CASE STUDY REPORT

1. Problem Statement

The Twitter Producer should get data from Twitter based on some keywords and put them in a Kafka topic. The Consumer component should get data from your twitter Kafka topic using either elastic search consumer or spark stream or kafka stream and insert it into Mongo/ElasticSearch after doing some basic transformations .

To write an API service(Java,Scala,Python, your choice) which can be queried using any tool like Postman etc to give the following answers.

Need to implement following functionalities (Minimum of 3) :

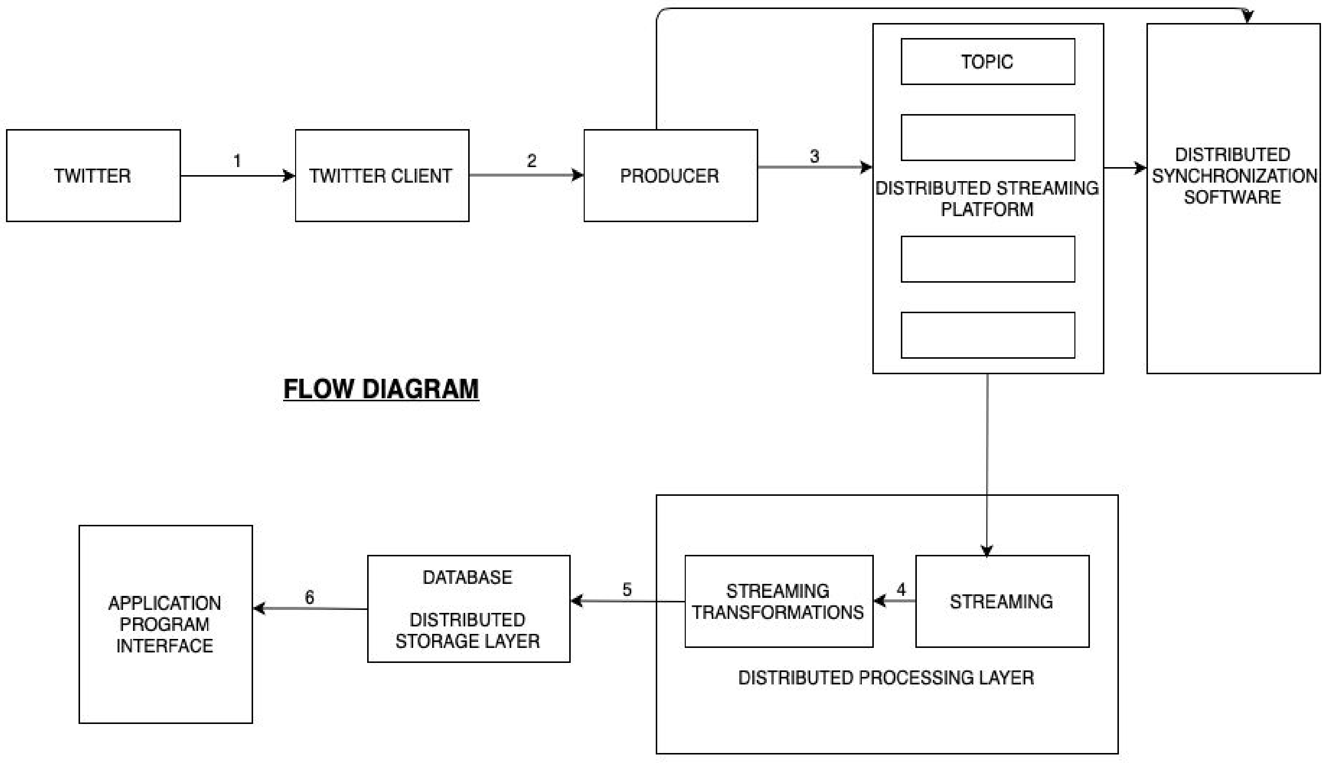
1. **Need to find overall number of tweets on coronavirus (keywords can be virus/covid-19/corona etc
   etc) per country in the last 3 months.**
2. **Next, need to find overall number of tweets per country on a daily basis.**
3. **Also need to find the top 100 words occurring on tweets involving coronavirus. (words should be nouns/verbs and not involving common ones like the, is, are, etc etc).**
4. **Find the top 100 words occurring on tweets involving coronavirus on a per country basis.**
5. Top 10 preventive / precautionary measures suggested by WHO worldwide /country wise.
6. **Total no. of donations received towards COVID 19 country wise, in all the affected countries.**
7. Ranking of impacted countries over the last 2 month on a week basis, to see who was standing where at any given week.

2. Technical Requirements

Twitter developer account - For the twitter data.
Twitter Client - To fetch the data from twitter.
Kafka Producer - To store the data in the topic of distributed streaming platforms.
Streaming - To get the data, stream it, transform it in the distributed processing layer.

Database - to store all the data in a distributed storage layer.
Rest API - to fetch the required data from the database.

3. Flow diagram


TWITTER
1
TWITTER CLIENT
2
PRODUCER
3
TOPIC
DISTRIBUTED STREAMING PLATFORM
DISTRIBUTED SYNCHRONIZATION SOFTWARE
FLOW DIAGRAM
APPLICATION PROGRAM INTERFACE
6
DATABASE
DISTRIBUTED STORAGE LAYER
5
STREAMING TRANSFORMATIONS
4
STREAMING
DISTRIBUTED PROCESSING LAYER

1. TWITTER CLIENT GETTING THE INFORMATION (TWEETS)
2. FETCHING TWITTER DATA
3. DISTRIBUTED SYNCHRONISATION SOFTWARE RUNNING IN THE BACKGROUND (ZOOKEEPER IN THIS CASE)
4. WRITING DATA TO A DISTRIBUTED STREAMING PLATFORM (KAFKA IN THIS CASE)
5. STREAM AND TRANSFORMATIONS (SPARKSQL IN THIS CASE)

6. STORING DATA IN THE DATABASE (MONGODB IN THIS CASE)
7. TO ACCESS THE DATABASE AND GET THE DESIRED OUTPUT USING API

4. Tech stack

Twitter Streaming API:
Twitter4J is an unofficial Java library for the Twitter API. Using this library, we can stream real-time tweets, and filter to only return Tweet objects streamed from this API in JSON format.

Apache Kafka:
Apache Kafka is a distributed streaming platform. Kafka is used in this application as a publish-subscribe messaging system to reliably get data between pipeline components. Tweets are published to a *topic* in the Kafka brokers, where Kafka manages their partitions and order, and eventually consumed by a subscriber.
Broker: Kafka is run as a cluster on one or more servers that can span multiple datacenters. Those servers are usually called brokers.

Apache Spark Stream:
Stream processing can be done using Kafka streaming or Spark streaming technology. Spark Streaming using Structured streaming is preferred. The Spark SQL engine will take care of running it incrementally and continuously updating the final result as streaming data continues to arrive.
As the data passes through Spark, we simply filtered each record to only keep the payload (Tweet data), and discarded any Kafka metadata. Our focus when doing this project was on the preprocessing/transformations/analytics of the streaming data can be defined in this component.

MongoDB:

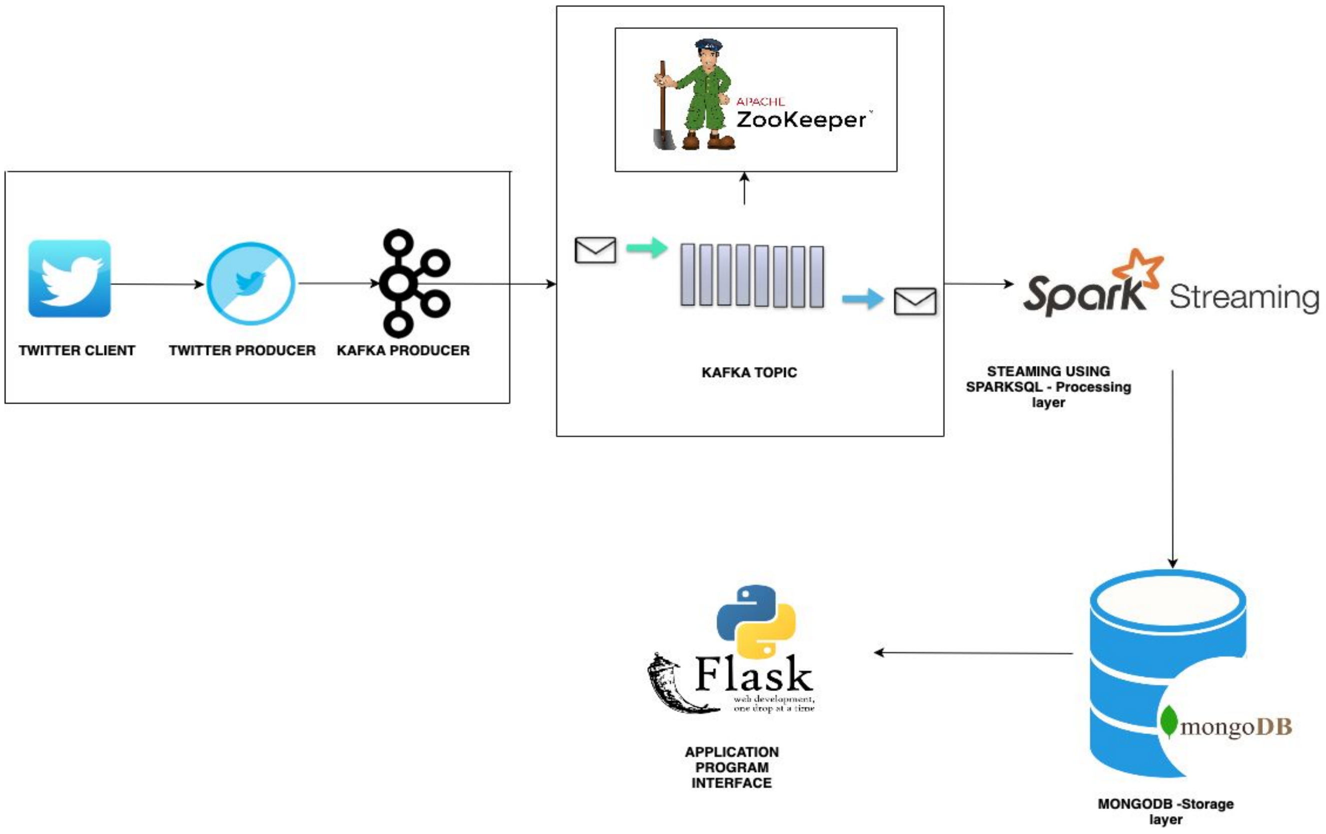
As a database, we can use mongoDB, elasticsearch and Cassandra.
When it comes to ingesting the data, elasticsearch has lower write throughput. Cassandra has the highest write throughput and MongoDB is better than elasticsearch. Thus we do not use Elasticsearch as our database.
Also as the number of queries increase dealing with Cassandra becomes tedious - for serving every new database query a new table has to be created. Therefore we prefer to use mongoDB as the database.
MongoDB was the choice of storage because of the JSON format of Tweets. Not all tweets have the same fields. Also it has the capability to do deep queries and can scale easily.

Flask:

REST stands for REpresentational State Transfer and is an architectural style used in modern web development. It defines a set or rules/constraints for a web application to send and receive data.

We have used Flask API as a drop-in replacement for Flask that provides an implementation of browsable APIs similar to what Django REST framework provides. It gives you properly content negotiated-responses and smart request parsing.

5. Architecture Diagram



6. Proposed Solution:

Overview:
A data pipeline to stream live Tweets through a message broker (Kafka), a processing engine (Spark), storing in a database (MongoDB) and finally a querying API (Flask).

Using a Twitter client and Kafka producer, we plan to fetch the data from the twitter based on keywords given into a kafka topic.
Next, we will use a read stream to fetch the data from the kafka topic and then perform some transformations to filter the data and put it into the database after retrieving the necessary fields from the tweet.

Stream processing can be done using Kafka streaming or Spark streaming technology. SparkStreaming using Structured streaming is preferred. The Spark SQL engine will take care of running it incrementally and continuously updating the final result as streaming data continues to arrive.

As the data passes through Spark, we simply filtered each record to only keep the payload (Tweet data), and discarded any Kafka metadata. Our focus when doing this project was on the preprocessing/transformations/analytics of the streaming data can be defined in this component.

As a database, we can use mongoDB, elasticsearch and Cassandra.

When it comes to ingesting the data, elasticsearch has lower write throughput. Cassandra has the highest write throughput and MongoDB is better than elasticsearch. Thus we do not use Elasticsearch as our database.
Also as the number of queries increase dealing with Cassandra becomes tedious - for serving every new database query a new table has to be created. Therefore we prefer to use mongoDB as the database.
MongoDB was the choice of storage because of the JSON format of Tweets. Not all tweets have the same fields. Also it has the capability to do deep queries and can scale easily.

During our Spark Streaming, we formatted the tweets to get three fields that would solve our problem statement: country, date and tweet text.

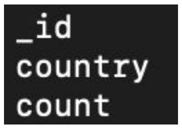
- ➔ We have observed that users who have turned their geo-location, have null in their country field. We have also seen weird invalid characters in the country field. So, the extracted country is looked up in a set of predefined countries.
- ➔ The extracted tweeted text is filtered using a set of stopwords.
- ➔ The extracted date is formatted as mmm dd, yyyy.

These three data were inserted into 4 collections and an update of count is maintained. This is done to make the query faster.

We plan to perform the required queries through a Python built API service using Flask.

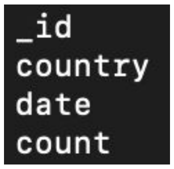
Schema of MongoDB Collections:

Collection1:



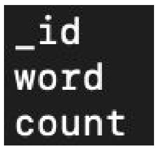
```
_id
country
count
```

Collection2:



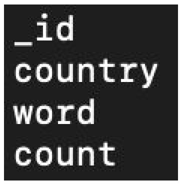
```
_id
country
date
count
```

Collection3:



```
_id
word
count
```

Collection4:



```
_id
country
word
count
```

API Schema:

- ➔ localhost:5000/tweets_per_country : Finding count of overall tweets per country.
- ➔ localhost:5000/tweets_per_country/<country_name> : Finding count of overall tweets per country for a particular searched country.
- ➔ localhost:5000/tweets_per_country_date : Finding overall count of tweets per country on a daily basis.
- ➔ localhost:5000/tweets_per_country_date/<country_name> : Finding count of overall tweets per country on a daily basis for a particular searched country.
- ➔ localhost:5000/tweets_per_country_date/date/<date> : Finding count of overall tweets per country on a daily basis for a particular searched date.
- ➔ localhost:5000/tweets_per_country_date/date_country/<country_name>/<date> : Finding count of overall tweets per country on a daily basis for a particular searched date and country.
- ➔ localhost:5000/overall_top_100words : Finding the top 100 words occurring on tweets.
- ➔ localhost:5000/top_100words_country: Finding top 100 words per country.
- ➔ localhost:5000/top_100words_country/<country_name>: Finding top 100 words per country for a particular searched country.
- ➔ localhost:5000/overall_donations_country: Finding total number of donations per country.
- ➔ localhost:5000/overall_donations_country/<country_name>: Finding total number of donations per country for a particular searched country.

CORNER CASES HANDLED IN THE API

- ➔ When a wrong URL is entered which is not found , we have handled that “Page Not found-404” error with an error handler that displays a default message.
- ➔ When a database connection fails, we have handled “Database Connection Failure - 500” with another error handler and a default message.
- ➔ For the questions we have solved we have given an additional option to enter a country name as an input in the URL.
- ➔ We have handled the issue of lowercase and uppercase letters, while entering the country name. So now the country name can be entered in any format.
- ➔ For the countries whose name is more than a single word, we have handled that case and given an option to enter such names along with an underscore in between consecutive words.
- ➔ Also for date, the month, date and year are to be entered in order and with underscores in between.

7. MongoDB Queries:

```
client = MongoClient("mongodb://localhost:27017")

'''Establishing a DB and collection instance'''
db=client.DB
tasks_collection1 = db.CountryAndCount
tasks_collection2= db.CountryCountAndDate
tasks_collection3 = db.WordCount
tasks_collection4 = db.WordCountCountry
```

1) **Need to find overall number of tweets on coronavirus (keywords can be virus/covid-19/corona etc) per country.**

Query :

```
cursor = tasks_collection1.aggregate([
         {
              '$sort' : {'count' : -1}
         }
     ],allowDiskUse=True)
```

Output:

```
{
    "count": 22042,
    "country": "India"
  },
  {
    "count": 8625,
    "country": "Indonesia"
  },
  {
    "count": 7578,
    "country": "Nigeria"
  },
  {
    "count": 6595,
    "country": "England"
  },
  {
    "count": 5907,
    "country": "United States"
  }
```

2) **Need to find overall number of tweets on coronavirus (keywords can be virus/covid-19/corona etc) per country where a country is given as input**

Query :

```
'''country fix'''
    new_country=str(country_name).replace("_"," ").title()

    cursor = tasks_collection1.aggregate([
        {
            '$match':{
                'country':new_country
            }
        }
    ],allowDiskUse=True)
```

Output :

```
[
  {
    "count": 22042,
    "country": "India"
  }
]
```

3) **Need to find overall number of tweets per country on a daily basis**

Query:

```
cursor1 = tasks_collection2.aggregate([
        {
            '$project':{
                'country':1,
                'date':1,
                'count':1
            }
        },
        {
            '$sort' : {'country' : 1}
        }
    ],
        allowDiskUse=True
    )
```

Output:

```
{
    "count": 16,
    "country": "Afghanistan",
    "date": "Apr 16, 2020"
  },
  {
    "count": 9,
    "country": "Afghanistan",
    "date": "Apr 17, 2020"
  },
  {
    "count": 6,
    "country": "Afghanistan",
    "date": "Apr 18, 2020"
  },
  {
    "count": 20,
    "country": "Afghanistan",
    "date": "Apr 14, 2020"
  }
```

4) **Need to find overall number of tweets per country on a daily basis where a country is given as input**

Query:

```
'''country fix'''
    new_country=str(country_name).replace("_"," ").title()

    cursor21 = tasks_collection2.aggregate([
        {
            '$match':{
                'country':new_country
            }
        },

        {
            '$sort' : {'count' : -1}
        }
    ],
        allowDiskUse=True
    )
```

Output:

```
{
    "count": 7403,
    "country": "India",
    "date": "Apr 18, 2020"
  },
  {
    "count": 5991,
    "country": "India",
    "date": "Apr 16, 2020"
  },
  {
    "count": 4539,
    "country": "India",
    "date": "Apr 14, 2020"
  },
  {
    "count": 4059,
    "country": "India",
    "date": "Apr 17, 2020"
  }
```

5) **Need to find overall number of tweets per country on a daily basis where a date is given as input**

Query :

```
'''date fix'''
    new_date=str(date).replace("_"," ").capitalize()
    new_date1=new_date[:6]+","+new_date[6:]
```

```
cursor21 = tasks_collection2.aggregate([
    {
        '$match':{
            'date':new_date1
        }
    },

    {
        '$sort' : {'count' : -1}
    }
],
    allowDiskUse=True
)
```

Output:

```
{
    "count": 7403,
    "country": "India",
    "date": "Apr 18, 2020"
  },
  {
    "count": 3847,
    "country": "Nigeria",
    "date": "Apr 18, 2020"
  },
  {
    "count": 3192,
    "country": "Indonesia",
    "date": "Apr 18, 2020"
  },
  {
    "count": 2737,
    "country": "United States",
    "date": "Apr 18, 2020"
  }
```

6) **Need to find overall number of tweets per country on a daily basis where date and country are given as input**

Query :

```
 '''country fix'''
    new_country=str(country_name).replace("_"," ").title()

    '''date fix'''
    new_date=str(date).replace("_"," ").capitalize()
    new_date1=new_date[:6]+","+new_date[6:]

    cursor21 = tasks_collection2.aggregate([
        {
            '$match':{
                'country':new_country,
                'date':new_date1
            }
        }
    ],
        allowDiskUse=True
    )
```

Output:

```
 {
    "count": 7403,
    "country": "India",
    "date": "Apr 18, 2020"
  }
```

7) **Need to find the top 100 words occurring on tweets involving coronavirus. (words should be nouns/verbs and not involving common ones like the, is, are, etc etc).**

Query :

```
    cursor3 = tasks_collection3.aggregate([
            {
                '$sort':{
                    'count':-1
                }
            },
            {
                '$limit':100
            }
```

```
        ],
            allowDiskUse=True
        )
```

Output :

```
{
    "count": 17886,
    "word": "virus"
  },
  {
    "count": 16848,
    "word": "corona"
  },
  {
    "count": 4853,
    "word": "people"
  },
  {
    "count": 4803,
    "word": "que"
  },
  {
    "count": 3361,
    "word": "amp"
  },
  {
    "count": 2915,
    "word": "lockdown"
  },
  {
    "count": 2908,
    "word": "coronavirus"
  },
  {
    "count": 2860,
    "word": "cases"
  },
  {
    "count": 2541,
    "word": "del"
  },
  {
    "count": 2474,
    "word": "china"
  },
  {
    "count": 2244,
    "word": "para"
  },
  {
    "count": 2161,
    "word": "new"
  },
  {
    "count": 2157,
    "word": "por"
  },
  {
    "count": 2104,
    "word": "india"
  },
  {
    "count": 2099,
    "word": "fight"
  },
  {
    "count": 2037,
    "word": "world"
  },
  {
    "count": 2031,
    "word": "one"
  }
```

8) Find the top 100 words occurring on tweets involving coronavirus on a per country basis.
Query:

```
cursor = tasks_collection4.aggregate([

        {
            '$group':{
                '_id':"$country",
                'top100_words_forThisCountry':{
                    '$push':"$word"
                }
            }
        },
        {
            '$project':{
                'country':1,
                'top100_words_forThisCountry':{
                    '$slice':[ "$top100_words_forThisCountry", 100 ]
                }
            }
        },
        {
            '$sort':{
                '_id':1
            }
        }

    ],

        allowDiskUse=True
    )
```

Output :

```
 {
    "_id": "Afghanistan",
    "top100_words_forThisCountry": [
      "test",
      "would",
      "like",
      "whether",
      "destination",
      "country",
      "requires",
      "certificates",
      "yousuf",
"aryubi",
"one",
"died",
"days",
"ago",
"brother",
"says",
"family",
"members",
"also",
"lost",
"india",
"use",
"saarc",
"emergency",
"fund",
"send",
"antimalarial",
"drug",
"afghanistan",
"please",
"download",
"books",
"children",
"circulate",
"among",
"networks",
"afghan",
"access",
"outbreakmore",
"afghans",
"returned",
"iran",
"unprecedented",
"return",
"short",
"time",
"afghanis",
"iom",
"briefed",
"member",
"states",
"orgs",
"operations",
"crisis",
"andisha",
"thanked",
"govt",
"try",
"identify",
"support",
"returness",
"become",
"threat",
"whole",
"added",
"longer",
"lasts",
"obstacles",
"free",
"flow",
"people",
"goods",
"capital",
"place",
"warm",
"qoute",
"ernest",
"hemingway",
"admist",
"krachi",
"port",
"pakistan",
"morethen",
"containers",
"traders",
"stucked",
"due",
        "jesus",
        "virus",
        "idiots",
        "symptom",
        "control",
        "doctors",
        "sorry",
        "shame",
        "gani",
        "poppet",
        "regime",
        "since",
        "beginning"
      ]
    }
```

9) **Find the top 100 words occurring on tweets involving coronavirus on a per country basis where a country is given as input.**
   Query:

```
'''country fix'''
    new_country=str(country_name).replace("_"," ").title()

    cursor = tasks_collection4.aggregate([
        {
            '$match':{
                'country':new_country
            }
        },

        {
            '$group':{
                '_id':"$country",
                'top100_words_forThisCountry':{
                    '$push':"$word"
                }
            }
        },
        {
            '$project':{
                'country':1,
                'top100_words_forThisCountry':{
                    '$slice':[ "$top100_words_forThisCountry", 100 ]
                }
            }
        },
        {
            '$sort':{
                '_id':1
            }
        }

    ],

        allowDiskUse=True
    )
```

Output:

```
 {
    "_id": "United States",
    "top100_words_forThisCountry": [
      "given",
      "fact",
      "china",
      "moved",
      "quickly",
      "shut",
      "travel",
      "domestically",
      "wuhan",
      "rest",
      "suggested",
      "responsible",
      "coronavirus",
      "pandemic",
      "excoriated",
      "analysis",
      "global",
      "virus",
      "response",
      "stymied",
      "lack",
      "leadership",
"extraordinary",
"time",
"requires",
"shared",
"resilience",
"partnership",
"committed",
"bringing",
"highperformanc",
"trump",
"knows",
"concocted",
"level",
"lab",
"told",
"everyone",
"blunt",
"truth",
"ccp",
"infected",
"people",
"good",
"defund",
"mohammed",
"hamayda",
"healthcare",
"worker",
"beaumont",
"hospital",
"dearborn",
"treating",
"patients",
"science",
"politics",
"merkel",
"offers",
"cautious",
"reentry",
"katrin",
"bennhold",
"via",
"nyt",
"heck",
"real",
"truthsand",
"furthermore",
"think",
"rush",
"open",
"ame",
"call",
"chinese",
"youre",
"racist",
"socialism",
"bullshit",
"still",
"going",
"saying",
"staying",
"fuck",
"home",
"wreaking",
"havoc",
"ability",
"make",
"thingsincluding",
"vaccines",
"need",
"prayers",
"mission",
"also",
"challenge",
"liveswe",
"strong",
"asked",
"reports",
"may"
```

```
    ]
  }
```

10) **Total no. of donations received towards COVID 19 country wise, in all the affected countries.**
Query:

```
cursor5 = tasks_collection4.aggregate([
        {
            '$match':{
                "word":{
                    '$in':[ "donated","donate","donation","contribute"]
                }
            }
        },
        {
            '$project':{
                'country':1,
                'count':1
            }
        },
        {
            '$group':{
                '_id':'$country',
                'count_per_country':{'$sum':'$count'}
            }
        },
        {
            '$sort':{
                '_id':1
            }
        }

    ],

        allowDiskUse=True
    )
```

Output:

```
{
    "_id": "Afghanistan",
    "count_per_country": 1
  },
  {
    "_id": "Algeria",
    "count_per_country": 1
  },
  {
    "_id": "Antarctica",
    "count_per_country": 1
  },
  {
    "_id": "Argentina",
    "count_per_country": 3
  },
  {
    "_id": "Australia",
    "count_per_country": 6
  },
  {
    "_id": "Austria",
    "count_per_country": 1
  }
```

11) **Total no. of donations received in a particular country towards COVID 19 where a country is given as input.**
    Query:

```
'''country fix'''
    new_country=str(country_name).replace("_"," ").title()
    cursor5 = tasks_collection4.aggregate([
        {
            '$match':{
                'country':new_country
            }
        },
        {
            '$match':{
                "word":{
                    '$in':[ "donated","donate","donation","contribute"]
                }
            }
        },
        {
            '$project':{
                'country':1,
                'count':1
            }
        },
            {
                '$group':{
                    '_id':'$country',
                    'count_per_country':{'$sum':'$count'}
                }
            },

        ],

            allowDiskUse=True
        )
```

Output:

```
{
   "_id": "India",
   "count_per_country": 236
 }
```

8. Test Cases

Twitter producer : For this module we have verified if the credentials passed in the code are the same as given by us. In case of any change, we will get to know about the invalid access of twitter.

Spark Streaming : This is the module which contains the business logic involving the formatting of tweets. We have test cases wherein we pass a sample tweets to check if the formatting is occurring properly.

Flask API : For this module we have verified if the entered URL is in the proper format for all the queries.

9. Challenges

We need to study various kinds of transformation APIs to get our required data from the database. (https://www.elastic.co/guide/en/elasticsearch/reference/current/put-transform.html)

The documentation of dev.twitter.com suggests that we have access to only 7 days of past tweets. In that case, 3 month old tweets are challenging to retrieve.