Twitter Data Analysis using Hadoop

Dhanya Nary Biju
Department of Computer Science & Engineering
Amity University, Haryana, India

Yojna Arora
Department of Computer Science & Engineering
Amity University, Haryana, India

Abstract

In today’s highly developed world, every minute, people around the globe express themselves via various platforms on the Web. And in each minute, a huge amount of unstructured data is generated. Such data is termed as big data. Twitter, one of the largest social media sites receives millions of tweets every day on variety of important issues. This huge amount of raw data can be used for industrial, social, economic, government policies or business purpose by organizing according to our requirement and processing. Hadoop is one of the best tool options for twitter data analysis as it works for distributed big data, streaming data, time stamped data, text data etc. Hence, Flume is used to extract real time twitter data into HDFS. Hive and Pig which is SQL like query language is used for some extraction and analysis. The goal of this project is to compare the results of Hive and Pig and analyse the retrieval time for a query. Finally we will conclude which framework will work fast and scalable for our dataset.

I. INTRODUCTION

Over past ten years, industries and organizations didn’t need to store and perform much operations and analytics on data of the customers. But around from 2005, the need to transform everything into data is much entertained to satisfy the requirements of the people. So, big data came into picture in the real time business analysis of processing data. From 20th century onwards this WWW has completely changed the way of expressing their views. Presently, people are expressing their thoughts through online blogs, discussion forms and also some online applications like Facebook, Twitter, etc. If we take Twitter as our example nearly 1TB of text data is generating within a week in the form of tweets. So, by this it is understand clearly how this Internet is changing the way of living and style of people. Among these, tweets can be categorized by the hash value tags for which they are commenting and posting their tweets. So, now many companies and also the survey companies are using this for doing some analytics such that they can predict the success rate of their product or also they can show the different view from the data that they have collected for analysis. But, to calculate their views is very difficult in a normal way by taking these heavy data that are going to generate day by day.

1.1. Objective

Twitter has over a billion users and everyday people generate billions of tweets over 100 hours per minute and this number is ever increasing. To analyse and understand the activity occurring on such a massive scale, a relational SQL database is not enough. Such kind of data is well suited to a massively parallel and distributed system like Hadoop.

The main objective of this project is to focus on how data generated from Twitter can be mined and utilized by different companies to make targeted, real time and informed decisions about their product that can increase their market share or to find out the views of people on a specific topic of interest. This can be done by using Hadoop concepts. The given project will focus on how data generated from Twitter can be mined and utilized. There are multiple applications of this project. Companies can use this project to understand how effective and penetrative their marketing programs are through sentiment analysis. In addition to it, companies can also evaluate the popular hash tags which are trending nowadays. Applications for Twitter data can be endless. This project can also help in analysing new emerging trends and knowing about people's changing behaviour with time. Also people in different countries have different preferences. By analysing the tweets/hash tags/sentiment etc., companies can understand what are the likes /dislikes of people around the world and work on their preferences accordingly.
1.2. Existing and Proposed System

The major issues involved in big data are the following:

- The first challenge faced is storing and accessing the information from the large huge amount of data sets from the clusters. We need a standard computing platform to manage large data since the data is growing, and data stores in different data storage locations in a centralized system, which will scale down the huge data into sizable data for computing.

- The second challenge is retrieving the data from the large social media data sets. In the scenarios where the data is growing daily, it’s somewhat difficult to accessing the data from the large networks if we want to do specific action to be performed.

- The third challenge concentrates on the algorithm design for handling the problems raised by the huge data volume and the dynamic data characteristics.

The main scope of the project is to fetching and analysing the tweets on demonetization and to perform sentiment analysis to find the most popular hash tags which is trending and finding the average rating of each tweet based on that topic.

Sentiment Analysis is the process of detecting the contextual polarity of text. A common use case for this technology is to discover how people feel about a particular topic. Here, our topic will be Demonetization. We can generate the popular hash tags used by the people on their tweets and sentiment analysis. After this, we will perform comparative analysis between Hive and Pig to find out which one is better for the analysis based on its execution time.

II. BACKGROUND STUDY

These days internet is being widely used than it was used a few years back. Billions of people are using social media and social networking every day all across the globe. Such a huge number of people generate a flood of data which have become quite complex to manage. Considering this enormous data, a term has been coined to represent it. This term is called Big Data. Big Data is the term coined to refer this huge amount of data. The concept of big data is fast spreading its arms all over the world.

2.1. Big Data

Data which is very large in size and yet growing exponentially with time is called as Big data. It refers to the large volume of data which may be structured or unstructured and which make use of certain new technologies and techniques to handle it.

![Fig.1: Big Data](image)

Hadoop is a programming framework used to support the processing of large data sets in a distributed computing environment. It provides storage for a large volume of data along with advanced processing power. It also gives the ability to handle multiple tasks and jobs. Hadoop was developed by Google’s MapReduce that is a software framework where an application break down into various parts. The Apache Hadoop ecosystem consists of the Hadoop Kernel, MapReduce, HDFS and numbers of various components like Apache Flume, Apache Hive and Apache Pig which are being used in this project.

2.1.1. Categories of Big Data
1) **Structured Data:** The data which can be stored and processed in table (rows and column) format is called as a structured data. Structured data is relatively simple to enter, store and analyze. Example - Relational database management system.

2) **Unstructured Data:** The data with unknown form or structure is called as unstructured data. They are difficult for nontechnical users and data analysts to understand and process. Example - Text files, images, videos, email, webpages, PDF files, PPT, social media data etc.

3) **Semi-structured Data:** Semi-structured data is data that is neither raw data nor organized in a rational model like a table. XML and JSON documents are semi structured documents.

### 2.1.2. Characteristics of Big Data

The characteristics of Big Data are defined by three V’s:

- **Volume** – It refers to the amount of data that is generated. The data can be low density, high volume, structured/unstructured or data with unknown value. The data can range from terabytes to petabytes.

- **Velocity** – It refers to the rate at which the data is generated. The data is received at an unprecedented speed and is acted upon in a timely manner.

- **Variety** – Variety refers to different formats of data. It may be structured, unstructured or semi-structured. The data can be audio, video, text or email.

### 2.2. Hadoop

As organizations are getting flooded with massive amount of raw data, the challenge here is that traditional tools are poorly equipped to deal with the scale and complexity of such kind of data. That's where Hadoop comes in. Hadoop is well suited to meet many Big Data challenges, especially with high volumes of data and data with a variety of structures.

Hadoop is a framework for storing data on large clusters of commodity hardware, everyday computer hardware that is affordable and easily available and running applications against that data. A cluster is a group of interconnected computers (known as nodes) that can work together on the same problem. The Current Apache Hadoop ecosystem consists of the Hadoop Kernel, Map-Reduce, HDFS and numbers of various components like Apache Hive, Pig, Flume etc.

Hadoop consists of two main components:

1) **HDFS (Data Storage)**

2) **Map-Reduce (Analysing and Processing)**

### 2.2.1. Architecture of Hadoop

HDFS is the main component of Hadoop architecture. It stands for Hadoop Distributed File Systems. It is used to store a large amount of data and multiple machines are used for this storage. MapReduce is another component of big data architecture. The data is processed here in a distributed manner across multiple machines. So, HDFS works as a storage part and MapReduce works as a processing part. Hive and Pig are the components of Hadoop ecosystem. These are high level data flow languages. MapReduce is the inner most layer of Hadoop ecosystem. [3]
2.2.1 Technologies Used

- **Apache Flume**: Apache Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of streaming data into the Hadoop Distributed File System (HDFS). It can be used for dumping twitter data in Hadoop HDFS. It has a simple and flexible architecture based on streaming data flows; and is robust and fault tolerant with tuneable reliability mechanisms for failover and recovery. Flume lets Hadoop users ingest high-volume streaming data into HDFS for storage. [5]

- **Apache Hive**: Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analysing easy. Hive provides the ability to store large amounts of data in HDFS. Hive was designed to appeal to a community comfortable with SQL. Hive uses an SQL like language known as HIVEQL. Its philosophy is that we don’t need yet another scripting language. Hive supports maps and reduced transform scripts in the language of the user’s choice which can be embedded with SQL. Supporting SQL syntax also means that it is possible to integrate with existing tools like. Hive has an ODBC (Open Database Connectivity) driver that allows and facilitates easy queries. It also adds support for indexes which allows support for queries common in such environment. Hive is a framework for performing analytical queries. Big Data enterprises require fast analysis of data collected over a period of time. Hive is an excellent tool for analytical querying of historical data. It is to be noted that the data needs to be well organized, which would allow Hive to fully unleash its processing and analytical powers. [5]

- **Apache Pig**: Pig comes from the language Pig Latin. Pig Latin is a procedural programming language and fits very naturally in the pipeline paradigm. When queries become complex with most of joins and filters then Pig is strongly recommended. Pig Latin allows users to store data at any point in the pipeline without disturbing the pipeline execution. Pig Latin allows developers to insert their own code almost anywhere in the data pipeline which is useful for pipeline development. This is accomplished through a user defined functions UDFS (User Defined Functions). UDFS allows user to specify how data is loaded, how data is stored and how data is processed. When you are looking to process clusters of unorganized, unstructured, decentralized data and don’t want to devote too much from your solid SQL foundation, Pig is the option to go with. You no longer need to get into writing core MapReduce jobs. If you already have SQL background, the learning curve will be smooth and development time will be faster. [5]

III. IMPLEMENTATION

This section will lead you to the steps which are involved throughout the development of the project.
3.1 Creating Twitter Application

First of all if we want to do sentiment analysis on Twitter data we want to get Twitter data first so to get it we want to create an account in Twitter developer and create an application.

1. Open the website dev.twitter.com/apps in the Mozilla Firefox Browser.

![Website to create Twitter API](image1)

**Fig 3:** Website to create Twitter API

2. We will now see the website suggesting us to sign in. So, we sign into our twitter account. Click on Create New App.

![Create New App](image2)

**Fig 4:** Create New App

3. Fill in all the required fields to make the application and use the website as google.com.
4. Now, scroll down and tick the option Yes, I agree and then click Create your Twitter application.

5. Click on manage keys and access tokens.
6. Now click on **Create my access token**.

7. Now, we open **flume.conf** file in the directory `/usr/lib/flume/conf` and then change the following keys in the file. These keys will be obtained from the page above.
8. These are the keys which we will change in the flume.conf file:

**Access Token, Access Token Secret, Consumer Key (API Key), Consumer Secret (API Secret)**

Also add the keywords that we want to extract from twitter. Here, we are extracting data on Demonetization.

![Flume Configuration File](image)

**Fig. 9: Flume Configuration File**

3.2 Getting Data using Flume

After creating an application in the Twitter developer site, we can now access the Twitter and we can get the information that we want. Here we will get everything in JSON format and this is stored in the HDFS that we have given the location where to save all the data that comes from the Twitter. After running the Flume, the Twitter data will automatically will save into HDFS.

Following are the steps followed to collect and store dataset from Twitter into HDFS:-

1. Open the terminal and start all the services using the `start-all.sh` command. Then check all the hadoop services which are running using `jps` command.
We will now start the flume agent using the following command:

```
```

3. This is the list of twitter data extracted which contains the keyword as specified in the conf file.
4. The dataset will look like this:

Fig 14: Twitter Dataset

4.3. Sentiment Analysis

We have collected and stored the tweets in HDFS using Flume in the previous section. The tweets are located in the following location of the HDFS: /flumedir/data/tweets_raw/

Now, we will be performing sentiment analysis on twitter data using both Hive and Pig.

4.3.1. Determining Popular Hash Tags

Using Hive:

- As the tweets coming in from Twitter are in Json format, we need to load the tweets into the Hive using json input format. Let’s use Cloudera Hive json serde for this purpose. We need to ADD the jar file into Hive as shown below:

  ADD jar /usr/local/hive/lib/hive-serdes-1.0-SNAPSHOT.jar

- After successfully adding the Jar file, let’s create a Hive table to store the Twitter data.

For calculating the hashtags, we need the tweet_id and hashtag_text, so we will create a Hive table that will extract the id and hashtag_text from the tweets using the Cloudera Json serde. So, let’s create an external table in Hive in the same directory where our tweets are present i.e., '/flumedir/data/tweets_raw/', so that the tweets present in this location will be automatically stored in the Hive table. The command for creating a Hive table to store id and hashtag text of the tweets is as follows:

CREATE EXTERNAL TABLE tweets (id BIGINT, entities STRUCT<hashtags:ARRAY<STRUCT<text:STRING>>>) ROW FORMAT SERDE 'com.cloudera.hive.serde.JSONSerDe' LOCATION '/flumedir/data/tweets_raw';

Table 1: Structure of tweets Table

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>bigint</td>
</tr>
<tr>
<td>entities</td>
<td>struct&lt;hashtags:array&lt;struct<a href="">text:string</a>&gt;&gt;</td>
</tr>
</tbody>
</table>

Here, entities is a structures in which hashtags is an array consisting of another structures in it where the text is inside the structure.

- Now, from this structure, we need to extract only the hashtags array which consists of the text. We can achieve this by using the following command:
create table hashtags as select id as id, entities. hashtags.text as words from tweets;

Here, we are creating a table with the name ‘hashtags’ and in that table we are storing the tweet_id and the hashtags text array.

Table 2: Structure of hashtags Table

<table>
<thead>
<tr>
<th>Id</th>
<th>Bigint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Words</td>
<td>array&lt;string&gt;</td>
</tr>
</tbody>
</table>

- Here, we can see that there are two or more hashtags in each tweet, so we need to extract each hashtag in a new row. So, let’s split each word inside the array as a new row. In order to do that, we need to use a UDTF/User Defined Table Generating Function, which generates each new row for each value inside an array. We have a built-in UDTF called explode which will extract each element from an array and create a new row for each element.

Now, let’s create another table which can store id and the hashtag text using the below command:

create table hashtag_word as select id as id, hashtag from hashtags LATERAL VIEW explode(words) w as hashtag;

In general, explode UDTF has some limitations; explode cannot be used with other columns in the same select statement. So we will add LATERAL VIEW in conjunction with explode so that the explode function can be used in other columns as well.

Table 3: Structure of hashtag_word Table

<table>
<thead>
<tr>
<th>id</th>
<th>bigint</th>
</tr>
</thead>
<tbody>
<tr>
<td>hashtag</td>
<td>string</td>
</tr>
</tbody>
</table>

- Now, let’s use the query to calculate the number of times each hashtag has been repeated.

select hashtag, count(hashtag) as total_count from hashtag_word group by hashtag order by total_count desc;

Fig. 15: Hive Script for Hashtag Count

- Now, we will run the hive script using the following command:
Fig. 16: Execution of hashtag.sql Script

Output:

In the below screenshot, we can see that the hashtag and the number of times it is repeated in the Twitter data we have. Here, we have counted the number of popular hashtags in Twitter using Hive.

Fig. 17: Popular Hashtags using Hive

Using Pig:

- The data from Twitter is in 'Json' format, so a Pig JsonLoader is required to load the data into Pig. So, we have to register the downloaded jars in Pig by using the following commands:

  REGISTER '/home/dhanya/Desktop/elephant-bird-hadoop-compat-4.1.jar';
  REGISTER '/home/dhanya/Desktop/elephant-bird-pig-4.1.jar';
  REGISTER '/home/dhanya/Desktop/json-simple-1.1.1.jar';

- The tweets are in nested Json format and consist of map data types. We need to load the tweets using JsonLoader which supports maps, so we are using elephant bird JsonLoader to load the tweets. Below is the first Pig statement that is required to load the tweets into Pig:

  load_tweets = LOAD '/flumedir/data/tweets_raw/' USING com.twitter.elephantbird.pig.load.JsonLoader('nestedLoad') AS myMap;

- Now, let's extract the id and the hashtag from the above tweets and the Pig statement for doing this is as shown below:

  extract_details = FOREACH load_tweets GENERATE FLATTEN(myMap#entities') as (m:map[]),FLATTEN(myMap#id') as id;

In the tweet, the hashtag is present in the map object entities. Since the hashtags are inside the map entities, we have extracted the entities as map[ ] data type.

- Now, from the entities, we have to extract the hashtags which is again a map. So we will extract the hashtags as map[ ] data type as well.
hash = foreach extract_details generate FLATTEN(m#hashtags') as(tags:map[]), id as id;

- Now, from the extracted hashtags, we need to extract text which contains the actual hashtag. This can be done using the following command:

  txt = foreach hash generate FLATTEN(tags#text') as text, id;

  Here, we have extracted the text which starts with # and named it with an alias name text.

- Now, we will group the relation by hashtag’s text by using the below relation:

  grp = group txt by text;

- The next thing to do is, count the number of times the hashtag is repeated by the user. This can be achieved using the below relation:

  cnt = foreach grp generate group as hashtag_text, COUNT(txt.text) as hashtag_cnt;

Fig 18: Pig Script for Hashtag Count

- Now, we will run the pig script using the following command:

  pig Desktop/hashtag.pig

Fig 19: Execution of hashtag.pig Script

OUTPUT:-

Now we have the hashtags and its count in a relation as shown in the below screen shot.
4.3.2. Determining Average Rating of Tweets

Using Hive:

- Here also, we need to add the Cloudera Hive json serde jar file into Hive.
- For performing Sentiment Analysis, we need the tweet_id and tweet_text, so we will create an external Hive table in the same directory where our tweets are present so that tweets which are present in this location will be automatically stored in the Hive table. We will extract the id and tweet_text from the tweets using the Cloudera Json serde.

The command for creating a Hive table to store id and text of the tweets is as follows:

```sql
CREATE EXTERNAL TABLE load_tweets(id BIGINT, text STRING) ROW FORMAT SERDE 'com.cloudera.hive.serde.JSONSerDe' LOCATION '/flumedir/data/tweets_raw/';
```

<table>
<thead>
<tr>
<th>Table 4: Structure of load_tweets Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
<tr>
<td>text</td>
</tr>
</tbody>
</table>

- Next, we will split the text into words using the split() UDF available in Hive. If we use the split() function to split the text as words, it will return an array of values. So, we will create another Hive table and store the tweet_id and the array of words.

```sql
create table split_words as select id as id, split(text,' ') as words from load_tweets;
```

<table>
<thead>
<tr>
<th>Table 5: Structure of split_words Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
<tr>
<td>words</td>
</tr>
</tbody>
</table>

- Next, let’s split each word inside the array as a new row. So, let’s create another table which can store id and word.

```sql
create table tweet_word as select id as id, word from split_words LATERAL VIEW explode(words) w as word;
```

<table>
<thead>
<tr>
<th>Table 6: Structure of tweet_word Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
<tr>
<td>word</td>
</tr>
</tbody>
</table>
Let’s use a dictionary called AFINN to calculate the sentiments. AFINN is a dictionary which consists of 2500 words rated from +5 to -5 depending on their meaning.

We will create an external table to store the contents of AFINN dictionary which is residing in the HDFS directory: /flumedir/data/dictionary/

```
CREATE EXTERNAL TABLE dictionary(word string, rating int) ROW FORMAT DELIMITED FIELDS TERMINATED BY 't' STORED AS TEXTFILE LOCATION '/flumedir/data/dictionary/';
```

<table>
<thead>
<tr>
<th>Table 7: Structure of dictionary Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>word</td>
</tr>
<tr>
<td>rating</td>
</tr>
</tbody>
</table>

Now, let’s load the AFINN dictionary into the table by using the following command:

```
load data inpath 'flumedir/data/dictionary/AFINN.txt' into TABLE dictionary;
```

Now, we will join the tweet_word table and dictionary table so that the rating of the word will be joined with the word.

```
create table word_join as select tweet_word.id, tweet_word.word, dictionary.rating from tweet_word LEFT OUTER JOIN dictionary ON(tweet_word.word = dictionary.word);
```

<table>
<thead>
<tr>
<th>Table 8: Structure of word_join Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
<tr>
<td>word</td>
</tr>
<tr>
<td>rating</td>
</tr>
</tbody>
</table>

Here, the rating column has been added along with the id and the word. Whenever there is a match with the word of the tweet in the dictionary, the rating will be given to that word else NULL will be present.

Now we will perform the ‘groupby’ operation on the tweet_id so that all the words of one tweet will come to a single place. And then, we will be performing an Average operation on the rating of the words of each tweet so that the average rating of each tweet can be found.

```
select id, AVG(rating) as rating from word_join GROUP BY word_join.id order by rating DESC;
```

In the above command, we have calculated the average rating of each tweet by using each word of the tweet and arranging the tweets in the descending order as per their rating.
Now, we will run the hive script to perform analysis.

OUTPUT:-

In the below screen shot, we can see the tweet_id and its rating.
Using Pig:

- Here also, we have to register the Pig JsonLoader jar files which are required to load the data into Pig.

- The tweets are in nested JSON format and consist of map data types. We need to load the tweets using JsonLoader which supports maps, so we are using elephant bird JsonLoader to load the tweets. Below is the first Pig statement required to load the tweets into Pig:

  ```pig
  load_tweets = LOAD '/flumedir/data/tweets_raw/' USING com.twitter.elephantbird.pig.load.JsonLoader('nestedLoad') AS myMap;
  ```

- Now, we shall extract the id and the tweet text from the above tweets. The Pig statement necessary to perform this is as shown below:

  ```pig
  extract_details = FOREACH load_tweets GENERATE myMap#'id' as id, myMap#'text' as text;
  ```

  We have the tweet id and the tweet text in the relation named as extract_details.

- Now, we shall extract the words from the text using the TOKENIZE keyword in Pig.

  ```pig
tokens = foreach extract_details generate id, text, FLATTEN(TOKENIZE(text)) As word;
  ```

  We will load the dictionary into Pig by using the below statement:

  ```pig
dictionary = load '/flumedir/data/dictionary/AFINN.txt' using PigStorage('\t') AS(word:chararray, rating:int);
  ```

- Now, let’s perform a map-side join by joining the tokens statement and the dictionary contents using this command:

  ```pig
  word_rating = join tokens by word left outer, dictionary by word using 'replicated';
  ```

  Here, the word_rating has joined the tokens(consists of id, tweet text, word) statement and the dictionary(consists of word, rating).
• Now we will extract the id, tweet text and word rating(from the dictionary) by using the below relation:

\[
\text{rating} = \text{foreach word_rating generate tokens::id as id, tokens::text as text, dictionary::rating as rate;}
\]

Here, our relation now consists of id, tweet text, and rate(for each word).

• Now, we will group the rating of all the words in a tweet by using the below relation:

\[
\text{word_group} = \text{group rating by (id, text)};
\]

Here we have grouped by two constraints, id and tweet text.

• Now, let’s perform the Average operation on the rating of the words per each tweet.

\[
\text{avg_rate} = \text{foreach word_group generate group, AVG(rating.rate) as tweet_rating};
\]

• At last, we will order the tweets in descending order to see the tweets ranging from positive to negative manner.

\[
\text{ordr} = \text{order avg_rate by $1 desc;}\
\]

Fig 24: Pig Script for Average Rating

• Now we will run the pig script to perform analysis.
OUTPUT:
From the above relation, we will get all the tweets i.e., both positive and negative.

Here, we can classify the positive tweets by taking the rating of the tweet which can be from 0-5. We can classify the negative tweets by taking the rating of the tweet from -5 to -1.

IV. RESULTS

Here, at last, we can compare between Hive and Pig based on the execution time of the queries in this section.

5.1. Comparative Analysis

After performing operations on the dataset using pig and hive, we can now perform the comparative analysis between them by considering the total execution time of both the scripts performing hash tag count and average rating on the tweets. So, this analysis result can help industries, corporation and individual for taking any decision regarding company, issues and many things.

In our experiment we also introduced hive which is more useful as compared to pig on analysis of datasets. We can say that hive perform faster as compared to pig on the basis of various parameters, also the above previous queries which were performed demonstrates that the execution time taken by hive is very less as compared to pig. And the Map-Reduce jobs generated by hive are less as compared to pig whereby the execution time is less
in hive. Another benefit of using hive is number of lines of code, which are more in pig but in hive only one line of query is sufficient. The experimental results are shown below-

![Fig 27: Execution Time of Queries](image)

V. CONCLUSION

The task of big data analysis is not only important but also a necessity. In fact many organizations that have implemented Big Data are realizing significant competitive advantage compared to other organizations with no Big Data efforts. The project is intended to analyse the Twitter Big Data and come up with significant insights which cannot be determined otherwise.

As twitter post are very important source of opinion on different issues and topics. It can give a keen insight about a topic and can be a good source of analysis. Analysis can help in decision making in various areas. Apache Hadoop is one of the best options for twitter analysis. Once the system is set up using FLUME and HIVE, it helps in analysis of diversity of topics by just changing the keywords in query. Also it does the analysis on real time data, so is more useful. The analysis what I did could be helpful in finding out the moods of people on a particular topic like Demonetization, also effectively used to compute such results in order to determine the current trends with respect to particular topics. This can be very useful in the marketing sector. On the other hand, it provides consumers to distinguish properly among the institutions and make the service provider selection vigorously, based on the parameters like execution time, number of map-reduce jobs, lines of code it has been examined that hive holds better and efficient than pig.

VI. REFERENCES

[5] Big Data Hadoop by V. K. Jain