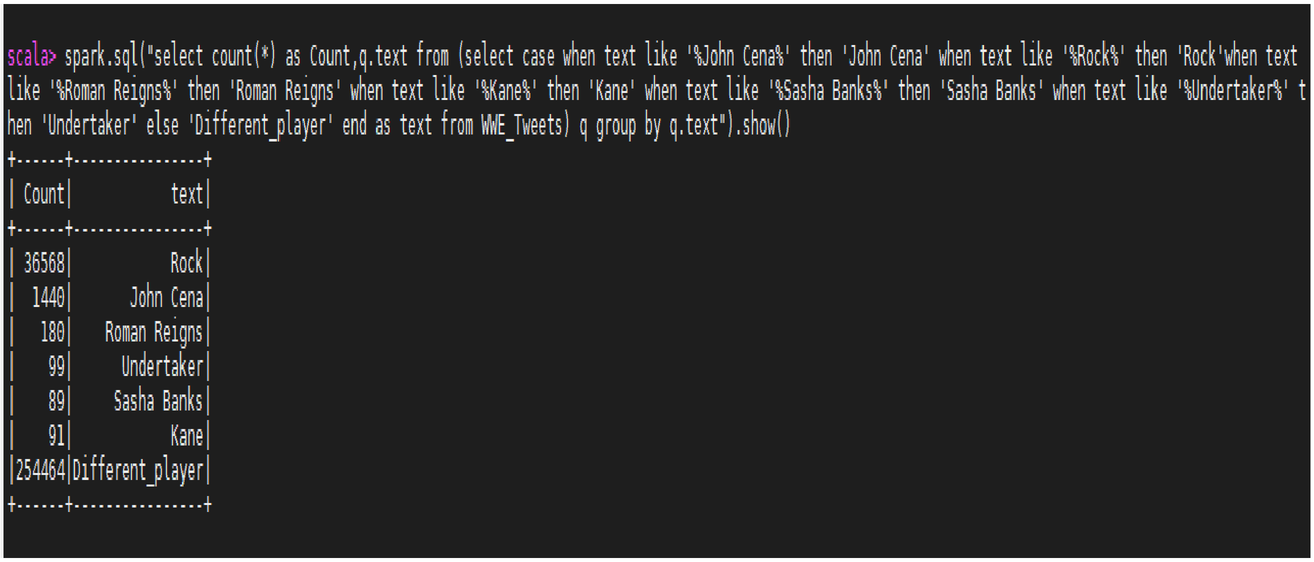
query4 = spark.sql("select user.screen_name,text,retweeted_status.retweet_count from WWE_Tweets order by retweeted_status.retweet_count DESC limit 10")

**Visualization output:**



**Query5:**

Number of Tweets for a particular player.

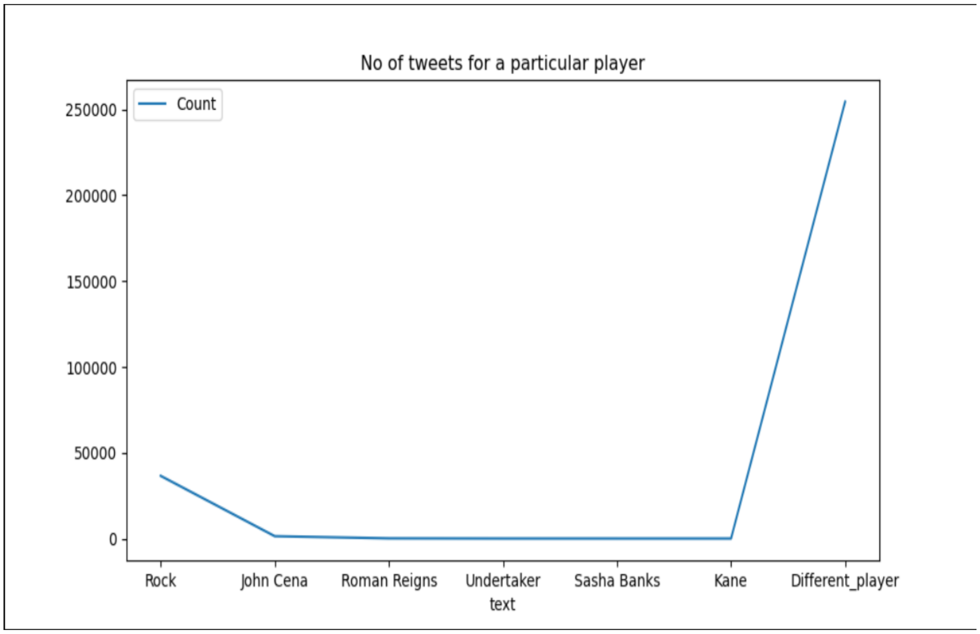


```
scala> spark.sql("select count(*) as Count,q.text from (select case when text like '%John Cena%' then 'John Cena' when text like '%Rock%' then 'Rock'when text
like '%Roman Reigns%' then 'Roman Reigns' when text like '%Kane%' then 'Kane' when text like '%Sasha Banks%' then 'Sasha Banks' when text like '%Undertaker%' t
hen 'Undertaker' else 'Different_player' end as text from WWE_Tweets) q group by q.text").show()
+------+----------------+
| Count|            text|
+------+----------------+
| 36568|            Rock|
|  1440|       John Cena|
|   180|    Roman Reigns|
|    99|      Undertaker|
|    89|     Sasha Banks|
|    91|            Kane|
|254464|Different_player|
+------+----------------+
```

query5 = spark.sql("select count(*) as Count,q.text from (select case when text like '%John Cena%' then 'John Cena' when text like '%Rock%' then 'Rock'when text like '%Roman Reigns%' then 'Roman Reigns' when text like '%Kane%' then 'Kane' when text like '%Sasha Banks%' then

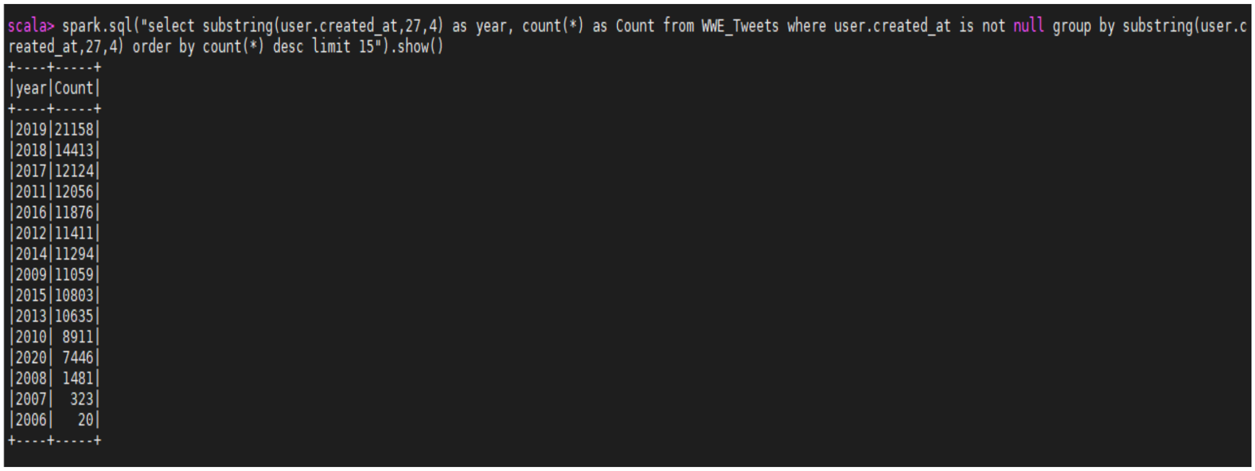
'Sasha Banks' when text like '%Undertaker%' then 'Undertaker' else 'Different_player' end as text from WWE_Tweets) q group by q.text")

**Visualization output:**



**Query6:**

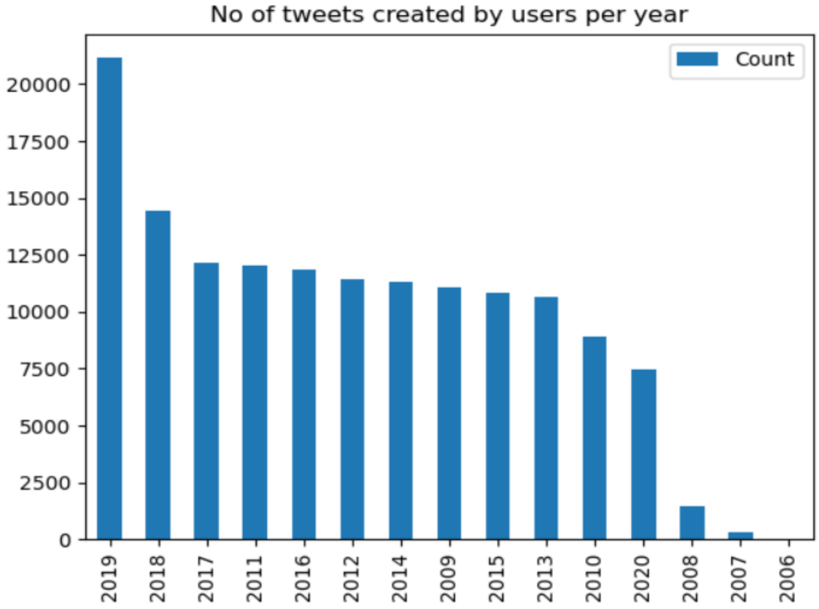
No of tweets created by users per year



```
scala> spark.sql("select substring(user.created_at,27,4) as year, count(*) as Count from WWE_Tweets where user.created_at is not null group by substring(user.c
reated_at,27,4) order by count(*) desc limit 15").show()
+----+-----+
|year|Count|
+----+-----+
|2019|21158|
|2018|14413|
|2017|12124|
|2011|12056|
|2016|11876|
|2012|11411|
|2014|11294|
|2009|11059|
|2015|10803|
|2013|10635|
|2010| 8911|
|2020| 7446|
|2008| 1481|
|2007|  323|
|2006|   20|
+----+-----+
```

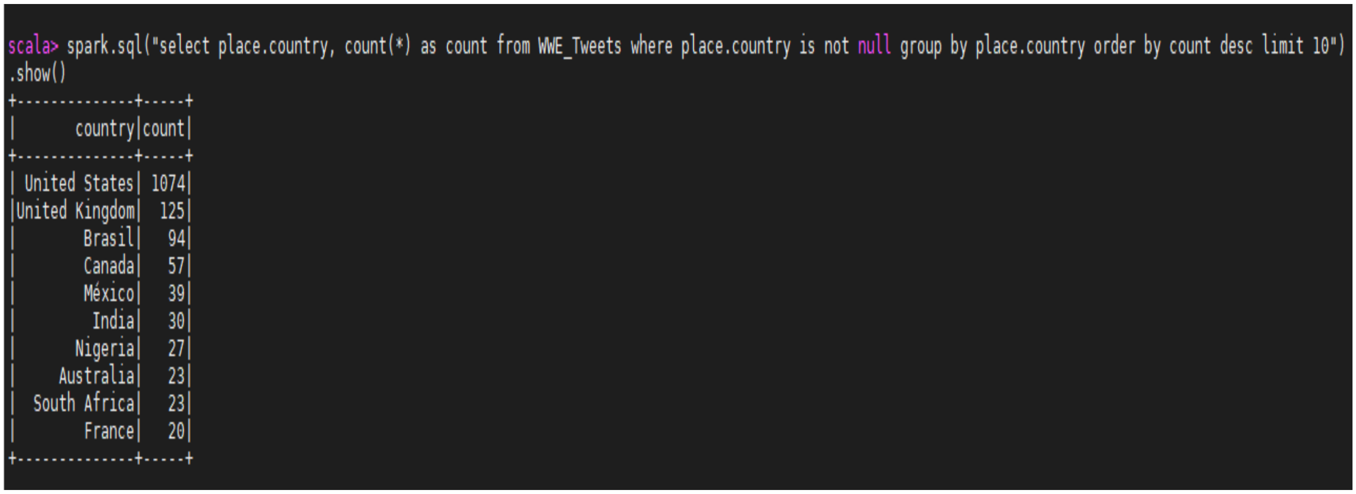
query6 = spark.sql("select substring(user.created_at,27,4) as year, count(*) as Count from WWE_Tweets where user.created_at is not null group by substring(user.created_at,27,4) order by count(*) desc")

**Visualization output:**



**Query7:**

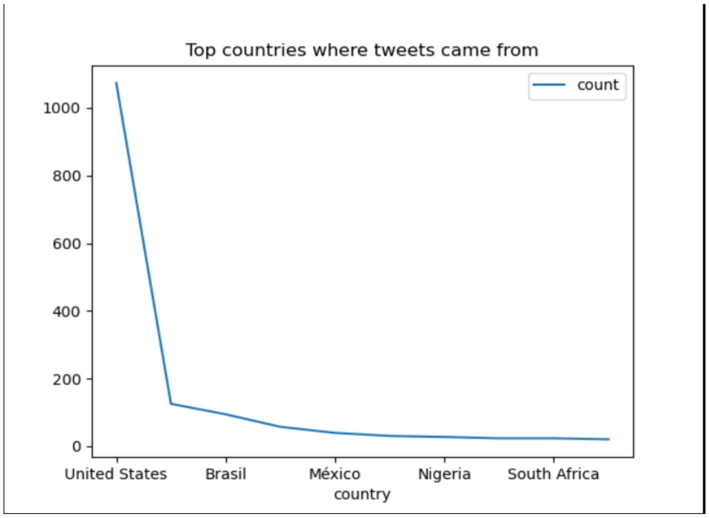
Top countries where tweets came from



```
scala> spark.sql("select place.country, count(*) as count from WWE_Tweets where place.country is not null group by place.country order by count desc limit 10")
.show()
+-------------+-----+
|      country|count|
+-------------+-----+
| United States| 1074|
|United Kingdom|  125|
|        Brasil|   94|
|        Canada|   57|
|        México|   39|
|         India|   30|
|       Nigeria|   27|
|     Australia|   23|
|  South Africa|   23|
|        France|   20|
+-------------+-----+
```

query7 = spark.sql("select place.country, count(*) as count from WWE_Tweets where place.country is not null group by place.country order by count desc limit 10")

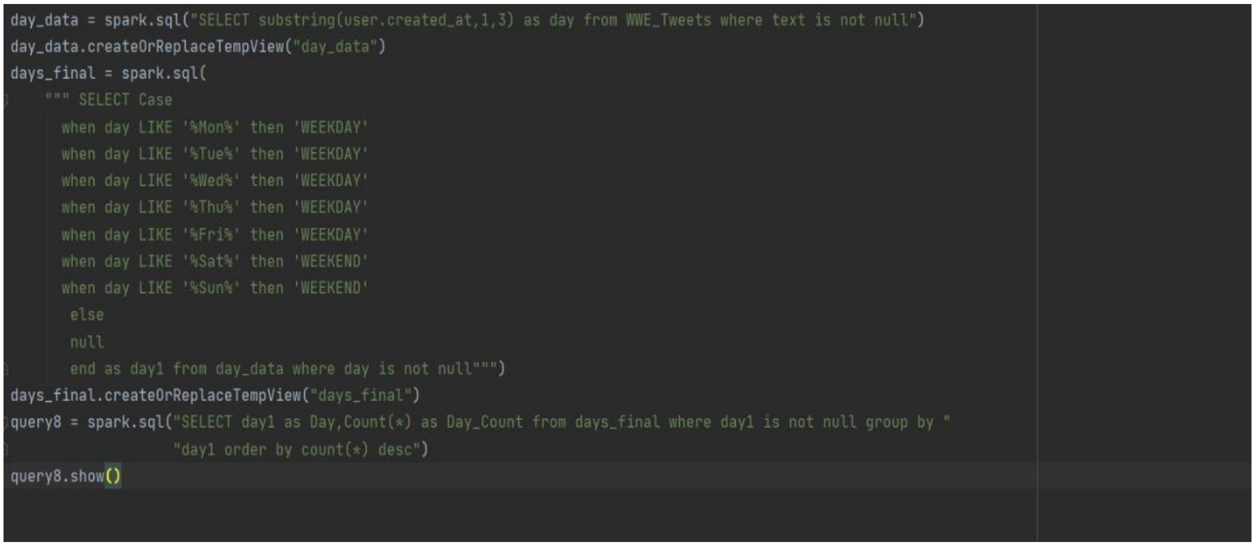
## Visualization output:



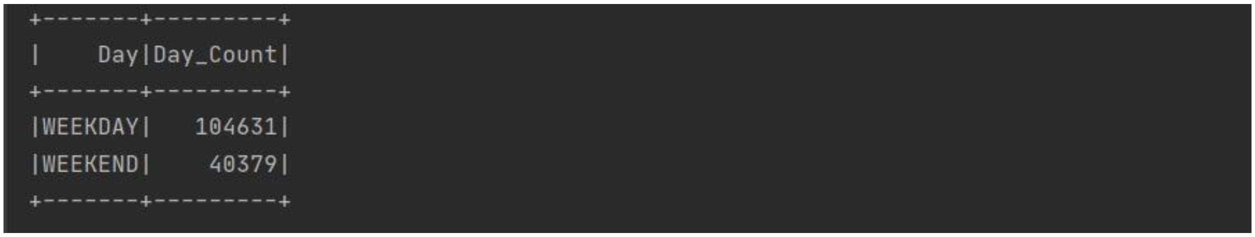
## Query8:

Tweets posted on weekend and weekday

```
day_data = spark.sql("SELECT substring(user.created_at,1,3) as day from WWE_Tweets where text is not null")

day_data.createOrReplaceTempView("day_data")


days_final = spark.sql(
        """ SELECT Case
         when day LIKE '%Mon%' then 'WEEKDAY'
         when day LIKE '%Tue%' then 'WEEKDAY'
         when day LIKE '%Wed%' then 'WEEKDAY'
         when day LIKE '%Thu%' then 'WEEKDAY'
         when day LIKE '%Fri%' then 'WEEKDAY'
         when day LIKE '%Sat%' then 'WEEKEND'
         when day LIKE '%Sun%' then 'WEEKEND'
          else
          null
          end as day1 from day_data where day is not null""")
days_final.createOrReplaceTempView("days_final")
query8 = spark.sql("SELECT day1 as Day,Count(*) as Day_Count from days_final where day1 is not null group by day1 order by count(*) desc")
```

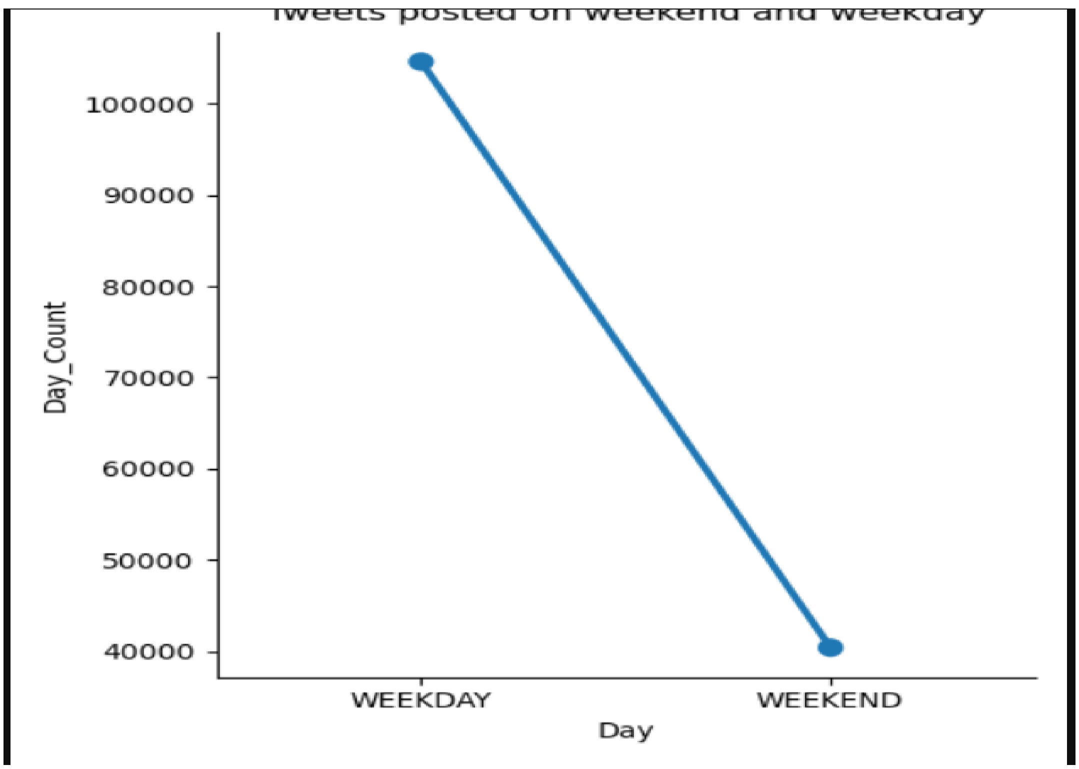


```
day_data = spark.sql("SELECT substring(user.created_at,1,3) as day from WWE_Tweets where text is not null")
day_data.createOrReplaceTempView("day_data")
days_final = spark.sql(
    """ SELECT Case
      when day LIKE '%Mon%' then 'WEEKDAY'
      when day LIKE '%Tue%' then 'WEEKDAY'
      when day LIKE '%Wed%' then 'WEEKDAY'
      when day LIKE '%Thu%' then 'WEEKDAY'
      when day LIKE '%Fri%' then 'WEEKDAY'
      when day LIKE '%Sat%' then 'WEEKEND'
      when day LIKE '%Sun%' then 'WEEKEND'
       else
       null
       end as day1 from day_data where day is not null""")
days_final.createOrReplaceTempView("days_final")
query8 = spark.sql("SELECT day1 as Day,Count(*) as Day_Count from days_final where day1 is not null group by "
                   "day1 order by count(*) desc")
query8.show()
```



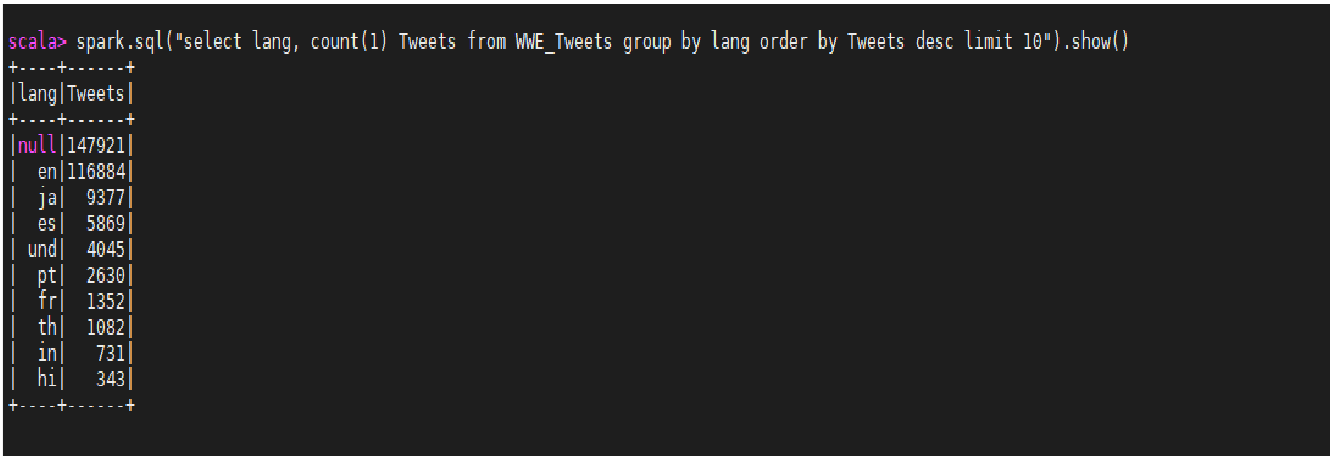
```
+-------+---------+
|    Day|Day_Count|
+-------+---------+
|WEEKDAY|   104631|
|WEEKEND|    40379|
+-------+---------+
```

**Visualization output:**



**Query9:**

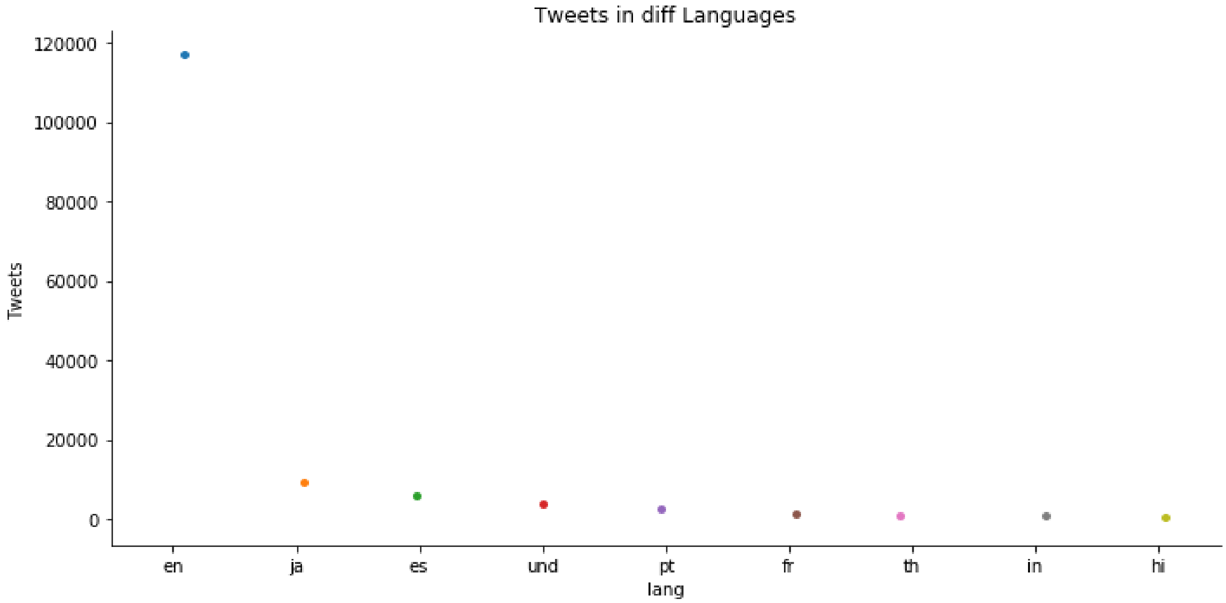
Tweet count based on language



```
scala> spark.sql("select lang, count(1) Tweets from WWE_Tweets group by lang order by Tweets desc limit 10").show()
+----+------+
|lang|Tweets|
+----+------+
|null|147921|
|  en|116884|
|  ja|  9377|
|  es|  5869|
| und|  4045|
|  pt|  2630|
|  fr|  1352|
|  th|  1082|
|  in|   731|
|  hi|   343|
+----+------+
```

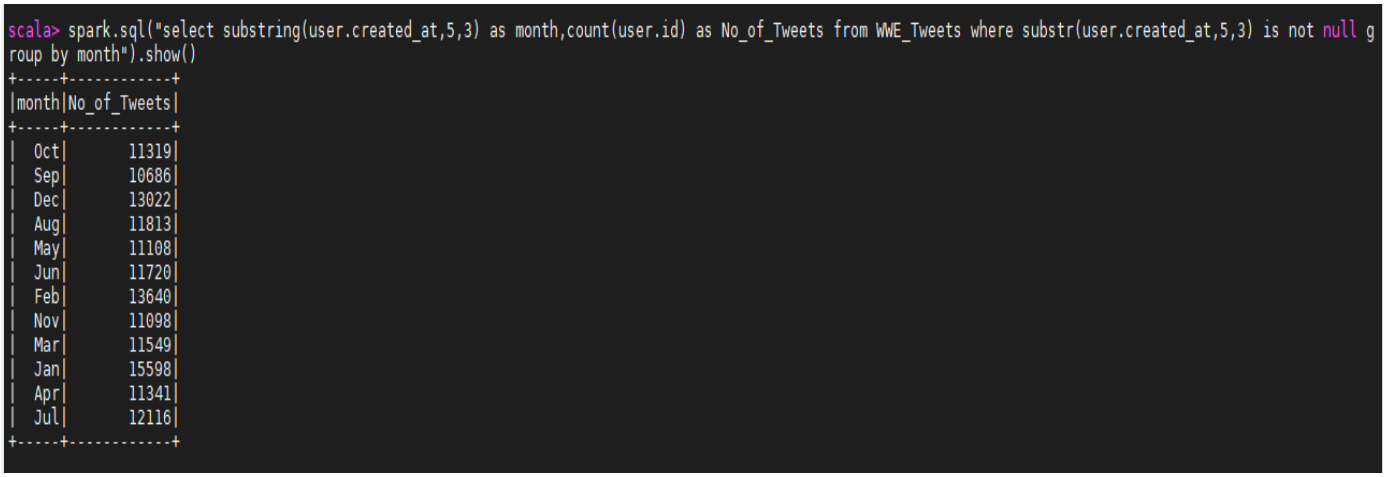
query9 = spark.sql("select lang, count(1) Tweets from WWE_Tweets group by lang order by Tweets desc limit 10")

**Visualization output:**



**Query10:**

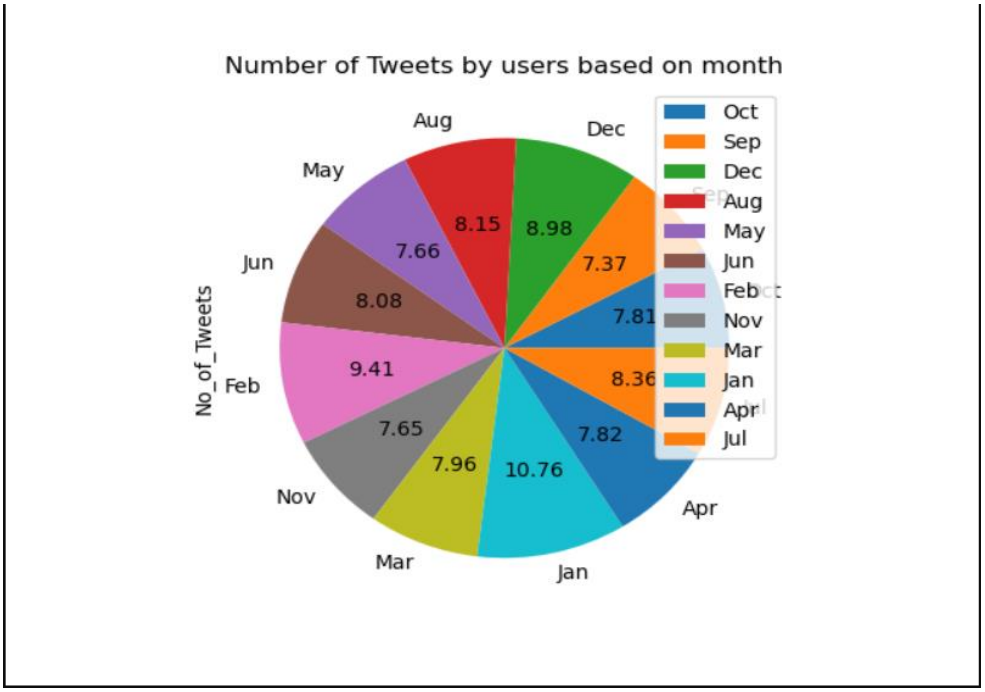
Number of Tweets by users based on month



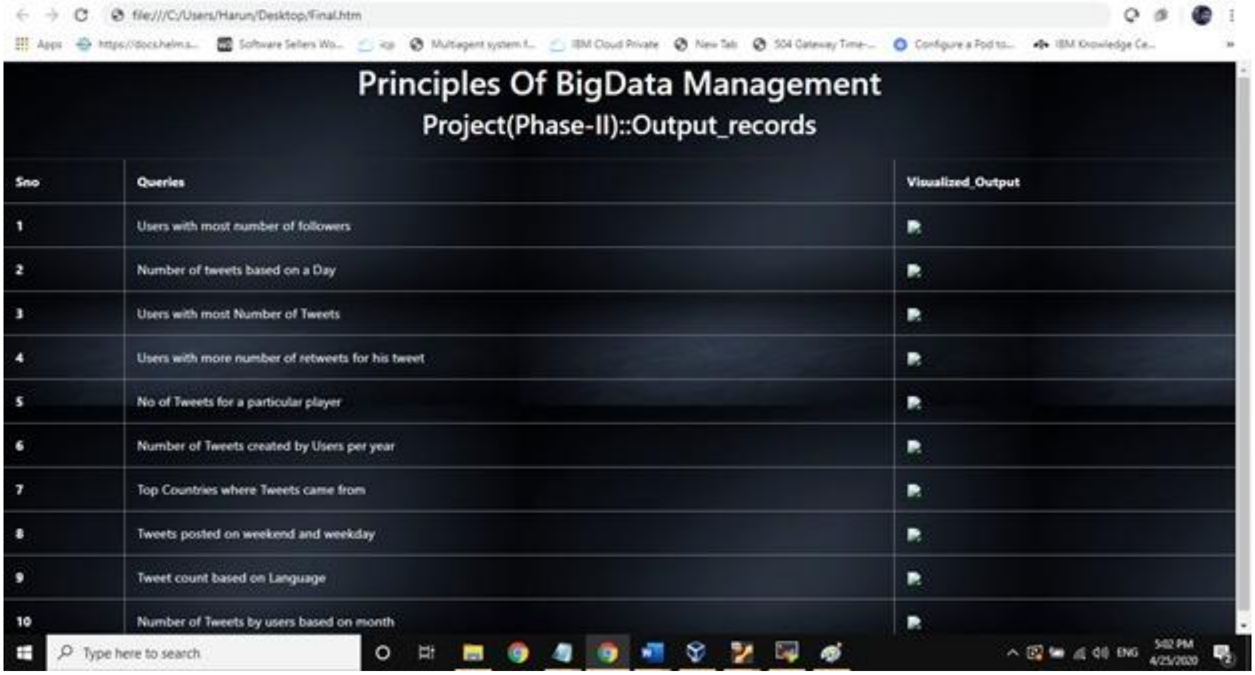
```
scala> spark.sql("select substring(user.created_at,5,3) as month,count(user.id) as No_of_Tweets from WWE_Tweets where substr(user.created_at,5,3) is not null g
roup by month").show()
+-----+------------+
|month|No_of_Tweets|
+-----+------------+
|  Oct|       11319|
|  Sep|       10686|
|  Dec|       13022|
|  Aug|       11813|
|  May|       11108|
|  Jun|       11720|
|  Feb|       13640|
|  Nov|       11098|
|  Mar|       11549|
|  Jan|       15598|
|  Apr|       11341|
|  Jul|       12116|
+-----+------------+
```

query10 = spark.sql("select substring(user.created_at,5,3) as month,count(user.id) as No_of_Tweets from WWE_Tweets where substr(user.created_at,5,3) is not null group by month")

**Visualization output:**



**Developed web application:**



Principles Of BigData Management

Project(Phase-II)::Output_records

| Sno | Queries | Visualized_Output |
|---|---|---|
| 1 | Users with most number of followers | |
| 2 | Number of tweets based on a Day | |
| 3 | Users with most Number of Tweets | |
| 4 | Users with more number of retweets for his tweet | |
| 5 | No of Tweets for a particular player | |
| 6 | Number of Tweets created by Users per year | |
| 7 | Top Countries where Tweets came from | |
| 8 | Tweets posted on weekend and weekday | |
| 9 | Tweet count based on Language | |
| 10 | Number of Tweets by users based on month | |

**Github Links:**

1. Link for Python Code with Queries: https://github.com/Harun2703/Principles-of-Biggdata/blob/master/Phase2/Main/main/main.py
2. Link for Web application code: https://github.com/Harun2703/Principles-of-Biggdata/blob/master/Phase2/pb_page.htm
3. Link for Output queries: https://github.com/Harun2703/Principles-of-Biggdata/blob/master/Phase2/Phase-2_Query_Outputs(scala).docx
4. Link for Web application based output queries: https://github.com/Harun2703/Principles-of-Biggdata/blob/master/Phase2/Phase-2_Visualised_Outputs(Web%20application).docx
5. Link for visualized output without Web application: https://github.com/Harun2703/Principles-of-Biggdata/blob/master/Phase2/final%20pb%20phase2%20prjct.ipynb

**Conclusion:**

By doing this project we have learned to extract the data. we have also learned how to write queries to analyse data and visualize them into table and chart form.