DATA ENGINEERING PROJECT

STREAMING AND ANALYSIS OF TWITTER DATA

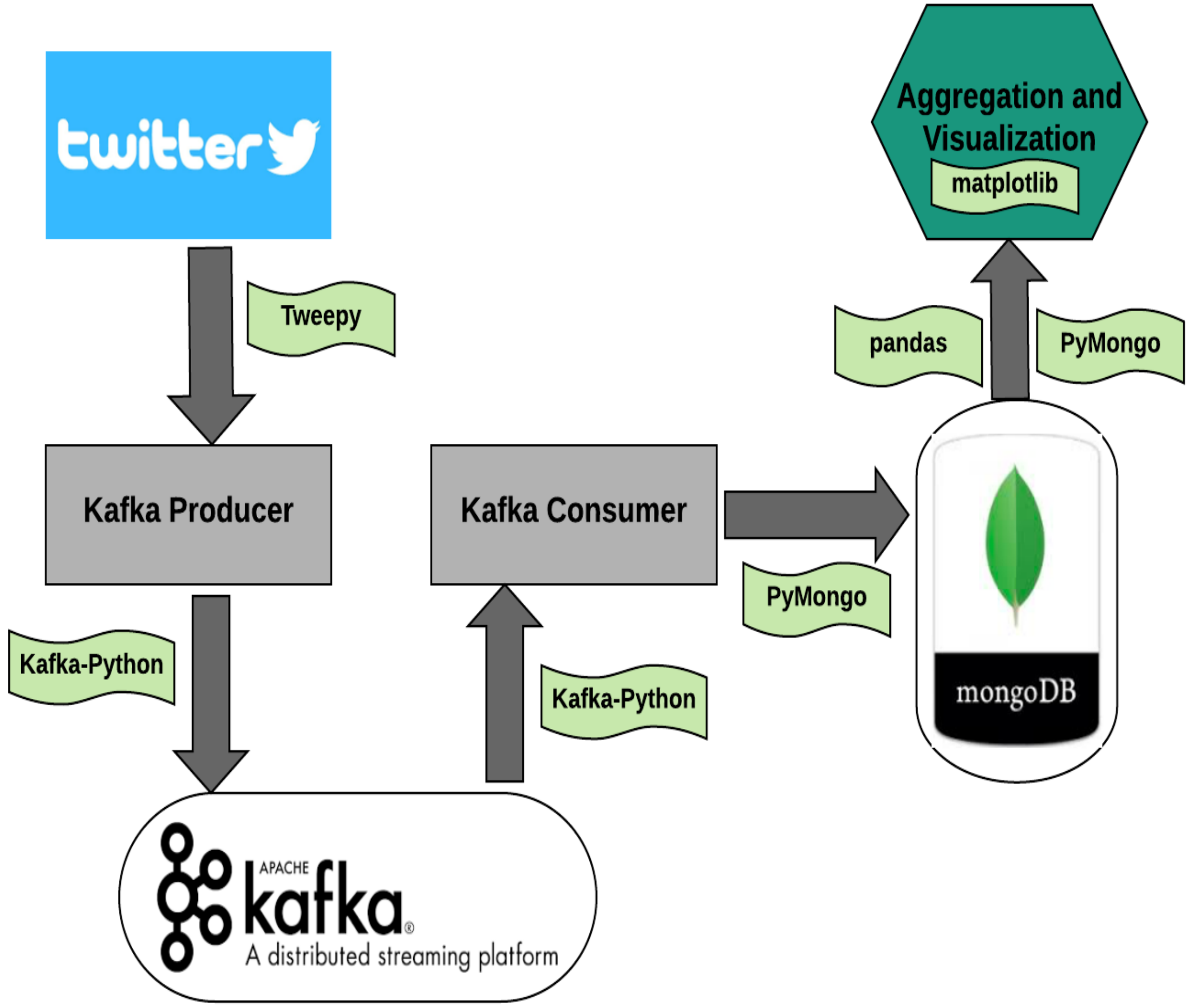
11.01.2019 **Presented By :- Kashish | Shekhar Singh | Siddarth Venkateswaran**

Introduction

This project consists of creating a pipeline for live Twitter streams for current trending topics. The main areas of focus are:

- Generating the current trending topics for a specific location by making a **Twitter API** call.
- Extracting live Twitter streams for select two trending topics.
- Making use of **Apache Kafka**, a distributed streaming platform to produce, store and consume collected tweets in the form of messages.
- Ingest the consumed messages into the database, **MongoDB**.
- Analyze 5 business questions on the data that has been ingested into the database.

Project Overview



The flowchart depicts the process that is followed in this pipeline in order to extract and analyze tweets:

1) **Live-Streaming of Tweets:** Using the Twitter API call, we first generated what the current trending topics were for a particular location. We then took the top two trending topics as keywords and collected tweets based on them for a set period of time. The producer then stored these tweets in the Kafka Broker in the form of messages to a topic name that we assigned.

2) **Consuming Messages:** All the tweets stored in a topic in the broker were sent in the form of messages to the Kafka consumer.

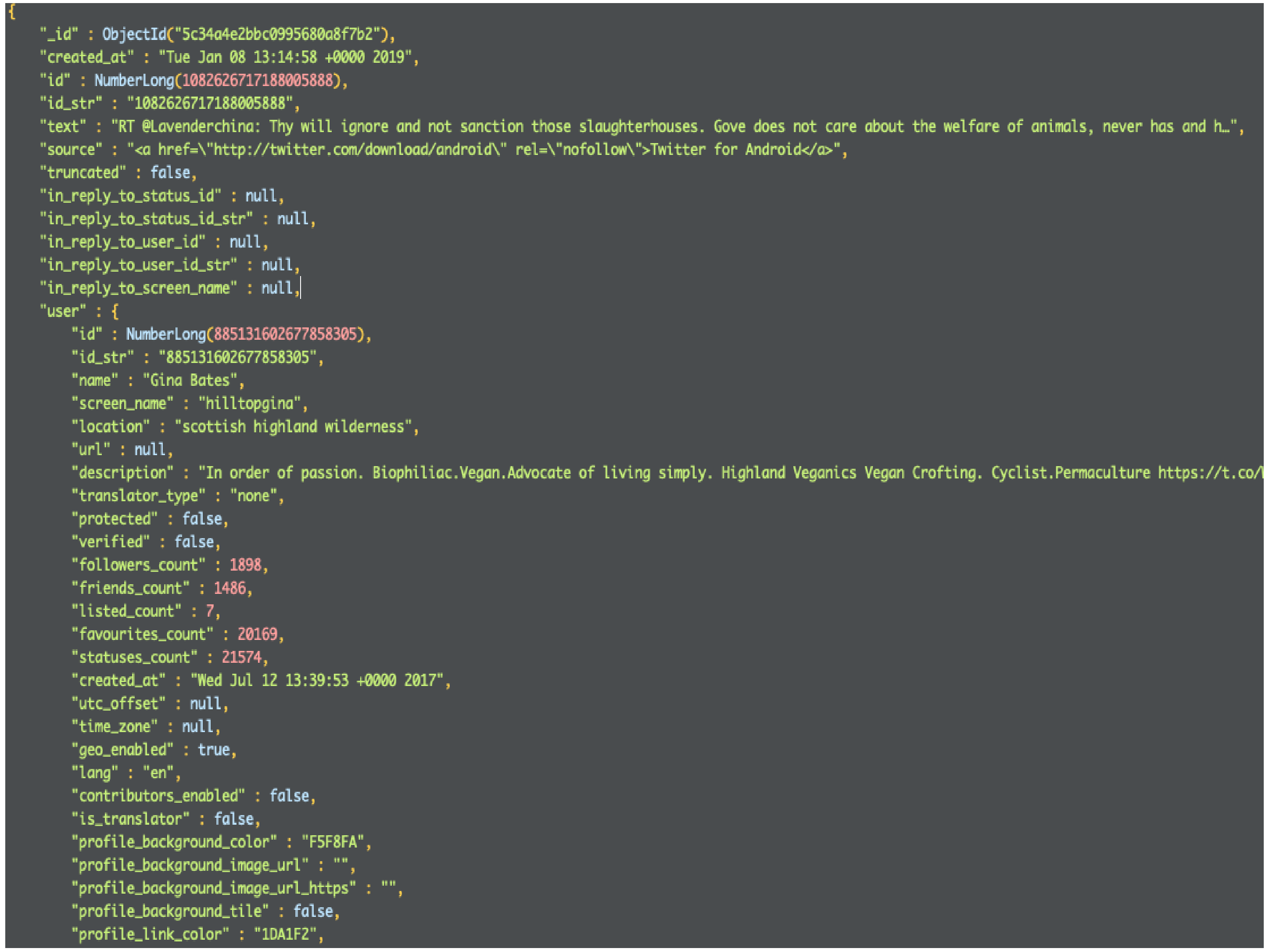
3) **Data Analysis:** Once all the tweets were consumed by the Kafka Consumer, they were then ingested into MongoDB in the JSON format. After that few queries were run in order to perform some analysis on them.

Data Source Used - Twitter

Twitter is an online news and social networking service on which people interact with each other with messages called as “Tweets”.

Features of a tweet:

- JSON format
- Tweet details
- User details
- Retweet status
- Entities



```
{
    "_id" : ObjectId("5c34a4e2bbc0995680a8f7b2"),
    "created_at" : "Tue Jan 08 13:14:58 +0000 2019",
    "id" : NumberLong(1082626717188005888),
    "id_str" : "1082626717188005888",
    "text" : "RT @Lavenderchina: Thy will ignore and not sanction those slaughterhouses. Gove does not care about the welfare of animals, never has and h…",
    "source" : "<a href=\"http://twitter.com/download/android\" rel=\"nofollow\">Twitter for Android</a>",
    "truncated" : false,
    "in_reply_to_status_id" : null,
    "in_reply_to_status_id_str" : null,
    "in_reply_to_user_id" : null,
    "in_reply_to_user_id_str" : null,
    "in_reply_to_screen_name" : null,
    "user" : {
        "id" : NumberLong(885131602677858305),
        "id_str" : "885131602677858305",
        "name" : "Gina Bates",
        "screen_name" : "hilltopgina",
        "location" : "scottish highland wilderness",
        "url" : null,
        "description" : "In order of passion. Biophiliac.Vegan.Advocate of living simply. Highland Veganics Vegan Crofting. Cyclist.Permaculture https://t.co/
        "translator_type" : "none",
        "protected" : false,
        "verified" : false,
        "followers_count" : 1898,
        "friends_count" : 1486,
        "listed_count" : 7,
        "favourites_count" : 20169,
        "statuses_count" : 21574,
        "created_at" : "Wed Jul 12 13:39:53 +0000 2017",
        "utc_offset" : null,
        "time_zone" : null,
        "geo_enabled" : true,
        "lang" : "en",
        "contributors_enabled" : false,
        "is_translator" : false,
        "profile_background_color" : "F5F8FA",
        "profile_background_image_url" : "",
        "profile_background_image_url_https" : "",
        "profile_background_tile" : false,
        "profile_link_color" : "1DA1F2",
```

Kafka Architecture

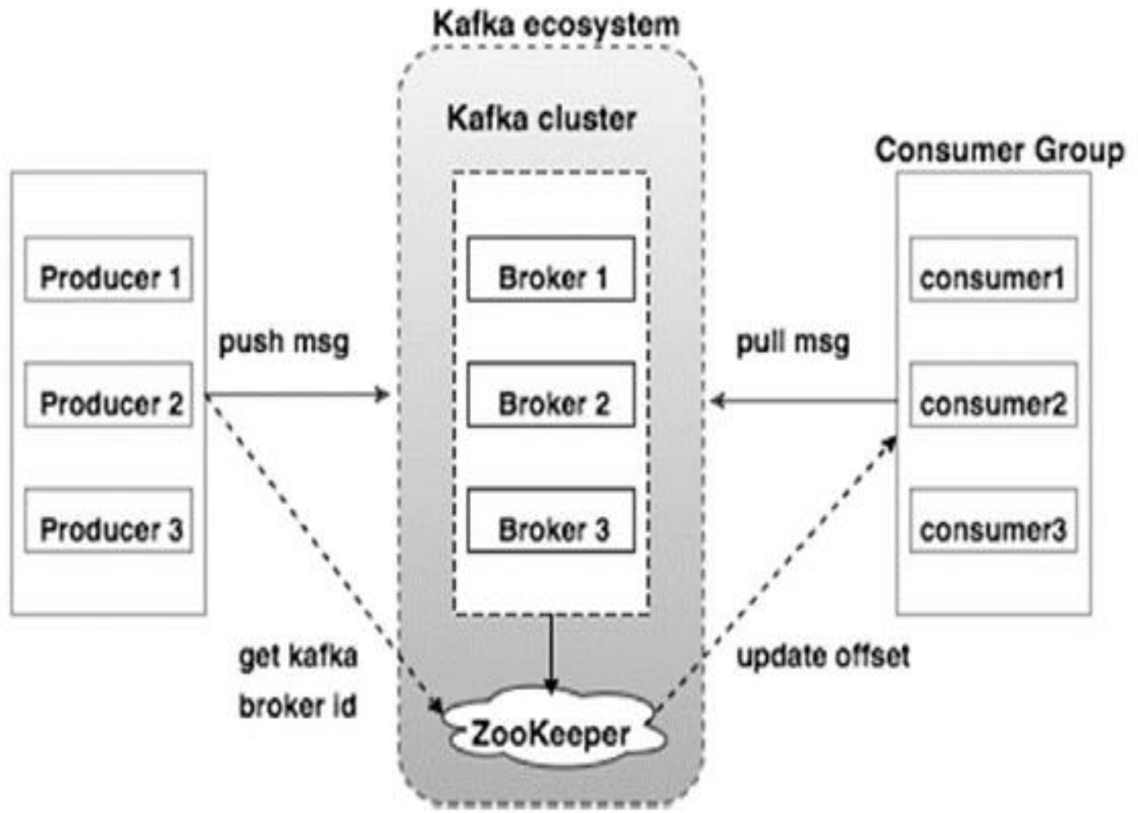
Apache Kafka is a distributed publish and subscribe messaging system which can handle large amount of data and enables the user to send messages from one point to another. A **Topic** is a feed/category name to which messages are stored and published. Kafka comprises of following components:-

1) **Broker:** Kafka cluster may consist of one or more brokers (Kafka Servers), which are basically running Kafka.

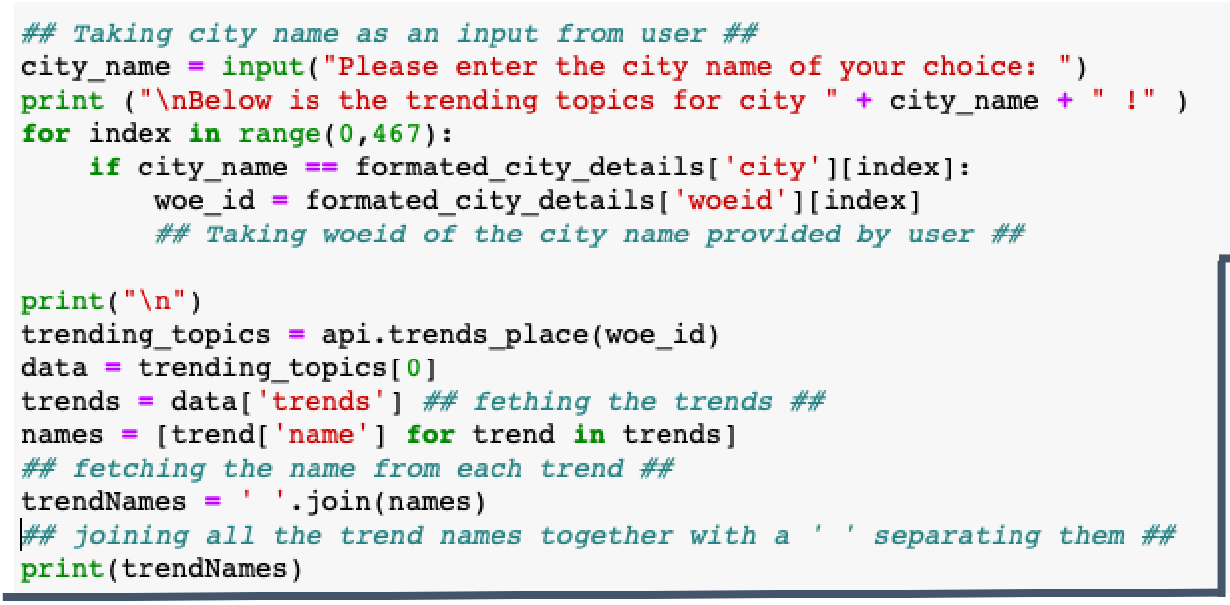
2) **ZooKeeper:** The Kafka broker is basically managed and coordinated by the ZooKeeper.

3) **Producers:** In Kafka, producers are responsible for publishing messages into one or more Kafka topics and sending it to broker.

4) **Consumers:** Consumers subscribe to one or more topics and consume published messages from the broker.

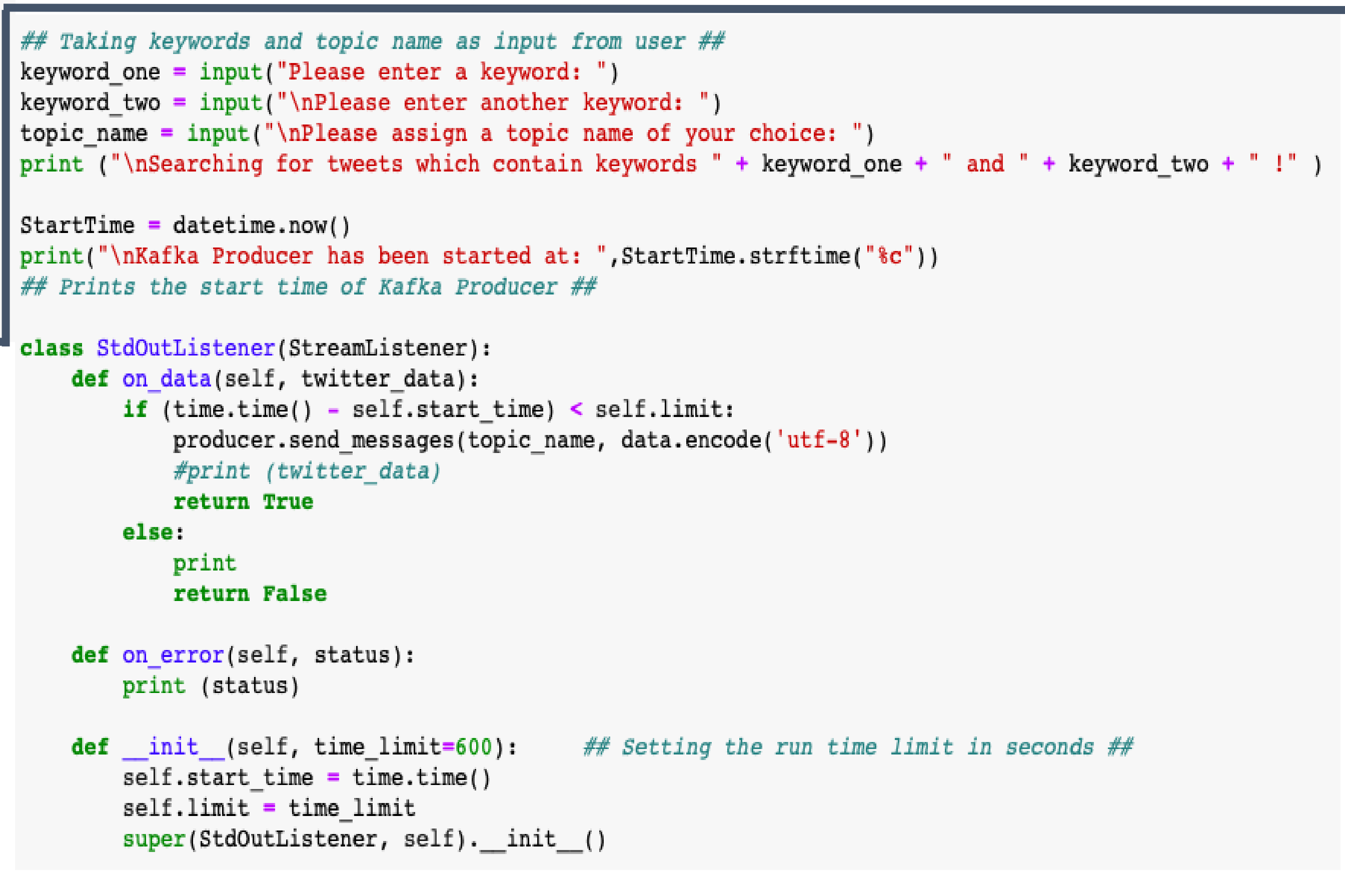


Producing Data



```python
## Taking city name as an input from user ##
city_name = input("Please enter the city name of your choice: ")
print ("\nBelow is the trending topics for city " + city_name + " !" )
for index in range(0,467):
    if city_name == formated_city_details['city'][index]:
        woe_id = formated_city_details['woeid'][index]
        ## Taking woeid of the city name provided by user ##

print("\n")
trending_topics = api.trends_place(woe_id)
data = trending_topics[0]
trends = data['trends'] ## fething the trends ##
names = [trend['name'] for trend in trends]
## fetching the name from each trend ##
trendNames = ' '.join(names)
## joining all the trend names together with a ' ' separating them ##
print(trendNames)
```

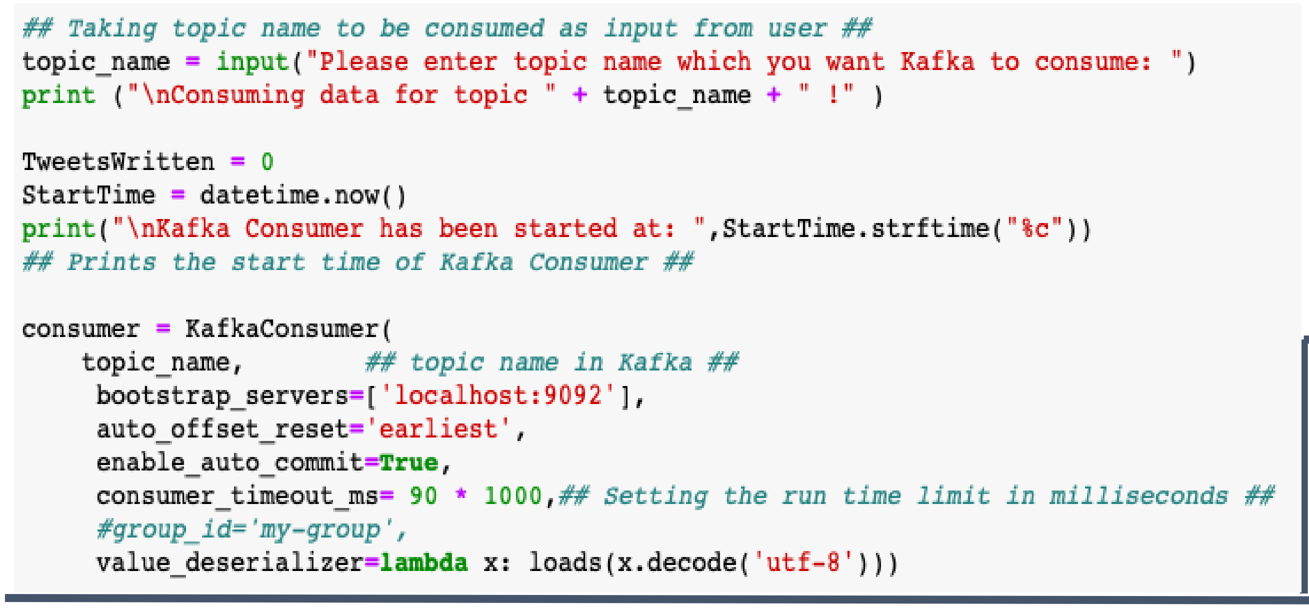
- Taking city name as input from user
- Fetching the corresponding WOEID of this city from MongoDB
- Displaying all trending topics based on this input



```python
## Taking keywords and topic name as input from user ##
keyword_one = input("Please enter a keyword: ")
keyword_two = input("\nPlease enter another keyword: ")
topic_name = input("\nPlease assign a topic name of your choice: ")
print ("\nSearching for tweets which contain keywords " + keyword_one + " and " + keyword_two + " !" )

StartTime = datetime.now()
print("\nKafka Producer has been started at: ",StartTime.strftime("%c"))
## Prints the start time of Kafka Producer ##

class StdOutListener(StreamListener):
    def on_data(self, twitter_data):
        if (time.time() - self.start_time) < self.limit:
            producer.send_messages(topic_name, data.encode('utf-8'))
            #print (twitter_data)
            return True
        else:
            print
            return False

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

    def __init__(self, time_limit=600):     ## Setting the run time limit in seconds ##
        self.start_time = time.time()
        self.limit = time_limit
        super(StdOutListener, self).__init__()
```

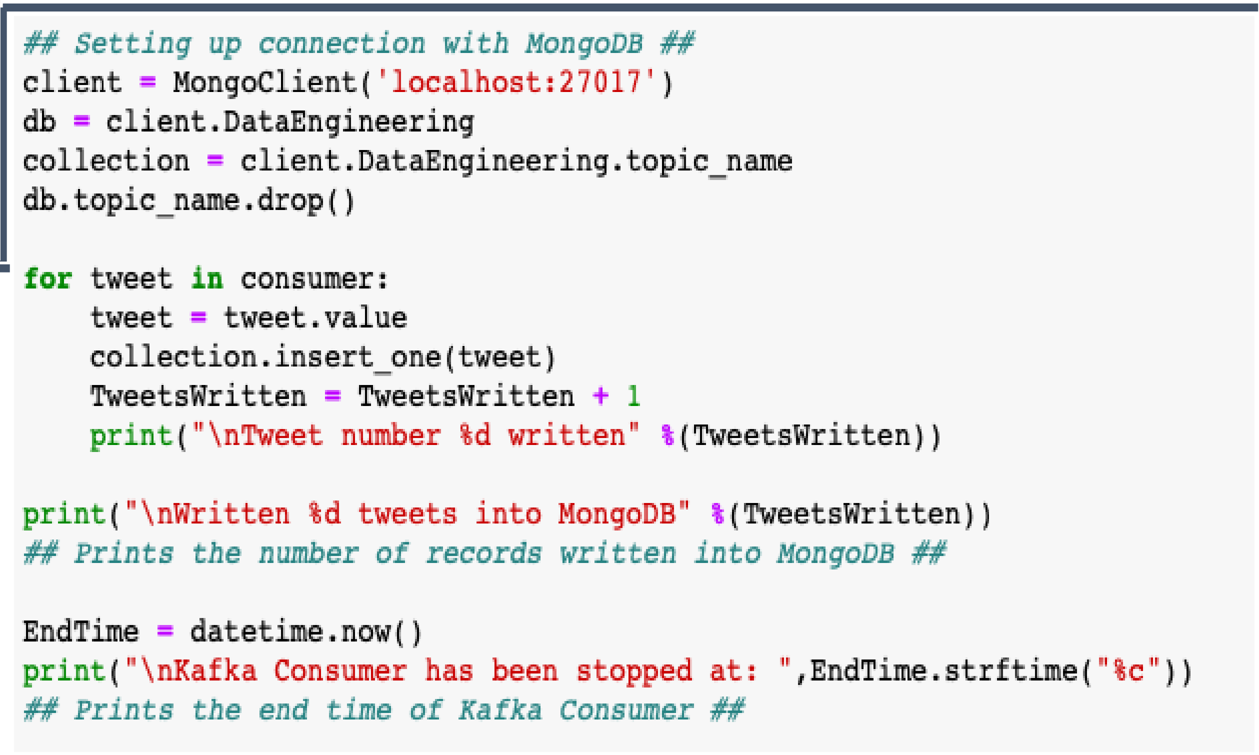
- Taking two trending topics as keyword inputs
- Assigning a topic name as per user’s choice
- Executing the Kafka Producer for a specific time period
- Fetching tweets based on the keywords provided using the stream listener
- Storing these tweets into the Kafka server in an encoded form in the user provided topic name

Consuming Data



```python
## Taking topic name to be consumed as input from user ##
topic_name = input("Please enter topic name which you want Kafka to consume: ")
print ("\nConsuming data for topic " + topic_name + " !" )

TweetsWritten = 0
StartTime = datetime.now()
print("\nKafka Consumer has been started at: ",StartTime.strftime("%c"))
## Prints the start time of Kafka Consumer ##

consumer = KafkaConsumer(
    topic_name,         ## topic name in Kafka ##
     bootstrap_servers=['localhost:9092'],
     auto_offset_reset='earliest',
     enable_auto_commit=True,
     consumer_timeout_ms= 90 * 1000,## Setting the run time limit in milliseconds ##
     #group_id='my-group',
     value_deserializer=lambda x: loads(x.decode('utf-8')))
```

- Taking topic name as input from user
- Subscribing to this topic in Kafka Consumer
- Consuming the data stored in topic after decoding it for a specific period of time.



```python
## Setting up connection with MongoDB ##
client = MongoClient('localhost:27017')
db = client.DataEngineering
collection = client.DataEngineering.topic_name
db.topic_name.drop()

for tweet in consumer:
    tweet = tweet.value
    collection.insert_one(tweet)
    TweetsWritten = TweetsWritten + 1
    print("\nTweet number %d written" %(TweetsWritten))

print("\nWritten %d tweets into MongoDB" %(TweetsWritten))
## Prints the number of records written into MongoDB ##

EndTime = datetime.now()
print("\nKafka Consumer has been stopped at: ",EndTime.strftime("%c"))
## Prints the end time of Kafka Consumer ##
```

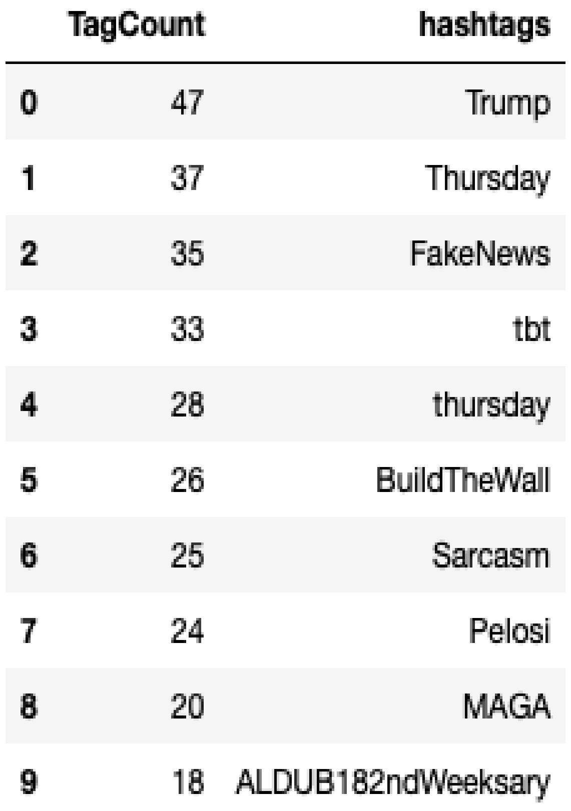
- Setting up connection with MongoDB
- Ingesting tweets into MongoDB simultaneously as they get consumed by the consumer into the collection name called "topic_name" in the "DataEngineering" database

Libraries Used

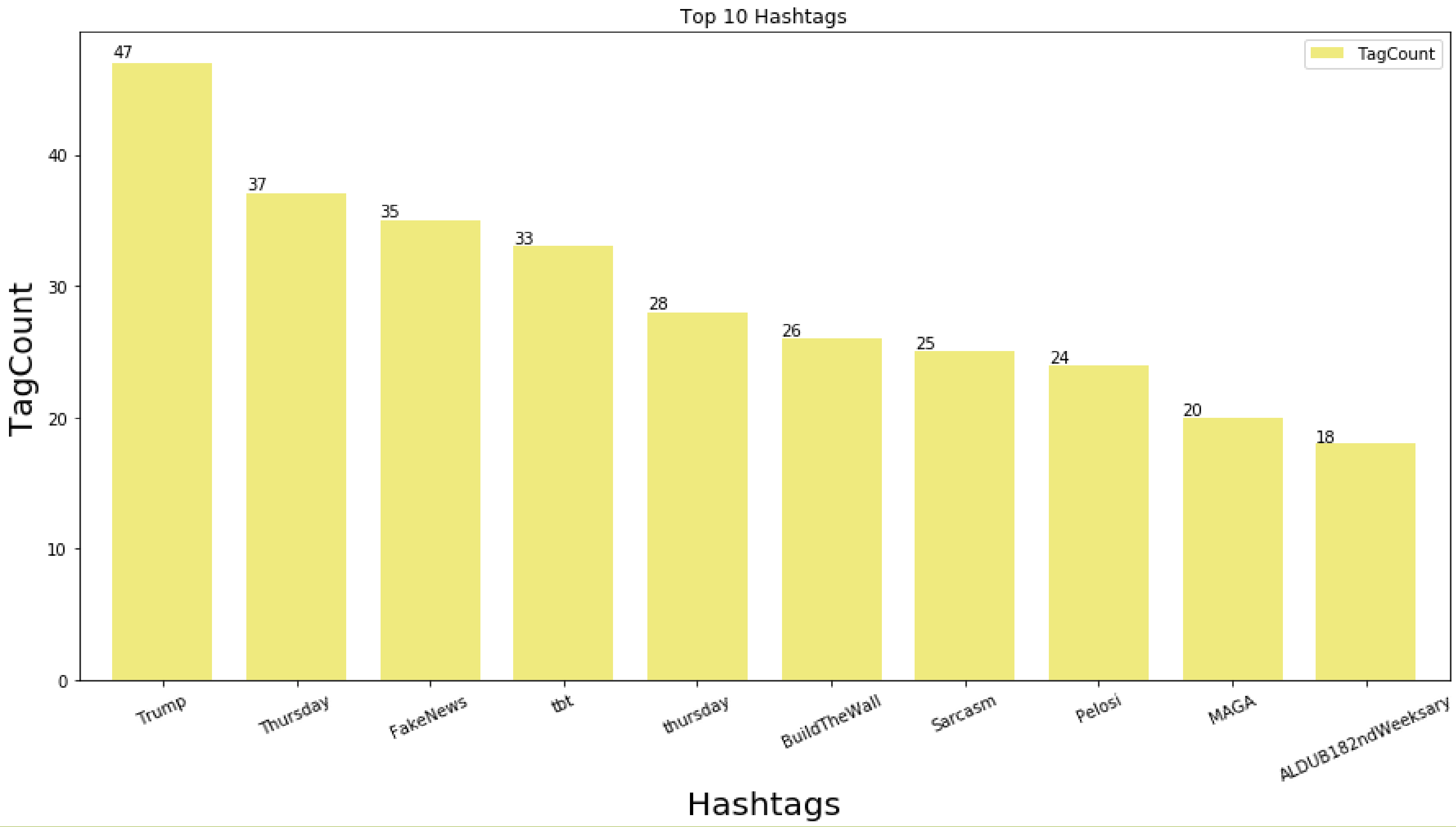
1) **Tweepy:** It is a Python library used to access the Twitter API easily.
2) **Kafka-Python:** It is the Python client for the Apache Kafka stream processing system.
3) **PyMongo:** It is a Python library which contains various tools used for working with MongoDB.
4) **Pandas:** It is an open sourced library which provided easy to use data-structures and data analysis tools for Python programming.
5) **Matplotlib:** It is a Python library which is used for data visualization in a graphical format.
6) **Datetime:** It is Python package to manipulate date and times.
7) **JSON:** It is a Python built-in package which is used for encoding or decoding data in JSON format.
8) **Beautiful Soup:** It is Python package mean for scraping web data.

Top 10 Hashtags

List of top 10 hashtags used in these tweets:

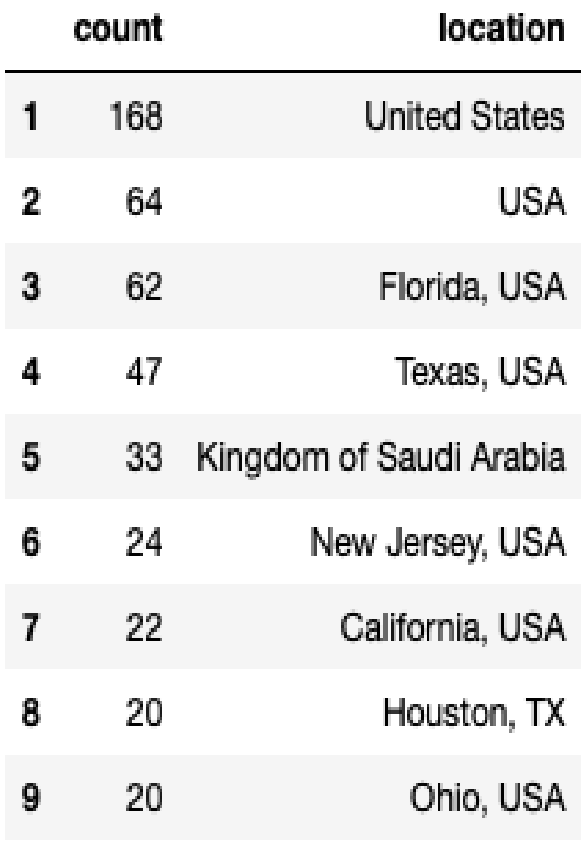


	TagCount	hashtags
0	47	Trump
1	37	Thursday
2	35	FakeNews
3	33	tbt
4	28	thursday
5	26	BuildTheWall
6	25	Sarcasm
7	24	Pelosi
8	20	MAGA
9	18	ALDUB182ndWeeksary

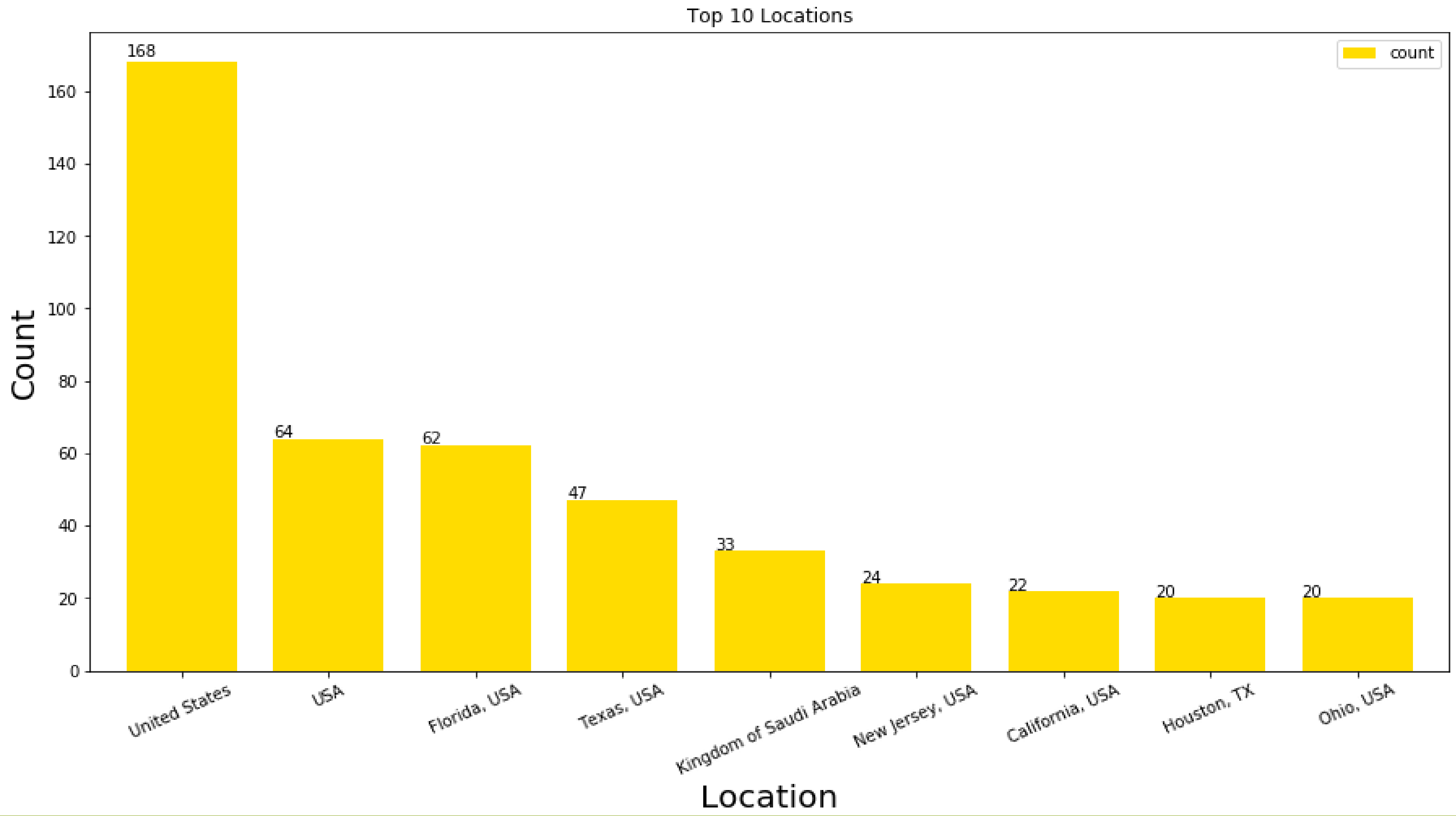


Top 10 Locations

List of top 10 locations that these tweets have been tweeted from:

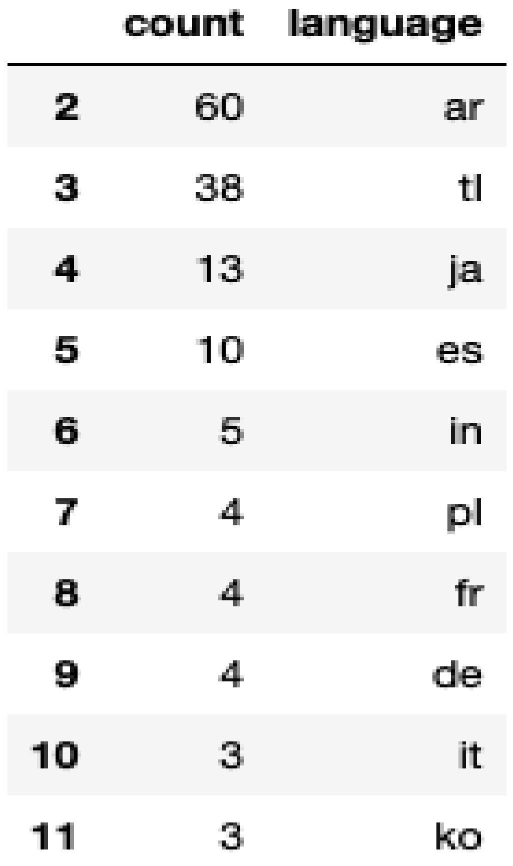


	count	location
1	168	United States
2	64	USA
3	62	Florida, USA
4	47	Texas, USA
5	33	Kingdom of Saudi Arabia
6	24	New Jersey, USA
7	22	California, USA
8	20	Houston, TX
9	20	Ohio, USA

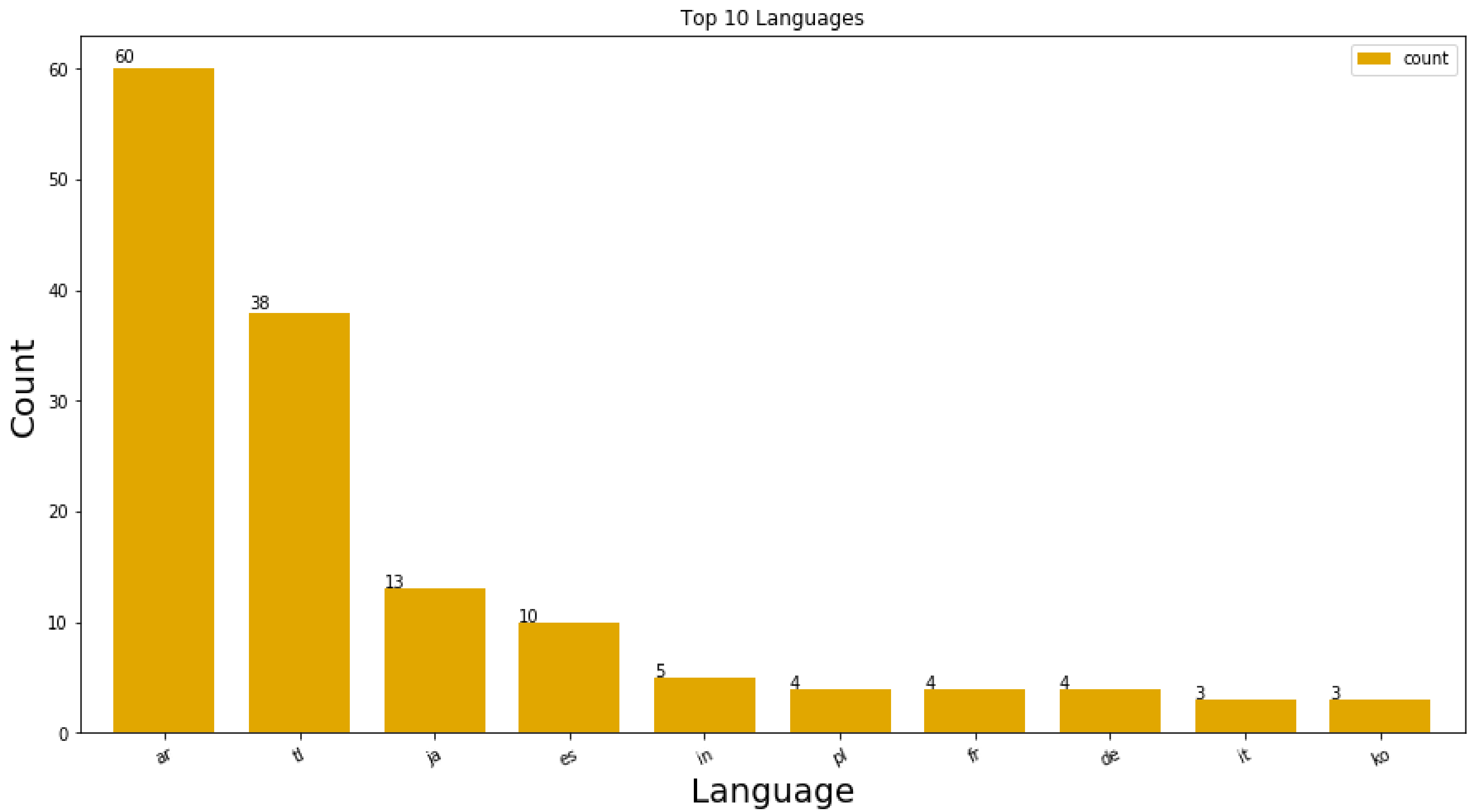


Top 10 Languages

List of top 10 languages used in these tweets apart from English and Undefined:

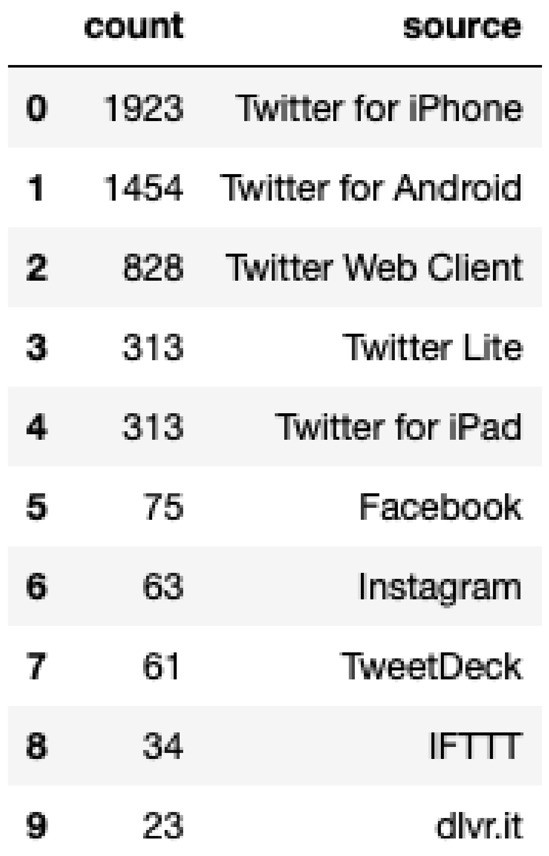


	count	language
2	60	ar
3	38	tl
4	13	ja
5	10	es
6	5	in
7	4	pl
8	4	fr
9	4	de
10	3	it
11	3	ko

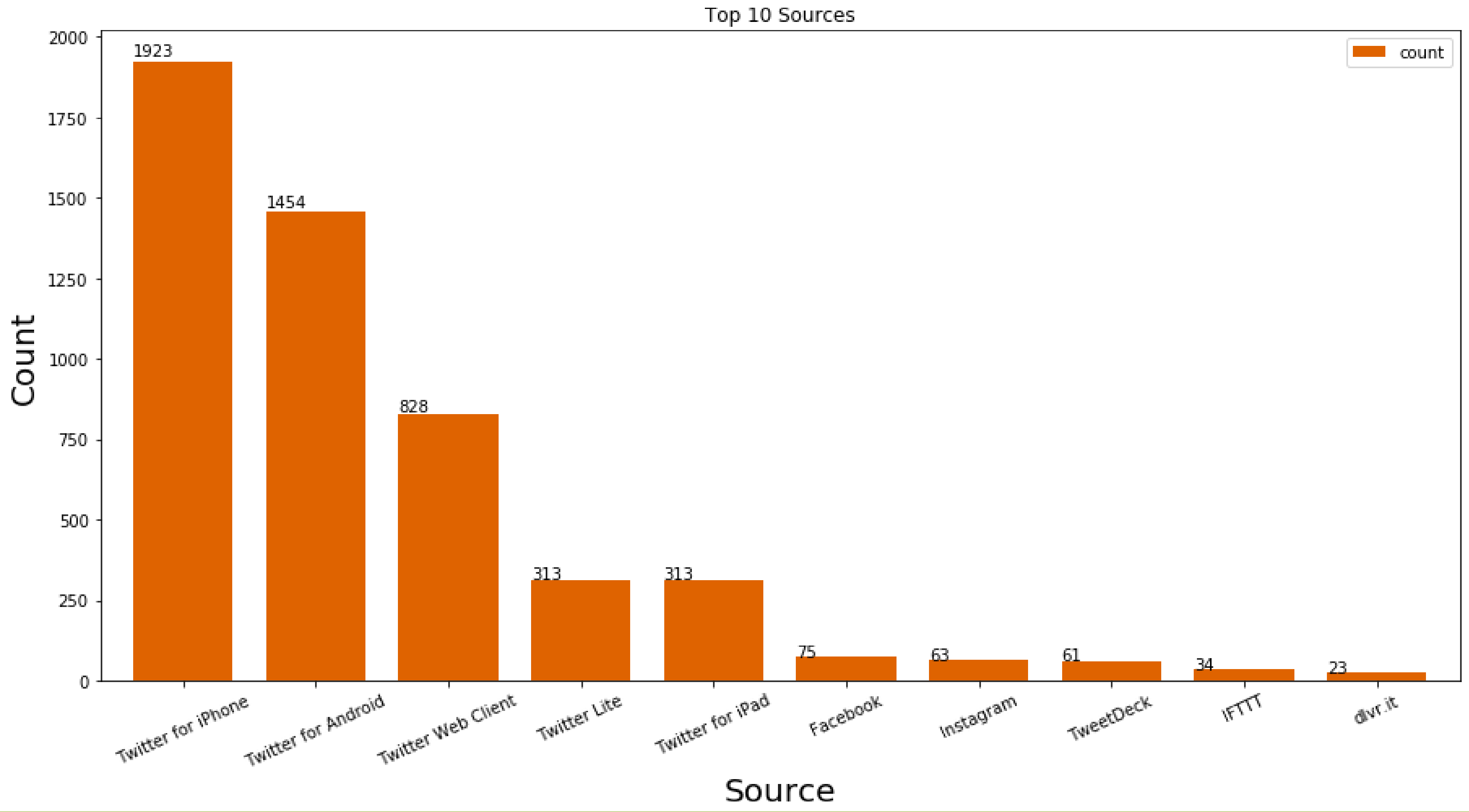


Top 10 Sources

List of top 10 sources that these tweets have been tweeted from:

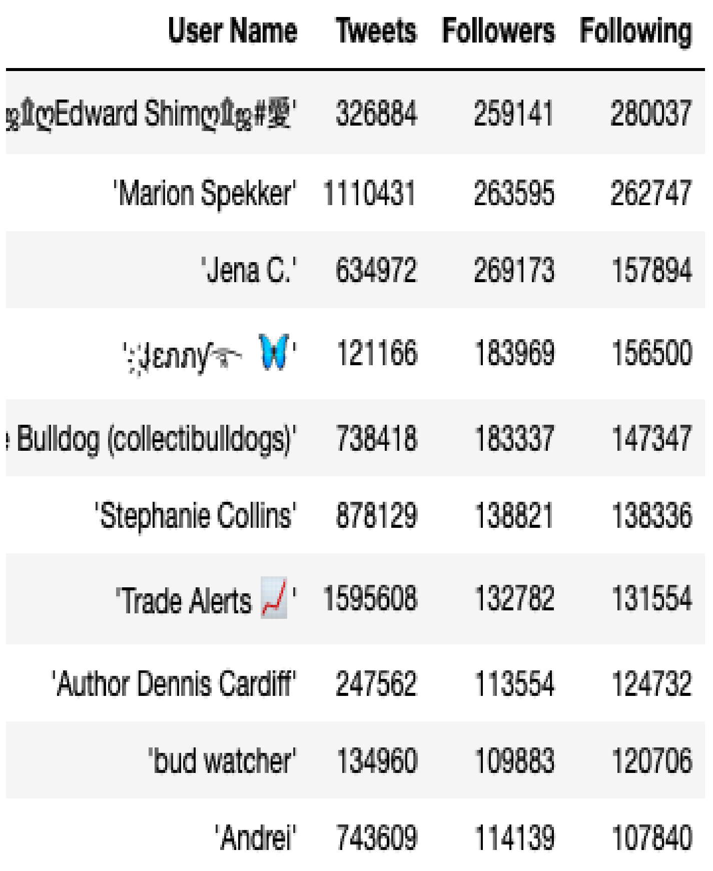


	count	source
0	1923	Twitter for iPhone
1	1454	Twitter for Android
2	828	Twitter Web Client
3	313	Twitter Lite
4	313	Twitter for iPad
5	75	Facebook
6	63	Instagram
7	61	TweetDeck
8	34	IFTTT
9	23	dlvr.it

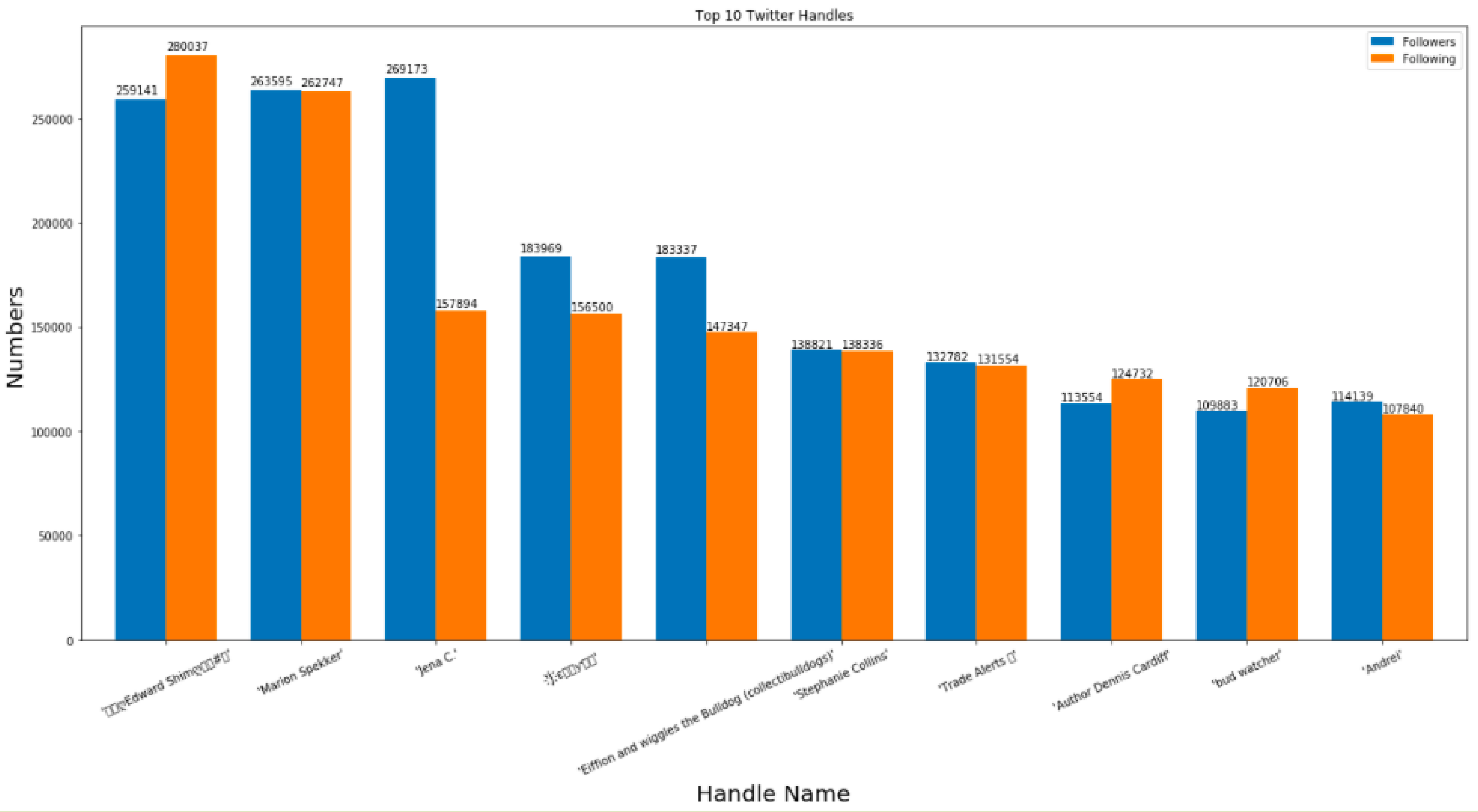


Top 10 Twitter Handles

List of top 10 Tweet generators and their popularity levels:



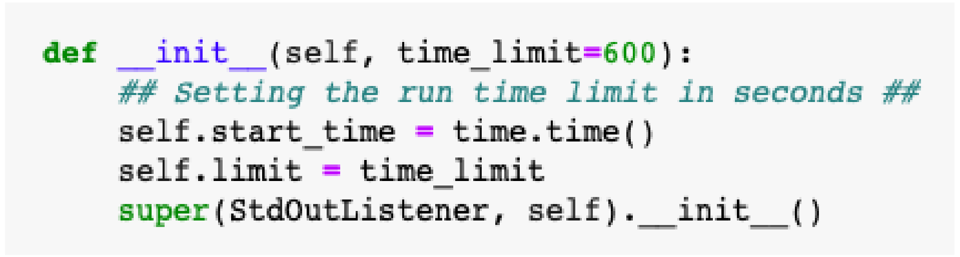
User Name	Tweets	Followers	Following
ஐ۩ღEdward Shimღ۩ஐ#愛'	326884	259141	280037
'Marion Spekker'	1110431	263595	262747
'Jena C.'	634972	269173	157894
'҉Jɛnny☜ 🦋'	121166	183969	156500
Bulldog (collectibulldogs)'	738418	183337	147347
'Stephanie Collins'	878129	138821	138336
'Trade Alerts 📈'	1595608	132782	131554
'Author Dennis Cardiff'	247562	113554	124732
'bud watcher'	134960	109883	120706
'Andrei'	743609	114139	107840



Challenges

1. **Limiting the number of tweets produced:**

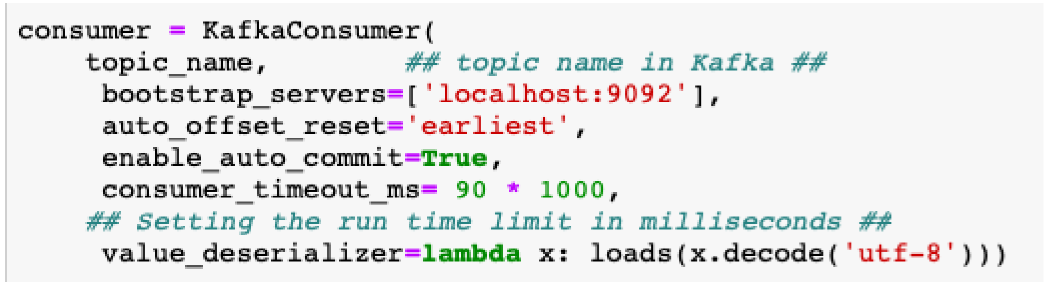
- Some topics have the capability to generate huge amount of tweets within a short space of time.
- We implemented a time function using the Python 'Time' module, through which we extracted tweets only for a limited period.
- Here, we've captured tweets for a topic for only 600 seconds.
- Once the time limit had elapsed, the Kafka Producer would stop producing tweets to the Kafka Broker.



```
def __init__(self, time_limit=600):
    ## Setting the run time limit in seconds ##
    self.start_time = time.time()
    self.limit = time_limit
    super(StdOutListener, self).__init__()
```

2. **Stopping the Kafka Consumer:**

- Kafka Consumer has the ability to consume huge number of tweets in a short span of time.
- For instance, when we produced tweets for only 600 seconds, these were getting consumed and ingested by the Kafka Consumer within few seconds.
- The challenge here lied in stopping the Kafka Consumer post consumption and ingestion.
- We implemented a consumer_timeout argument using which the consumer would consume tweets only for a limited period of time and would then stop running.



```
consumer = KafkaConsumer(
    topic_name,           ## topic name in Kafka ##
     bootstrap_servers=['localhost:9092'],
     auto_offset_reset='earliest',
     enable_auto_commit=True,
     consumer_timeout_ms= 90 * 1000,
    ## Setting the run time limit in milliseconds ##
     value_deserializer=lambda x: loads(x.decode('utf-8')))
```

References

1) For Kafka installation and testing on the macbook:
   https://medium.com/@Ankitthakur/apache-kafka-installation-on-mac-using-homebrew-a367cdefd273
2) For understanding the importance of Zookeeper for Kafka:
   https://stackoverflow.com/questions/23751708/is-zookeeper-a-must-for-kafka
3) Garnering WOEID data:
   https://codebeautify.org/jsonviewer/f83352
4) Getting the Twitter Trends using Tweepy:
   https://stackoverflow.com/questions/21203260/python-get-twitter-trends-in-tweepy-and-parse-json
5) For making a Twitter API call and building the Kafka Producer:
   https://www.bmc.com/blogs/working-streaming-twitter-data-using-kafka/
6) For getting an idea on performing Twitter Analysis:
   https://www.infoworld.com/article/2608083/application-development/do-twitter-analysis-the-easy-way-with-mongodb.html?page=3
7) For building the Kafka Consumer and ingesting the tweets on to the database:
   https://towardsdatascience.com/kafka-python-explained-in-10-lines-of-code-800e3e07dad1
8) For Understanding the nitty-gritties of Tweepy:
   https://www.youtube.com/watch?v=wlnx-7cm4Gg&t=38s
9) Installing Python Libraries:
   https://pypi.org/

THANK YOU