

# Computer Sci Electrical Engr 5590 0004

## Big Data Programming

### Project Title

Stock Market Analysis using Hadoop Ecosystem

### Team Members (Team-1)

- Jaisekhar Koya
- Sri Sai Nikhil Kantipudi
- Sai Rohith Guntupally
- Aarthi Nagireddy

### Introduction

Study the stock market with graphs, news, important dates, and many more is always hard which leads to inaccuracy in analysis and investment in particular stock. This project helps to analyze the stock market using stock market data and twitter data using Hadoop ecosystem. It collects the twitter data associated with the stock symbols and analyze/predict the stock movement.

### Background

There are many articles which tells about the handling of stock market using Hadoop Technologies. But there is no depth analysis of data. Only few articles talked about twitter data and did analysis only for couple of days and not used NLP Techniques. For example: <https://www.3pillarglobal.com/insights/analyze-big-data-hadoop-technologies>

### Goals & Objectives

The main goal of this project is to analyze the stock market with the help of stock market data and also the social data(twitter data in this case) using Hadoop ecosystem. This system collects the twitter data associated with the stock symbols and calculate the sentiment of the stock based on the user tweets and extract the trend from the analysis.

- Motivation

We have tried hard to study the stock market with graphs, news, important dates, and many more, but we failed many times which leads to loss of profits and sometimes even the

investments. We always have an extremely hard time analyzing that huge amount of data and coming to a decision based on the analysis. And analyzing that huge data takes a lot of time and got to concentrate on various works too. So, to do this task for us we need some sort of Big data tools to analyze and give us filtered results which we can cross-check and invest in that stock confidently. Not only that as big data tools analyze data extremely fast, we can make decisions quicker than we do. We can also avoid human error using these tools.

- **Significance**

- As more than half of the population are investing in stocks it is one of the most significant topics to be considered.
- 30% of investors failing to analyze the stock and predicting the uptrend or downtrend of stock is one of the major reasons.
- Big data tools help to analyze the huge data which helps to provide the efficient results.
- It reduces the analysis time of investors and helps in making decisions faster and error free.

- **Objectives**

To create a stock market analysis system, we will be collecting the twitter data using NLP techniques and processing the data. Different tools in Hadoop ecosystem will be used to analyze the data and help in predicting the movement of the stock.

- **Features**

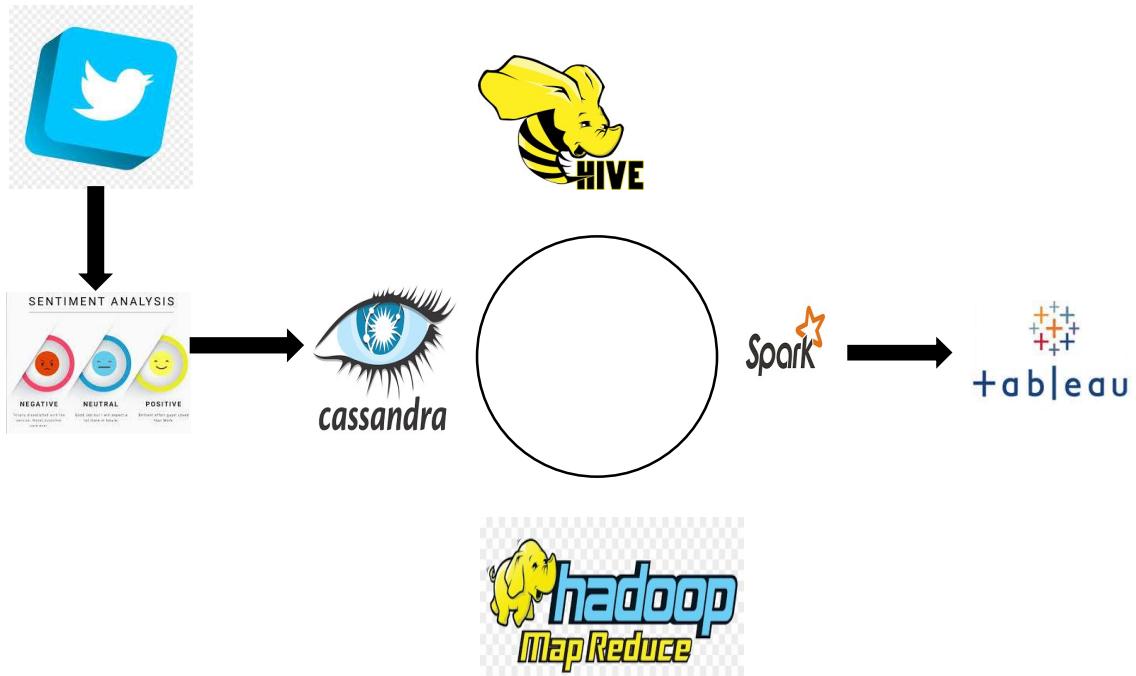
The main features of this project is twitter data extraction, sentimental analysis on the extracted data. Using HDFS to store the data. Apache spark to process the data and Hive for querying the data. A visualization tool for showing out the results.

## Dataset

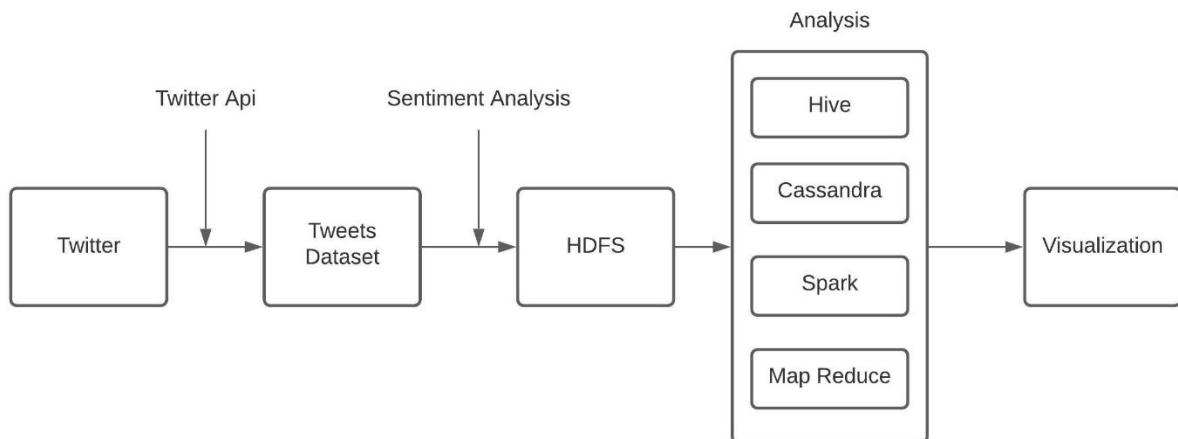
**Twitter Dataset:** (Only for project purpose. Do not share.)

[https://mailmissouri-my.sharepoint.com/:u/g/personal/skgyc\\_umsystem\\_edu/EXI5i-g3529CmwhdsDDSHoABtW7cOW4U\\_UtctybYaAsnyQ?e=WGriiy](https://mailmissouri-my.sharepoint.com/:u/g/personal/skgyc_umsystem_edu/EXI5i-g3529CmwhdsDDSHoABtW7cOW4U_UtctybYaAsnyQ?e=WGriiy)

## Architecture



## Design



## Analysis: (Data)

Twitter is one of the best data sources as it contains huge public opinions. Various number of users/companies tweets the information regarding many things. This dataset is about AAPL stock information. Hence, we extracted all the tweets related to the AAPL stock. Since, our project is to analyze the previous data and predict/analyze the stock movement we are considering the tweets from October 1<sup>st</sup> to October 28<sup>th</sup>. We extracted more than 100K tweets and every tweet object contains complete information about the specific tweet like created\_at, retweet\_count, user details, entity details etc.

Keyword: AAPL

Date Range: 10/01/20 – 10/28/20

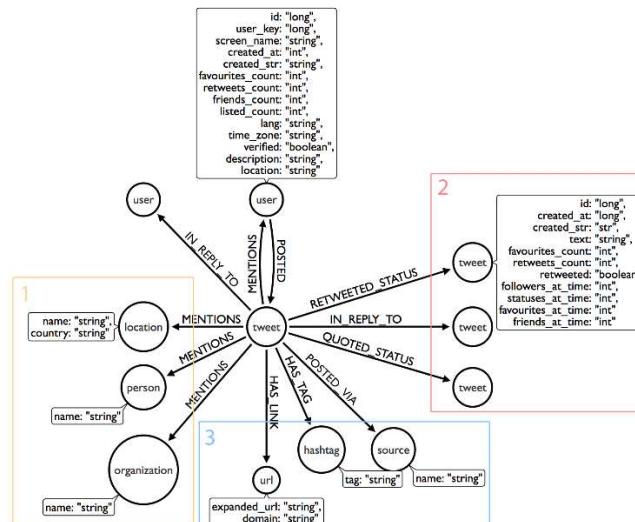
No. of tweets: 100k

The dataset will look like below

```
jaisechar@Jaidev:~/Desktop$ curl -X GET https://jaisechar.com/api/tweets11.json
/home/jaisechar/.jaisechar/tweets11.json
dou nano 2.9.3
```

The terminal output shows the command used to fetch the tweets and the path where the JSON file was saved. The file contains 83122 lines of tweet data.

## Features

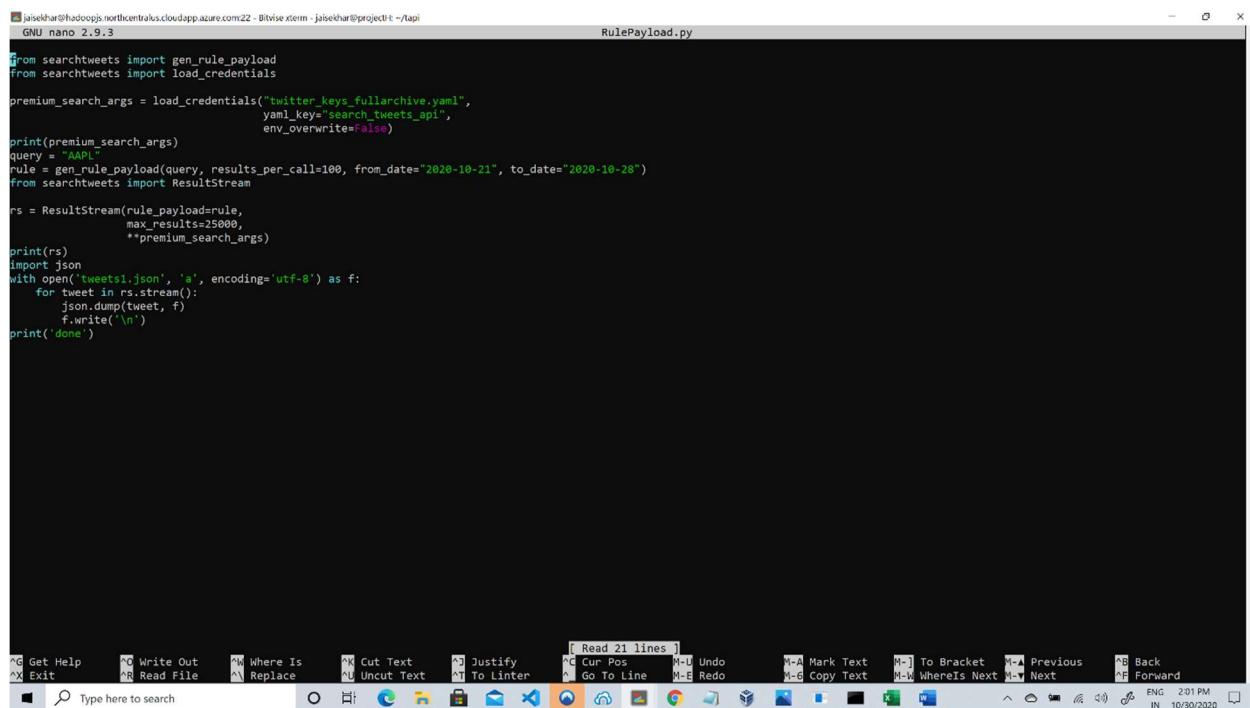


## Implementation

- Tweets Extraction:

To extract the tweets, we used different API's or library's in the python. Tweepy in the python is the widely used library to extract the tweets but it comes with limitations where we cannot extract tweets based on the data range. Even there were other libraries like twint or GetOldTweets3 but those were not working with the recent endpoint update at twitter api. Hence, we used searchtweets library which is nothing but a static library which makes direct API call to the twitter.

The below code is used to retrieve the tweets using searchtweets library. This code is importing the credentials from the other file and sending query(here we are querying AAPL), results\_per\_call, from\_date, to\_date (October month's) parameters to the api with max\_results constraint which is set to 25000 here. As we used four keys we are able to retrieve approx 100k tweets by running four times of below code using four different twitter keys of ours.



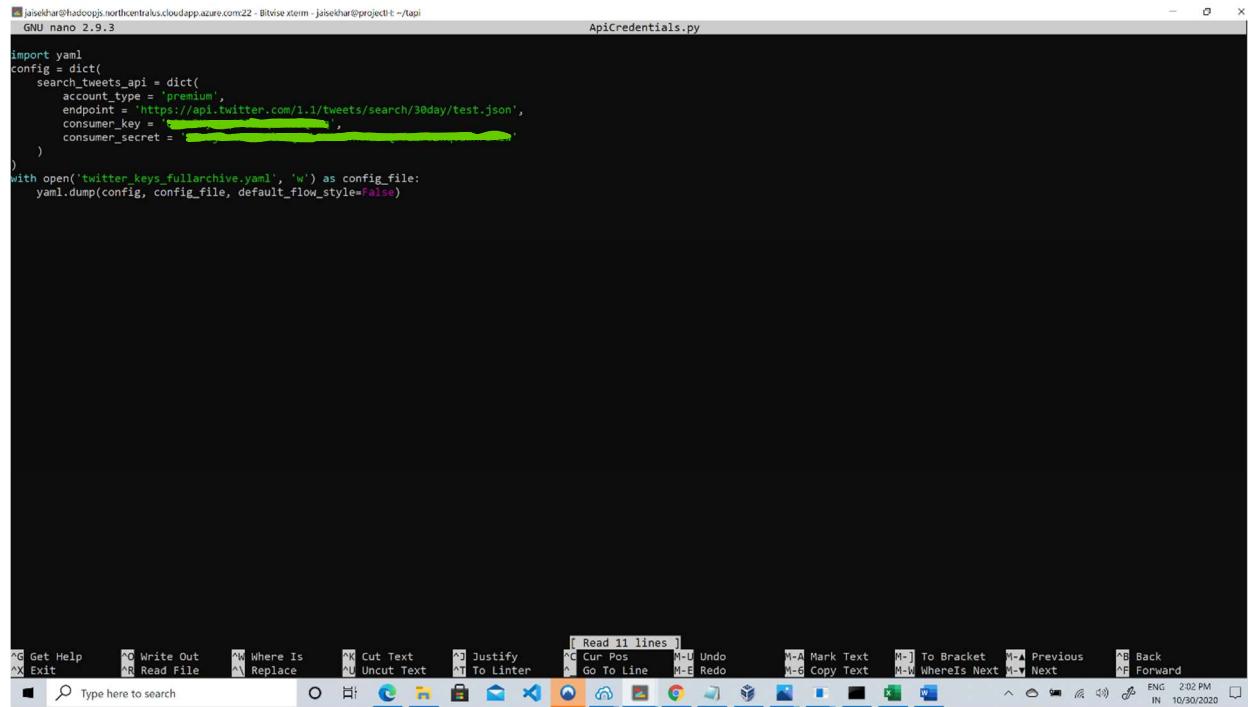
A screenshot of a Windows terminal window titled "RulePayload.py". The window shows a block of Python code for extracting tweets. The code imports searchtweets, loads credentials from a YAML file, defines a query for "AAPL", and sets up a ResultStream with a maximum of 25000 results per call. It then iterates through the stream, dumping each tweet to a JSON file named "tweets1.json". The terminal window has a dark background and includes standard Windows-style icons in the taskbar at the bottom.

```
jaiseelhar@hadoopjs:~/Desktop$ nano RulePayload.py
GNU nano 2.9.3
from searchtweets import gen_rule_payload
from searchtweets import load_credentials

premium_search_args = load_credentials("twitter_keys_fullarchive.yaml",
                                       yaml_key="search_tweets_api",
                                       env_overwrite=False)
print(premium_search_args)
query = "AAPL"
rule = gen_rule_payload(query, results_per_call=100, from_date="2020-10-21", to_date="2020-10-28")
from searchtweets import ResultStream

rs = ResultStream(rule_payload=rule,
                  max_results=25000,
                  **premium_search_args)
print(rs)
import json
with open('tweets1.json', 'a', encoding='utf-8') as f:
    for tweet in rs.stream():
        json.dump(tweet, f)
        f.write("\n")
print('done')
```

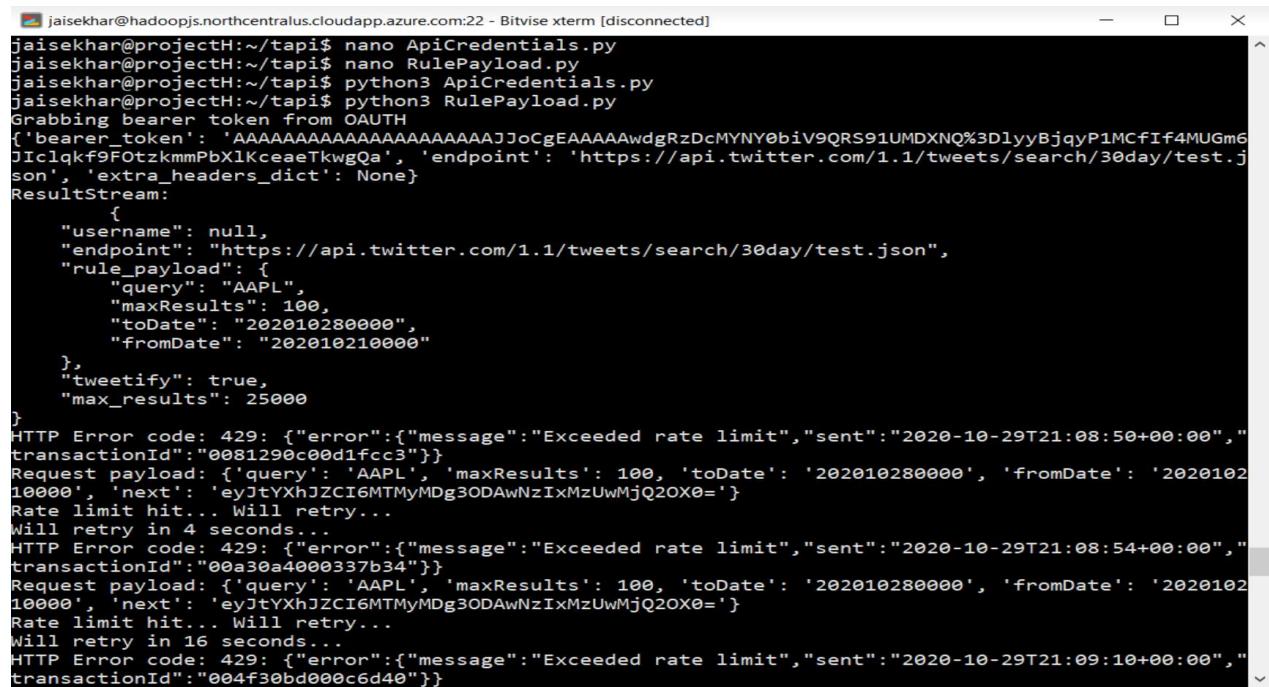
These are the credentials which are used to as the args to the searchtweets api .



```
GNU nano 2.9.3          ApiCredentials.py

import yaml
config = dict(
    search_tweets_api = dict(
        account_type = 'premium',
        endpoint = 'https://api.twitter.com/1.1/tweets/search/30day/test.json',
        consumer_key = 'XXXXXXXXXXXXXX',
        consumer_secret = 'XXXXXXXXXXXXXX'
    )
)
with open('twitter_keys_fullarchive.yaml', 'w') as config_file:
    yaml.dump(config, config_file, default_flow_style=False)
```

Below you can see the execution of all the python files to extract the tweets.



```
jaisekhar@projectH:~/tapi$ nano ApiCredentials.py
jaisekhar@projectH:~/tapi$ nano RulePayload.py
jaisekhar@projectH:~/tapi$ python3 ApiCredentials.py
jaisekhar@projectH:~/tapi$ python3 RulePayload.py
Grabbing bearer token from OAUTH
{'bearer_token': 'AAAAAAAAAAAAAAAAAAAAAAJJoCgEAAAAdwgRzDcMYNY0biV9QRs91UMDXNQ%3DlyyBjqyP1MCfif4MUGm6
Jic1qkf9FOtzkmmPbX1KceaeTkwgQa', 'endpoint': 'https://api.twitter.com/1.1/tweets/search/30day/test.json', 'extra_headers_dict': None}
ResultStream:
{
    "username": null,
    "endpoint": "https://api.twitter.com/1.1/tweets/search/30day/test.json",
    "rule_payload": {
        "query": "AAPL",
        "maxResults": 100,
        "toDate": "202010280000",
        "fromDate": "202010210000"
    },
    "tweetify": true,
    "max_results": 25000
}
HTTP Error code: 429: {"error":{"message":"Exceeded rate limit","sent":"2020-10-29T21:08:50+00:00","transactionId":"0081290c00d1fcc3"}}
Request payload: {'query': 'AAPL', 'maxResults': 100, 'toDate': '202010280000', ' fromDate': '202010210000', 'next': 'eyJtYXhJZCI6MTMyMDg3ODAwNzIxMzUwMjQ2OX0='}
Rate limit hit... Will retry...
Will retry in 4 seconds...
HTTP Error code: 429: {"error":{"message":"Exceeded rate limit","sent":"2020-10-29T21:08:54+00:00","transactionId":"00a30a4000337b34"}}
Request payload: {'query': 'AAPL', 'maxResults': 100, 'toDate': '202010280000', 'fromDate': '202010210000', 'next': 'eyJtYXhJZCI6MTMyMDg3ODAwNzIxMzUwMjQ2OX0='}
Rate limit hit... Will retry...
Will retry in 16 seconds...
HTTP Error code: 429: {"error":{"message":"Exceeded rate limit","sent":"2020-10-29T21:09:10+00:00","transactionId":"004f30bd000c6d40"}}
```

The results are saved in tweets1.json and number lines in the below snippet is nothing but the number of tweets in the json file as the every line indicates one tweet object.

```
jaisekhar@hadoopjs.northcentralus.cloudapp.azure.com:22 - Bitvise xterm [disconnected]
Will retry in 4 seconds...
HTTP Error code: 429: {"error":{"message":"Exceeded rate limit","sent":"2020-10-29T21:14:40+00:00","transactionId":"0034a4410023995c"}}
Request payload: {'query': 'AAPL', 'maxResults': 100, 'toDate': '202010280000', 'fromDate': '202010210000', 'next': 'eyJtYXhJZCI6MTMxOTA3MDI1MjA3Mjg4MjE3N30='}
Rate limit hit... Will retry...
Will retry in 16 seconds...
HTTP Error code: 429: {"error":{"message":"Exceeded rate limit","sent":"2020-10-29T21:14:56+00:00","transactionId":"00d68a9200e28cccd"}}
Request payload: {'query': 'AAPL', 'maxResults': 100, 'toDate': '202010280000', 'fromDate': '202010210000', 'next': 'eyJtYXhJZCI6MTMxOTA3MDI1MjA3Mjg4MjE3N30='}
Rate limit hit... Will retry...
Will retry in 36 seconds...
HTTP Error code: 429: {"error":{"message":"Exceeded rate limit","sent":"2020-10-29T21:15:47+00:00","transactionId":"004e0ca10031ca4b"}}
Request payload: {'query': 'AAPL', 'maxResults': 100, 'toDate': '202010280000', 'fromDate': '202010210000', 'next': 'eyJtYXhJZCI6MTMxODczMDIyMTI0MDAzMzI4MX0='}
Rate limit hit... Will retry...
Will retry in 4 seconds...
HTTP Error code: 429: {"error":{"message":"Exceeded rate limit","sent":"2020-10-29T21:15:51+00:00","transactionId":"0035621e003bc5d7"}}
Request payload: {'query': 'AAPL', 'maxResults': 100, 'toDate': '202010280000', 'fromDate': '202010210000', 'next': 'eyJtYXhJZCI6MTMxODczMDIyMTI0MDAzMzI4MX0='}
Rate limit hit... Will retry...
Will retry in 16 seconds...
HTTP Error code: 429: {"error":{"message":"Exceeded rate limit","sent":"2020-10-29T21:16:07+00:00","transactionId":"00b4d98600dcfd332"}}
Request payload: {'query': 'AAPL', 'maxResults': 100, 'toDate': '202010280000', 'fromDate': '202010210000', 'next': 'eyJtYXhJZCI6MTMxODczMDIyMTI0MDAzMzI4MX0='}
Rate limit hit... Will retry...
Will retry in 36 seconds...
done
jaisekhar@projectH:~/tapi$ wc -l tweets1.json
83122 tweets1.json
jaisekhar@projectH:~/tapi$
```

- **Storing in HDFS:**

The tweets file (tweets1.json) is copied to HDFS for further analysis using the the below commad

```
hdfs dfs -copyFromLocal ~/tapi/tweets1.json /bdp
```

Later, the file is moved to other directory which only contains that particular json file that helps while creating table in hive.

```
jaisekhar@hadoopjs.northcentralus.cloudapp.azure.com:22 - Bitvise xterm [disconnected]
Last login: Thu Oct 29 23:35:40 2020 from 136.37.18.201
jaisekhar@projectH:~$ hdfs dfs -ls /bdp
Found 2 items
-rw-r--r-- 1 jaisekhar supergroup 85492 2020-10-29 23:58 /bdp/json-serde-1.3.8-jar-with-dependencies.jar
-rw-r--r-- 1 jaisekhar supergroup 457332044 2020-10-30 00:13 /bdp/tweets1.json
jaisekhar@projectH:~$ hdfs dfs -mkdir /bdp/twitter
jaisekhar@projectH:~$ hdfs dfs -cp /bdp/tweets1.json /bdp/twitter
jaisekhar@projectH:~$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/jaisekhar/hive/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/jaisekhar/hadoop-2.7.3/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
```

- Hive Operations: (initialization and Loading Data)

To start hive, we ran schematool to initialize the metastore\_db and then we started the hive.

```
jaisekhar@hadoopjs.northcentralus.cloudapp.azure.com:22 - Bitvise xterm [disconnected]
Initialization script completed
schemaTool completed
jaisekhar@projectH:~/hive$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/jaisekhar/hive/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/jaisekhar/hadoop-2.7.3/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/home/jaisekhar/hive/lib/hive-common-2.1.0.jar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
hive> set hive.support.sql11.reserved.keywords=false;
```

Later we added json serde dependency jar to the hive using following command. JSON SerDe is used to map the json object to the table schema. Thus, it helps in mapping the json data to the table.

```
jaisekhar@projectH:~/hive$ cd ..
jaisekhar@projectH:~$ hdfs dfs -ls /
Found 4 items
drwxr-xr-x  - jaisekhar supergroup 0 2019-09-17 17:25 /hbase
drwxr-xr-x  - jaisekhar supergroup 0 2019-09-19 20:08 /home
drwxr-xr-x  - jaisekhar supergroup 0 2019-09-16 17:35 /jstest
drwx-----  - jaisekhar supergroup 0 2019-09-10 20:20 /tmp
jaisekhar@projectH:~$ hdfs dfs -ls -mkdir /bdp
-ls: Illegal option -mkdir
Usage: hadoop fs [generic options] -ls [-d] [-h] [-R] [<path> ...]
jaisekhar@projectH:~$ hdfs dfs -mkdir /bdp
jaisekhar@projectH:~$ hdfs dfs -copyFromLocal json-serde-1.3.8-jar-with-dependencies.jar /bdp
jaisekhar@projectH:~$ hive
```

Next, we created the table in hive tweets\_raw with the few required attributes from the tweet object which we extracted from twitter api. If you see here, the rows were delimited by serde which points to the added jar of json serde and location is pointing to the directory in hdfs which contains the tweets json file.

```
jaisekhar@hadoopjs.northcentralus.cloudapp.azure.com:22 - Bitvise xterm [disconnected]
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/jaisekhar/hive/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/jaisekhar/hadoop-2.7.3/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/home/jaisekhar/hive/lib/hive-common-2.1.0.jar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
hive> set hive.support.sql11.reserved.keywords=false;
hive> CREATE EXTERNAL TABLE tweets_raw (
    >     id BIGINT,
    >     created_at STRING,
    >     source STRING,
    >     favorited BOOLEAN,
    >     retweet_count INT,
    >    retweeted_status STRUCT<
    >         text:STRING,
    >         user:STRUCT<screen_name:STRING,name:STRING>,
    >     entities STRUCT<
    >         urls:ARRAY<STRUCT<expanded_url:STRING>>,
    >         user_mentions:ARRAY<STRUCT<screen_name:STRING,name:STRING>>,
    >         hashtags:ARRAY<STRUCT<text:STRING>>>,
    >     text STRING,
    >     user STRUCT<
    >         screen_name:STRING,
    >         name:STRING,
    >         friends_count:INT,
    >         followers_count:INT,
    >         statuses_count:INT,
    >         verified:BOOLEAN,
    >         utc_offset:STRING, -- was INT but nulls are strings
    >         time_zone:STRING>,
    >     in_reply_to_screen_name STRING,
    >     year int,
    >     month int,
    >     day int,
    >     hour int
    > )
    > ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
    > LOCATION '/bdp/twitter'
    > ;
```

Now, the extracted tweets data is successfully imported to table in hive. Next, we will perform analysis on the data which we extracted into hive and hdfs.

- MapReduce – wordcount:

The below python code is used to parse the hashtags from the entities in tweets1.json

```
jaisekhar@hadoop-northcentralus.cloudapp.azure.com:22 - Bitvise Xterm - jaisekhar@project0 ~ /twitter
GNU nano 2.9.3                                     parse.py

# -*- coding: utf-8 -*-
import jsonpickle
import json

tweets = list(open('/home/jaisekhar/tapi/tweets1.json','rt'))

for t in tweets:
    t = json.loads(t)
    try:
        if len(t['entities']['hashtags']) != 0:
            for h in t['entities']['hashtags']:
                print h['text'].encode('utf8')
    except KeyError:
        pass

^G Get Help      ^Q Write Out     ^W Where Is      ^K Cut Text      ^J Justify      [ Read 14 lines ]      ^C Cur Pos      ^U Undo      ^A Mark Text      ^T To Bracket      ^P Previous      ^B Back
^X Exit         ^R Read File     ^L Replace      ^U Uncut Text    ^N To Linter     ^M Go To Line    ^E Redo       ^W WhereIs Next  ^Y Next        ^F Forward
^S Type here to search
```

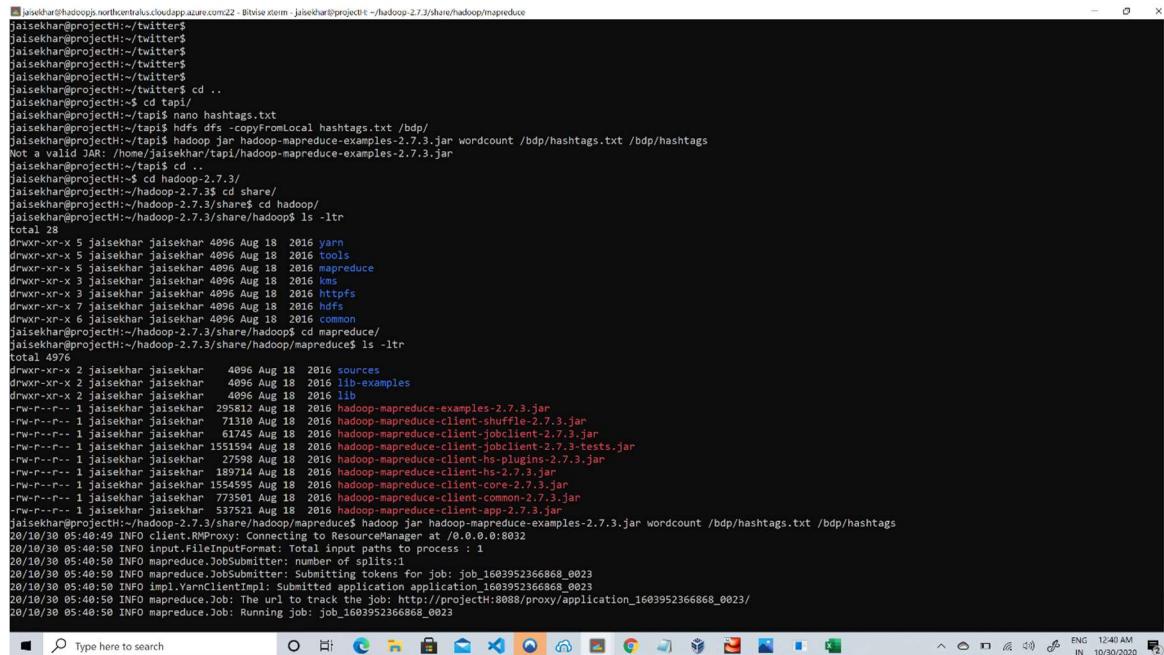
The python code is executed and saved the output in hashtags.txt using out operator.

We successfully got the hashtags into the hashtags.txt. And as shown below the hashtags.txt is copied to hdfs using the below command

```
hdfs dfs -copyFromLocal hashtags.txt /bdp
```

Now wordcount operation is performed using hadoop inbuilt example jar with hashtags.txt as input and saved the output in /bdp/hashtags directory in hdfs.

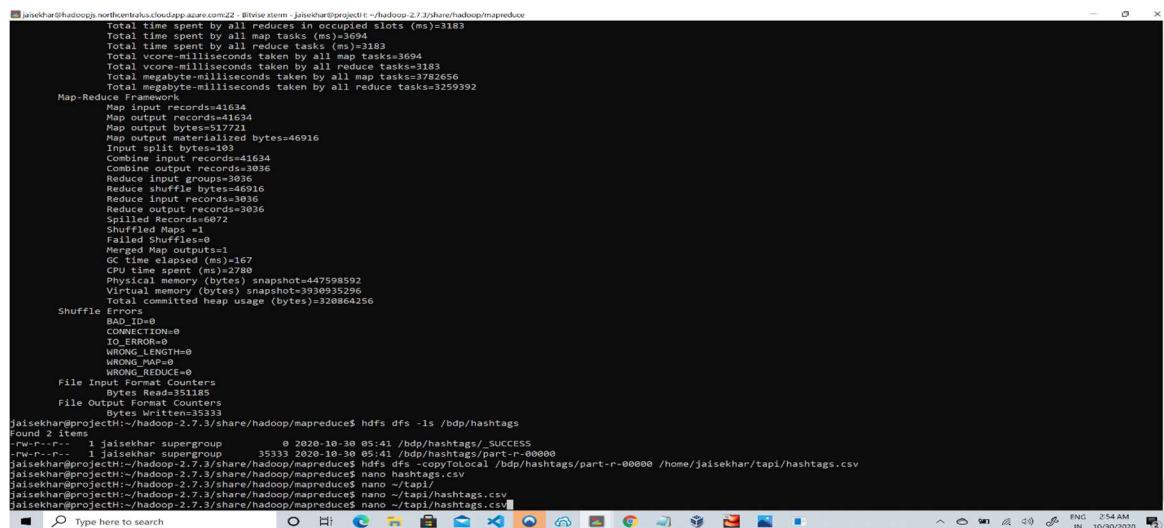
```
hadoop jar hadoop-mapreduce-examples-2.7.3.jar wordcount /bdp/hashtags.txt /bdp/hashtags
```



```
jaisekhar@projectH:/tapi$ hadoop jar hadoop-mapreduce-examples-2.7.3.jar wordcount /bdp/hashtags.txt /bdp/hashtags
20/10/30 05:40:50 INFO mapreduce.Job:  running job: job_1603952366868_0023
20/10/30 05:40:50 INFO mapreduce.Job:  number of splits:1
20/10/30 05:40:50 INFO mapreduce.JobSubmitter:  Connecting to ResourceManager at /0.0.0.0:8020
20/10/30 05:40:50 INFO mapreduce.JobSubmitter:  Starting token process :1
20/10/30 05:40:50 INFO mapreduce.JobSubmitter:  Submitted tokens for job: job_1603952366868_0023
20/10/30 05:40:50 INFO YarnClientImpl: Submitted application application_1603952366868_0023
20/10/30 05:40:50 INFO mapreduce.Job:  The url to track the job: http://projectH:8088/proxy/application_1603952366868_0023/
20/10/30 05:40:50 INFO mapreduce.Job:  Running job: job_1603952366868_0023
```

After the wordcount using the mapreduce, the saved output was sent to local for future analysis using the below commands.

```
hdfs dfs -copyToLocal /bdp/hashtags/part-r-00000 /home/jaisekhar/tapi/hashtags.csv
```



```
jaisekhar@projectH:/tapi$ hdfs dfs -ls /bdp/hashtags
Found 2 items
-rw-r--r-- 1 jaisekhar supergroup 35333 2020-10-30 05:41 /bdp/hashtags/part-r-00000
jaisekhar@projectH:/tapi$ hdfs dfs -copyToLocal /bdp/hashtags/part-r-00000 /home/jaisekhar/tapi/hashtags.csv
jaisekhar@projectH:/tapi$ cat /home/jaisekhar/tapi/hashtags.csv
jaisekhar@projectH:/tapi$ hdfs dfs -ls /bdp/hashtags
jaisekhar@projectH:/tapi$ hdfs dfs -copyToLocal /bdp/hashtags/nano /home/jaisekhar/tapi/hashtags.csv
jaisekhar@projectH:/tapi$ hdfs dfs -ls /bdp/hashtags
jaisekhar@projectH:/tapi$ hdfs dfs -copyToLocal /bdp/hashtags/nano /home/jaisekhar/tapi/hashtags.csv
```

After performing the wordcount the output is like below where it shows the hashtag and their number of occurrences.

```
jaisekhar@hadoops:~$ hadoop jar /home/jaisekhar/tapi/hadoop-mapreduce-2.7.3/share/hadoop/mapreduce/hadoop-mapreduce-wordcount-2.7.3.jar -D mapreduce.map.memory.mb=512 -D mapreduce.map.java.opts=-Xms512m -D mapreduce.map.java.opts=-Xmx512m -D mapreduce.reduce.memory.mb=1024 -D mapreduce.reduce.java.opts=-Xms1024m -D mapreduce.reduce.java.opts=-Xmx1024m -D mapreduce.map.java.library.path=/home/jaisekhar/tapi/hadoop-mapreduce-2.7.3/share/hadoop/mapreduce/lib -D mapreduce.reduce.java.library.path=/home/jaisekhar/tapi/hadoop-mapreduce-2.7.3/share/hadoop/mapreduce/lib /home/jaisekhar/tapi/hashtags.csv
jaisekhar@hadoops:~$
```

## • Sentiment Analysis using Dictionary:

To perform sentiment analysis, we are using AFINN dictionary which contains more 2.5K words and values of polarity which ranges from -5.0 to +5.0 (Negative to Positive).

Initially, we separated the word using split function on text data in tweets\_raw table and saved into split\_words table using below command.

```
jaisekhar@hadoops:~$ hadoop jar /home/jaisekhar/tapi/hadoop-mapreduce-2.7.3/share/hadoop/mapreduce/hadoop-mapreduce-wordcount-2.7.3.jar -D mapreduce.map.memory.mb=512 -D mapreduce.map.java.opts=-Xms512m -D mapreduce.map.java.opts=-Xmx512m -D mapreduce.reduce.memory.mb=1024 -D mapreduce.reduce.java.opts=-Xms1024m -D mapreduce.reduce.java.opts=-Xmx1024m -D mapreduce.map.java.library.path=/home/jaisekhar/tapi/hadoop-mapreduce-2.7.3/share/hadoop/mapreduce/lib -D mapreduce.reduce.java.library.path=/home/jaisekhar/tapi/hadoop-mapreduce-2.7.3/share/hadoop/mapreduce/lib /home/jaisekhar/tapi/hashtags.csv
jaisekhar@hadoops:~$
```

## The split words in split\_words table will look like below

```
jaisekhar@hadoop01:~$ hadoop fs -ls /user/hive/warehouse/split_words
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1603952366868_0006, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0006/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0006
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 0
2020-10-30 03:46:03,543 Stage-1 map = 0%, reduce = 0%
2020-10-30 03:47:03,296 Stage-1 map = 36%, reduce = 0%, Cumulative CPU 27.72 sec
2020-10-30 03:47:06,514 Stage-1 map = 75%, reduce = 0%, Cumulative CPU 32.68 sec
2020-10-30 03:47:15,311 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 37.73 sec
MapReduce Total cumulative CPU time: 37 seconds 730 msec
Ended Job = job_1603952366868_0006
Stage-4 is filtered out by condition resolver.
Stage-3 is selected by condition resolver.
Stage-5 is filtered out by condition resolver.
Launching Job 3 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1603952366868_0007, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0007/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0007
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0
2020-10-30 03:47:29,759 Stage-3 map = 0%, reduce = 0%
2020-10-30 03:47:31,421 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 5.04 sec
MapReduce total cumulative CPU time: 5 seconds 40 msec
Ended Job = job_1603952366868_0007
Moving data to directory hdfs://localhost:9000/user/hive/warehouse/split_words
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 37.73 sec HDFS Read: 457361250 HDFS Write: 11972221 SUCCESS
Stage-Stage-3: Map: 1 Cumulative CPU: 5.04 sec HDFS Read: 11974063 HDFS Write: 9918572 SUCCESS
Total MapReduce CPU Time Spent: 42 seconds 770 msec
OK
Time taken: 69.486 seconds
hive> describe split_words;
OK
id          bigint
words        array<string>
Time taken: 0.074 seconds, Fetched: 2 row(s)
hive> select *from split_words limit 5;
OK
1313630114157010945  ["@EyoeftheStormZ","I","got","some","120c","for","10/23","today","I","think","people","are","hoping","aapl","goes","","up","and","it","did","look","strong","before","the","dump","today"]
1313629620298747905  ["RT","@StockHighAlertz:$OPTI","up",+18.75%,"OPTEC","International","Inc.","(OPTI)","to","Announce","Q1","Interim","Financial","Statement","Pre-Market","Wednesd
ay","Octo."]
1313629506297421284  ["$AAPL","lots","of","option","action","https://t.co/S1kbk8pASS"]
1313629440488869888  ["$TPTN","@thomasSci","the","third","largest","scientific","distributor","in","the","nation","about","to","launch","sales","of","QuikLAB","and
","SANIQuik","$"]
1313629305621155840  ["Apple","iPhone","launch","event","coming","October","13:","What","to","expect","$AAPL","https://t.co/SL2nrg05Tu"]
Time taken: 0.101 seconds, Fetched: 5 row(s)
hive>
```

Now, we are splitting each word in the array as the new row. For this, we are using explode function which extracts the element from array into new row. Since it got some limitations we are using LATERAL VIEW. The output also shown below.

```
jaisekhar@hadoop01:~$ hadoop fs -ls /user/hive/warehouse/tweet_word
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1603952366868_0008, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0008/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0008
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2020-10-30 03:49:47,469 Stage-1 map = 0%, reduce = 0%
2020-10-30 03:49:54,984 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.94 sec
MapReduce Total cumulative CPU time: 3 seconds 940 msec
Ended Job = job_1603952366868_0008
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to directory hdfs://localhost:9000/user/hive/warehouse/_hive-staging_hive_2020-10-30_03-49-48_342_4677166674228850463-1-ext-10001
Moving data to directory hdfs://localhost:9000/user/hive/warehouse/tweet_word
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Cumulative CPU: 3.94 sec HDFS Read: 9923285 HDFS Write: 29791380 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 940 msec
OK
Time taken: 15.885 seconds
hive> describe tweet_word;
OK
id          bigint
word        string
Time taken: 0.048 seconds, Fetched: 2 row(s)
hive> select *from tweet_word limit 5;
OK
1313630114157010945  @EyoeftheStormZ
1313630114157010945  I
1313630114157010945  got
1313630114157010945  some
1313630114157010945  120c
Time taken: 0.107 seconds, Fetched: 5 row(s)
hive>
```

We got every word in new rows. Now, we will create new table named dictionary and will load the AFINN dictionary data into that table as shown below.

```
jaisekhar@hadoopjs:~/hadoop$ hadoop fs -ls /user/hive/warehouse/tweet_word
Found 1 items
-rw-r--r-- 1 jaisekhar supergroup 1024 2020-10-30 03:49 /user/hive/warehouse/tweet_word/_SUCCESS

jaisekhar@hadoopjs:~/hadoop$ hive -e "
    create table dictionary as
    select * from tweet_word limit 5;
"
OK
Time taken: 15.885 seconds
hive> describe dictionary;
OK
id          bigint
word        string
Time taken: 0.048 seconds, Fetched: 2 row(s)
hive> select *from dictionary limit 5;
OK
1313630114157810945 @EyeoftheStormZ
1313630114157810945 I
1313630114157810945 got
1313630114157810945 some
1313630114157810945 128c
Time taken: 0.107 seconds, Fetched: 5 row(s)
hive> create table dictionary(word string,rating int) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';
OK
Time taken: 0.062 seconds
hive> LOAD DATA INPATH '/AFINN.txt' into TABLE dictionary;
FAILED: SemanticException Line 1:17 Invalid path ''/AFINN.txt'': No files matching path hdfs://localhost:9000/AFINN.txt
hive> LOAD DATA LOCAL INPATH '/AFINN.txt' into TABLE dictionary;
FAILED: SemanticException Line 1:23 Invalid path ''/AFINN.txt'': No files matching path file:/AFINN.txt
hive> LOAD DATA LOCAL INPATH '~/tapi/AFINN.txt' into TABLE dictionary;
FAILED: SemanticException Line 1:23 Invalid path '~/tapi/AFINN.txt'': No files matching path file:/home/jaisekhar/hive/~/tapi/AFINN.txt
hive> LOAD DATA LOCAL INPATH './tapi/AFINN.txt' into TABLE dictionary;
Loading data to table default.dictionary
OK
Time taken: 0.308 seconds
hive> select *from dictionary;
OK
Time taken: 0.084 seconds, Fetched: 5 row(s)
hive>
```

Create new table word\_join by joining dictionary and tweet\_word tables as shown below.

```
jaisekhar@hadoopjs:~/hadoop$ hive -e "
    create table word_join as
    select tweet_word.word, dictionary.rating
    from tweet_word
    LEFT OUTER JOIN dictionary
    ON(tweet_word.word = dictionary.word);
"
OK
Time taken: 0.084 seconds
hive> select *from word_join;
OK
abandon -2
abandoned -2
abandons -2
abducted -2
abduction -2
Time taken: 0.084 seconds, Fetched: 5 row(s)
hive> create table word_join as select tweet_word.word,dictionary.rating from tweet_word LEFT OUTER JOIN dictionary ON(tweet_word.word = dictionary.word);
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = jaisekhar_20201030035549_36795561_d671-4285-8fd1-2dbbf3cf3348
Total jobs = 1
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/jaisekhar/hive/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/jaisekhar/hadoop-2.7.3/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
2020-10-30 03:55:14  Starting to map local tasks to process map join;           maximum memory = 477626368
2020-10-30 03:55:15   Dump the reduce-table for tag: 1 with group count: 2477 into file: file:/tmp/jaisekhar/1219b18d-06d9-4391-8622-9e6e48671738/hive_2020-10-30_03-55-49_214_727643782734096
4210_1-local_10004/HashTable-Stage-4/MapJoin-mapfile01...-hashtable
2020-10-30 03:55:17   Uploaded 1 File to: file:/tmp/jaisekhar/1219b18d-06d9-4391-8622-9e6e48671738/hive_2020-10-30_03-55-49_214_727643782734096210-1-local-10004/HashTable-Stage-4/MapJoin-mapfile01...-hashtable (69200 bytes)
2020-10-30 03:55:17   End of local task; Time Taken: 1.193 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1603952366868_0009, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0009/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0009
Hadoop job information for Stage-4: number of mappers: 1; number of reducers: 0
2020-10-30 03:56:04.874 Stage-4 map = 0%, reduce = 0%
2020-10-30 03:56:15.656 Stage-4 map = 100%, reduce = 0%, Cumulative CPU 8.41 sec
MapReduce Total cumulative CPU time: 8 seconds 410 msec
Ended Job = job_1603952366868_0009
Moving data to directory hdfs://localhost:9000/user/hive/warehouse/word_join
MapReduce Jobs Launched:
Stage-Stage-4: Map: 1  Cumulative CPU: 8.41 sec  HDFS Read: 29797484 HDFS Write: 33104055 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 410 msec
OK
Time taken: 27.641 seconds
hive>
```

After joining now, we got the polarity value for each tweet in the word\_join table which is shown below.

```
jaisekhar@hadoop3:~$ hive
hive> select id,AVG(rating) as rating from word_join GROUP BY word_join.id order by rating DESC limit 10;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = jaisekhar_20201030035831_b1683b39-dfce-45a3-9d6c-39e41e47cc55
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1603952366868_0010, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0010/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0010
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-10-30 03:58:39,826 Stage-1 map = 0%, reduce = 0%
2020-10-30 03:58:49,726 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 9.09 sec
2020-10-30 03:58:58,235 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 14.14 sec
MapReduce Total cumulative CPU time: 14 seconds 140 msec
Ended Job = job_1603952366868_0010
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1603952366868_0011, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0011/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0011
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2020-10-30 03:59:12,128 Stage-2 map = 0%, reduce = 0%
2020-10-30 03:59:16,652 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 4.17 sec
2020-10-30 03:59:26,045 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 6.46 sec
MapReduce Total cumulative CPU time: 6 seconds 460 msec
Ended Job = job_1603952366868_0011
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 14.14 sec HDFS Read: 33112130 HDFS Write: 2333576 SUCCESS
Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 6.46 sec HDFS Read: 2339005 HDFS Write: 447 SUCCESS
Total MapReduce CPU Time Spent: 20 seconds 600 msec
OK
1320066591153938438 5.0
1318279840190005248 5.0
1318000560465903616 5.0
1315042659937984520 5.0
1315279419041345537 5.0
1320106069327302664 5.0
1312466814400565248 5.0
13115451109585241600 4.0
1311544751712137217 4.0
131164499275538691 4.0
Time taken: 55.286 seconds, Fetched: 10 row(s)
hive> create table testjoin as select id,AVG(rating) as polarity from word_join GROUP BY word_join.id;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = jaisekhar_20201030040157_l5624f5e-f8ce-47fe-a731-b4ffe5253835
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1603952366868_0012, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0012/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0012
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-10-30 04:02:04,689 Stage-1 map = 0%, reduce = 0%
2020-10-30 04:02:15,310 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 9.12 sec
2020-10-30 04:02:23,908 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 13.73 sec
MapReduce Total cumulative CPU time: 13 seconds 730 msec
Ended Job = job_1603952366868_0012
Moving data to directory hdfs://localhost:9000/user/hive/warehouse/testjoin
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 13.73 sec HDFS Read: 33112656 HDFS Write: 1944460 SUCCESS
Total MapReduce CPU Time Spent: 13 seconds 730 msec
OK
Time taken: 28.087 seconds
hive>
```

Now, we are saving the same id based grouped data into the testjoin table.

```
jaisekhar@hadoop3:~$ hive
hive> select id,AVG(rating) as rating from word_join GROUP BY word_join.id order by rating DESC limit 10;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = jaisekhar_20201030035831_b1683b39-dfce-45a3-9d6c-39e41e47cc55
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1603952366868_0010, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0010/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0010
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-10-30 03:59:12,128 Stage-1 map = 0%, reduce = 0%
2020-10-30 03:59:19,652 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.17 sec
2020-10-30 03:59:26,045 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 6.46 sec
MapReduce Total cumulative CPU time: 6 seconds 460 msec
Ended Job = job_1603952366868_0010
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 14.14 sec HDFS Read: 33112130 HDFS Write: 2333576 SUCCESS
Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 6.46 sec HDFS Read: 2339005 HDFS Write: 447 SUCCESS
Total MapReduce CPU Time Spent: 20 seconds 600 msec
OK
1320066591153938438 5.0
1318279840190005248 5.0
1318000560465903616 5.0
1315042659937984520 5.0
1315279419041345537 5.0
1320106069327302664 5.0
1312466814400565248 5.0
13115451109585241600 4.0
1311544751712137217 4.0
131164499275538691 4.0
Time taken: 55.286 seconds, Fetched: 10 row(s)
hive> create table testjoin as select id,AVG(rating) as polarity from word_join GROUP BY word_join.id;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = jaisekhar_20201030040157_l5624f5e-f8ce-47fe-a731-b4ffe5253835
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1603952366868_0012, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0012/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0012
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-10-30 04:02:04,689 Stage-1 map = 0%, reduce = 0%
2020-10-30 04:02:15,310 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 9.12 sec
2020-10-30 04:02:23,908 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 13.73 sec
MapReduce Total cumulative CPU time: 13 seconds 730 msec
Ended Job = job_1603952366868_0012
Moving data to directory hdfs://localhost:9000/user/hive/warehouse/testjoin
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 13.73 sec HDFS Read: 33112656 HDFS Write: 1944460 SUCCESS
Total MapReduce CPU Time Spent: 13 seconds 730 msec
OK
Time taken: 28.087 seconds
hive>
```

At last, we are joining the tweets\_raw table and test join table based on id and created new table ‘tweets’.

```

hive> create table tweets as select tweets_raw.* ,nvl(testjoin.polarity,0) as polarity from tweets_raw LEFT OUTER JOIN testjoin ON(tweets_raw.id =testjoin.id);
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = jaisekhar_20201030044201_610f07f8-be5f-4c87-b4b2-9a364ba55d51
Total jobs = 1
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/jaisekhar/hive/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/jaisekhar/hadoop-2.7.3/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
2020-10-30 04:42:08 Starting to launch local task to process map join; maximum memory = 477626368
2020-10-30 04:42:10 Dump the side-table for tag: 1 with group count: 83122 into file: file:/tmp/jaisekhar/1219b18d-06d9-4391-8622-9e6e48671738/hive_2020-10-30_04-42-01_928_13949070656869
72736-1/-local-10004/HashTable-Stage-4/MapJoin-mapfile31--.hashtable
2020-10-30 04:42:10 Uploaded 1 file to: file:/tmp/jaisekhar/1219b18d-06d9-4391-8622-9e6e48671738/hive_2020-10-30_04-42-01_928_1394907065686972736-1/-local-10004/HashTable-Stage-4/MapJoin
-mapfile31--.hashtable (2405610 bytes)
2020-10-30 04:42:10 End of local task; Time Taken: 1.906 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1603952366868_0018, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0018/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0018
Hadoop Job Information for Stage-4: number of mappers: 2; number of reducers: 0
2020-10-30 04:42:10 Stage-4 map = 0%, reduce = 0%
2020-10-30 04:42:41 884 Stage-4 map = 36%, reduce = 0%, Cumulative CPU 27.81 sec
2020-10-30 04:42:45 920 Stage-4 map = 75%, reduce = 0%, Cumulative CPU 31.14 sec
2020-10-30 04:42:49 216 Stage-4 map = 100%, reduce = 0%, Cumulative CPU 34.91 sec
MapReduce Total cumulative CPU time: 34 seconds 910 msec
Ended Job = job_1603952366868_0018
Moving data to directory hdfs://localhost:9000/user/hive/warehouse/tweets
MapReduce Jobs Launched:
Stage-Stage-4: Map: 2 Cumulative CPU: 34.91 sec HDFS Read: 457377858 HDFS Write: 36222244 SUCCESS
Total MapReduce CPU Time Spent: 34 seconds 910 msec
OK
Time taken: 48.472 seconds
hive> select * from tweets limit 10;
OK
1313630114157018945 Tue Oct 06 23:59:59 +0000 2020 <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a> false 0 NULL {"urls":[],"user_mentions":[{"screen_name":"EyeoftheStormZ","name":"cr8ig"},{}], "hashtags":[]}&#xA0; @EyeoftheStormZ I got some 120c for 10/23 today, I think people are "hoping" aapl goes up and it did look strong before the dump today {"screen_name":"house_money","name":"House Money","friends_count":583,"followers_count":121,"statuses_count":1397,"verified":false,"utc_offset":null,"time_zone":null}
} EyeoftheStormZ NULL NULL NULL NULL 0.5
1313629520298747905 Tue Oct 06 23:58:02 +0000 2020 <a href="http://twitter.com/#!/download/ipad" rel="nofollow">Twitter for iPad</a> false 0 {"text":">$OPTI up +18.75% OPTEC International, Inc. (OPTI) to Announce Q1 Interim Financial Statement Pre-Market Wednesday 0.. https://t.co/czegEdBhwX","user":{"screen_name":"StockHighAlertz","name":"Stocks On HIGH ALERTZ"}}, {"urls":[],"user_mentions":[{"screen_name":"StockHighAlertz","name":"Stocks On HIGH ALERTZ"}], "hashtags":[]}&#xA0; RT @StockHighAlertz: OPTI up +18.75% OPTEC International, Inc. (OPTI) to Announce Q1 Interim Financial Statement Pre-Market Wednesday Octo... {"screen_name":"Investoon","name":"Broke Investor","friends_count":1988,"followers_count":404,"statuses_count":1597,"verified":false,"utc_offset":null,"time_zone":null}
NULL NULL NULL NULL NULL 0.0
1313629506297421824 Tue Oct 06 23:57:35 +0000 2020 <a href="https://mobile.twitter.com" rel="nofollow">Twitter Web App</a> false 0 NULL {"urls":[],"user_mentions":[]}, "hashtags":[]

Type here to search

```

Below, we can see sample data which the results show the tweet text and polarity value. This is one of the important part in this project. If you see the first tweets, it says holding the AAPL for overnight is risky hence it showed negative polarity value.

```

hive> select distinct(text),polarity from finaltweets where polarity is not null limit 10;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = jaisekhar_20201030040926_65b52753-6dc4-49ea-ac1f-5293b2c25413
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1603952366868_0014, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0014/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0014
Hadoop Job Information for Stage-1: number of mappers: 1; number of reducers: 1
2020-10-30 04:09:35,503 Stage-1 map = 0%, reduce = 0%
2020-10-30 04:09:42,984 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.36 sec
2020-10-30 04:09:49,405 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 4.36 sec
MapReduce Total cumulative CPU time: 4 seconds 360 msec
Ended Job = job_1603952366868_0014
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 6.95 sec HDFS Read: 36170628 HDFS Write: 1696 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 950 msec
OK
!ALERT! Bought $AAPL $126c @ $1.95 with some profits from earlier. Holding overnight going to risk it for a gap up into the event tomorrow. -2.0
!ALERT! Bought $AAPL $127.50c @ $1.55 with some profits from earlier. Holding overnight going to risk it for a gap... https://t.co/imjIWbod0D -2.0
. . . just like the Aug turn into Sep expiry (Note: the same thing would occur if the client were to unwind the... https://t.co/p2pu1TSPqm 2.0
" Hope Investors" in ITC don't get it! Contrarian Investing doesn't work this way: Identify beaten down names.Wait f... https://t.co/oV7yQnubE -2.0
"Alerted live in chat with entry and exit. A cool 100% gain on this one and letting the runners go to work..." https://t.co/oZuun528Xk 1.5
"Alerted live in chat. Solid discipline. Give the stock time to play out your plan. This one never hit the stop so... https://t.co/1bfuI95mtF -1.0
"Alerted live in chat. Solid discipline. Give the stock time to play out your plan. This one never hit the stop so... https://t.co/bhglsSwe40 -1.0
"Alerted live in chat. This one had multiple chances to buy low and sell for profit. Getting the right direction is... https://t.co/zgZI94fqqs 2.0
"Alerted live in chat. This one made a solid move up for the rest of the day. Don't forget to take profit... https://t.co/TsjxMsnnNs 0.5
"Alerted live in chat. You would have had multiple chances to enter lower than our analyst. The important part of o... https://t.co/tknf28QxRl 2.0
Time taken: 24.68 seconds, Fetched: 10 row(s)
hive>

```

- Hive Analysis:

Below are the Queries for analysis using Hive.

- The below simple query shows number of records in the tweets table.

```
select count (*) from tweets;
```

```
jaisekar@hadoop:~$ hadoop fs -ls /user/jaisekar/project/tweets
[jaisekar@hadoop:~]$ hive -e "SELECT COUNT(*) FROM tweets"
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1603952366868_0019, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0019/
Kill Command = /home/jaisekar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0019
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-10-30 04:44:08,126 Stage-1 map = 0%, reduce = 0%
2020-10-30 04:44:15,630 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 3.67 sec
2020-10-30 04:44:22,013 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 5.92 sec
MapReduce Total cumulative CPU time: 5 seconds 920 msec
Ended Job = job_1603952366868_0019
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 5.92 sec HDFS Read: 3623524 HDFS Write: 106 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 920 msec
OK
238087
Time taken: 23.847 seconds, Fetched: 1 row(s)
hive: |
```

Type here to search ENG IN 11:44 PM 10/29/2020

- The below query shows the screenname of the user and their followers count in the descending order in whole dataset. In this dataset, Apple user has the most number of followers.

```
jaisekar@hadoop:~$ hive -e "SELECT DISTINCT user.screen_name AS name , user.followers_count AS count
> FROM tweets
> WHERE size(entities.hashtags) > 0
> ORDER BY count DESC
> LIMIT 10"
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1603952366868_0024, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0024/
Kill Command = /home/jaisekar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0024
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-10-30 05:42:34,491 Stage-1 map = 0%, reduce = 0%
2020-10-30 05:42:34,491 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.92 sec
2020-10-30 05:42:50,511 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 7.86 sec
MapReduce Total cumulative CPU time: 7 seconds 860 msec
Ended Job = job_1603952366868_0024
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1603952366868_0025, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0025/
Kill Command = /home/jaisekar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0025
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2020-10-30 05:43:08,212 Stage-2 map = 0%, reduce = 0%
2020-10-30 05:43:08,212 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 1.44 sec
2020-10-30 05:43:10,052 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 3.93 sec
2020-10-30 05:43:17,052 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 3.93 sec
MapReduce Total cumulative CPU time: 3 seconds 930 msec
Ended Job = job_1603952366868_0025
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 7.86 sec HDFS Read: 3623574 HDFS Write: 87747 SUCCESS
Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 3.93 sec HDFS Read: 93131 HDFS Write: 232 SUCCESS
Total MapReduce CPU Time Spent: 11 seconds 790 msec
OK
Apple 5462984
thestreet 736435
Nasdaq 608042
hive: |
```

Type here to search ENG IN 12:44 AM 10/30/2020

- Using the below query, we are fetching the user screennames and number of tweets tweeted by them in the descending order. In this dataset, TIN-Tech Bloggers has tweeted more about AAPL.

```
jaisekhar@hadoop-northcentralus.cloudapp.azure.com:22 - BlvVis xterm - jaisekhar@projectt ~$hive
hive> SELECT user.name, user.screen_name, count(1) as cc
   > FROM tweets
   > WHERE text like "%AAPL%" and user.name is not null
   > GROUP BY user.name,user.screen_name
   > ORDER BY cc desc limit 5;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = jaisekhar_20281030054655_54084153-d322-45d4-a2ef-ba68b25a3df5
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1603952366868_0028, Tracking URL = http://projectt:8088/proxy/application_1603952366868_0028/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0028
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-10-30 05:47:17,480 Stage-1 map = 0%, reduce = 0%
2020-10-30 05:47:17,480 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.56 sec
2020-10-30 05:47:17,480 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 7.72 sec
MapReduce Total cumulative CPU time: 7 seconds 729 msec
Ended Job = job_1603952366868_0028
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1603952366868_0029, Tracking URL = http://projectt:8088/proxy/application_1603952366868_0029/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0029
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2020-10-30 05:47:37,636 Stage-2 map = 0%, reduce = 0%, Cumulative CPU 2.43 sec
2020-10-30 05:47:37,636 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 2.43 sec
2020-10-30 05:47:44,061 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 4.77 sec
MapReduce Total cumulative CPU time: 4 seconds 770 msec
Ended Job = job_1603952366868_0029
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 7.72 sec HDFS Read: 36236574 HDFS Write: 520124 SUCCESS
Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 4.77 sec HDFS Read: 525896 HDFS Write: 384 SUCCESS
Total MapReduce CPU Time Spent: 12 seconds 490 msec
OK
TIN-Tech Bloggers      TINTechBloggers 477
FinBuzz PortfolioBuzz 442
Investor News  newsfilterio  435

```

- In the below query, we are getting the number of positive tweets which contains the keyword 'Iphone'. There are 234 positive tweets about Iphone.

```
jaisekhar@hadoop-northcentralus.cloudapp.azure.com:22 - BlvVis xterm - jaisekhar@projectt ~$hive
hive> select count(*) From tweets where text like '%Iphone%' or text like '%iphone%' and polarity > 0.0;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = jaisekhar_2028103005519_f1d087d9-fe6f-428f-b901-578898c97a34
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1603952366868_0031, Tracking URL = http://projectt:8088/proxy/application_1603952366868_0031/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0031
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-10-30 05:57:27,002 Stage-1 map = 0%, reduce = 0%
2020-10-30 05:57:34,389 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.16 sec
2020-10-30 05:57:41,839 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 7.0 sec
MapReduce Total cumulative CPU time: 7 seconds 0 msec
Ended Job = job_1603952366868_0031
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 7.0 sec HDFS Read: 36236745 HDFS Write: 103 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 0 msec
OK
234
Time taken: 23.217 seconds. Fetched: 1 row(s)
hive> select count(*) From tweets where text like '%event%' or '%Event%' and polarity < 0.0;
FAILED: ClassCastException org.apache.hadoop.hive.serde2.objectinspector.primitive.WritableConstantStringObjectInspector cannot be cast to org.apache.hadoop.hive.serde2.objectinspector.primitive.BooleanObjectInspector
hive> select count(*) From tweets where text like '%event%' or text like '%Event%' and polarity < 0.0;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = jaisekhar_2028103006055_a108454d-ee0c-4083-a01b-252584c0960a
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
```

- Here, we are getting the number of negative tweets which contains the keyword 'Event' and there are 1462 tweets in this dataset.

```
jaisekhar@hadoopjs:~/hive$ bin/hive -e "SELECT count(*) FROM tweets WHERE text like '%event%' or '%Event%' and polarity < 0;" 
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1603952366868_0031, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0031/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0031
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-10-30 05:57:27,002 Stage-1 map = 0%, reduce = 0%
2020-10-30 05:57:34,389 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.16 sec
2020-10-30 05:57:41,839 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 7.0 sec
MapReduce Total cumulative CPU time: 7 seconds 0 msec
Ended Job = job_1603952366868_0031
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1   Cumulative CPU: 7.0 sec   HDFS Read: 36236745 HDFS Write: 103 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 0 msec
OK
234
Time taken: 23.217 seconds, Fetched: 1 row(s)
hive> select count(*) FROM tweets WHERE text like '%event%' or '%Event%' and polarity < 0;
FAILED: ClassCastException org.apache.hadoop.hive.serde2.objectinspector.primitive.WritableConstantStringObjectInspector cannot be cast to org.apache.hadoop.hive.serde2.objectinspector.primitive.BooleanObjectInspector
hive> select count(*) FROM tweets WHERE text like '%event%' or text like '%Event%' and polarity < 0;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = jaisekhar_20201030060055_a108454d-ee0c-4083-a01b-252584c0960a
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set mapreduce.job.reducers=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1603952366868_0032, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0032/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0032
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-10-30 06:01:02,514 Stage-1 map = 0%, reduce = 0%
2020-10-30 06:01:10,965 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 5.35 sec
2020-10-30 06:01:18,430 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 7.9 sec
MapReduce Total cumulative CPU time: 7 seconds 900 msec
Ended Job = job_1603952366868_0032
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1   Cumulative CPU: 7.9 sec   HDFS Read: 36236729 HDFS Write: 104 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 900 msec
OK
1462
Time taken: 24.028 seconds, Fetched: 1 row(s)
hive>
```

- The below basic query returns the created time, source of the tweet, screenname, username and their polarities.

```
jaisekhar@hadoopjs:~/hive$ bin/hive -e "SELECT created_at,source,year,day FROM tweets LIMIT 10;"
Time taken: 0.039 seconds, Fetched: 15 row(s)
hive> select created_at,source,year,day FROM tweets LIMIT 10;
OK
Tue Oct 06 23:59:59 +0000 2020 NULL NULL NULL
Tue Oct 06 23:58:02 +0000 2020 NULL NULL NULL
Tue Oct 06 23:57:32 +0000 2020 NULL NULL NULL
Tue Oct 06 23:56:12 +0000 2020 NULL NULL NULL
Tue Oct 06 23:56:07 +0000 2020 NULL NULL NULL
Tue Oct 06 23:56:30 +0000 2020 NULL NULL NULL
Tue Oct 06 23:56:11 +0000 2020 NULL NULL NULL
Tue Oct 06 23:56:02 +0000 2020 NULL NULL NULL
Tue Oct 06 23:55:51 +0000 2020 NULL NULL NULL
NULL NULL NULL
Time taken: 0.081 seconds, Fetched: 10 row(s)
hive> insert overwrite local directory '/home/jaisekhar/tapi/savedata' row format delimited fields terminated by ',' select created_at,source,retweeted_count,user.screen_name,user.name,polarity from tweets;
FAILED: SemanticException [Error 10004]: Line 1:136 Invalid table alias or column reference 'retweeted_count': (possible column names are: id, created_at, source, favorited, retweet_count, retweeted_status, entities, text, user, in_reply_to_screen_name, year, month, day, hour, polarity)
hive> insert overwrite local directory '/home/jaisekhar/tapi/savedata' row format delimited fields terminated by ',' select created_at,source,retweet_count,user.screen_name,user.name,polarity from tweets;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = jaisekhar_20201030044930_22e0a15c-4826-48b1-acba-1ceede010cfc0
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1603952366868_0020, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0020/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0020
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2020-10-30 04:49:38,086 Stage-1 map = 0%, reduce = 0%
2020-10-30 04:49:45,464 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.12 sec
MapReduce Total cumulative CPU time: 4 seconds 120 msec
Ended Job = job_1603952366868_0020
Moving data to local directory /home/jaisekhar/tapi/savedata
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 4.12 sec   HDFS Read: 36229022 HDFS Write: 16707511 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 120 msec
OK
Time taken: 17.678 seconds
hive> select created_at,source,retweet_count,user.screen_name,user.name,polarity from tweets limit 10;
OK
Tue Oct 06 23:59:59 +0000 2020 <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a> 0 house_money_ House Money 0.5
Tue Oct 06 23:58:02 +0000 2020 <a href="http://twitter.com/#/download/ipad" rel="nofollow">Twitter for iPad</a> 0 InvestOp Broke Investor 0.0
Tue Oct 06 23:57:35 +0000 2020 <a href="https://mobile.twitter.com" rel="nofollow">Twitter Web App</a> 0 sunchartist Sunchartist 0.0
Tue Oct 06 23:57:39 +0000 2020 <a href="https://mobile.twitter.com" rel="nofollow">Twitter Web App</a> 0 PennyPresident Penny President 0.0
Tue Oct 06 23:56:47 +0000 2020 <a href="https://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a> 0 notstockadvice3 Alex 0.0
Tue Oct 06 23:56:30 +0000 2020 <a href="https://mobile.twitter.com" rel="nofollow">Twitter Web App</a> 0 Morewealthal Susan Moore 0.0
Tue Oct 06 23:56:16 +0000 2020 <a href="https://mobile.twitter.com" rel="nofollow">Twitter Web App</a> 0 Pucktalk dailypicks -2.0
Tue Oct 06 23:56:02 +0000 2020 <a href="http://twitter.com/download/android" rel="nofollow">Twitter for Android</a> 0 MichaelJobe Michael Jobe 0.0
Time taken: 10.04 seconds, Fetched: 10 row(s)
```

And the same data is stored in other csv file which helps in later part of analysis (Cassandra, visualization)

```
jaisekhar@hadoop5:northcentralus.cloudapp.azure.com:22 - Bltwise xterm - jaisekhar@projectH: ~/hive
Time taken: 31.836 seconds. Fetched: 20 row(s)
hive> create table outtable as select created_at,source,reweet_count,user.screen_name,user.name,nvl(polarity,0) as polarity from tweets_raw join testjoin on (tweets_raw.id==testjoin.id) where created_at is not null;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = jaisekhar_20201030062834_b43e53f0-bc7e-4383-99c6-b166a1f47873
Total jobs = 1
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/jaisekhar/hive/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/jaisekhar/hadoop-2.7.3/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
2020-10-30 06:28:41 Starting to launch local task to process map join; maximum memory = 477626368
2020-10-30 06:28:42 Dump the side-table for tag: 1 with group count: 83122 into file: file:/tmp/jaisekhar/1219b18d-06d9-4391-8622-9e6e48671738/hive_2020-10-30_06:28:43_B09_4708584734984437282-1-local-10004/HashTable-Stage-4/MapJoin-Mapfile61--.hashtable
2020-10-30 06:28:43 Uploaded 1 file to: file:/tmp/jaisekhar/1219b18d-06d9-4391-8622-9e6e48671738/hive_2020-10-30_06:28:43_B09_4708584734984437282-1-local-10004/HashTable-Stage-4/MapJoin-Mapfile61--.hashtable (2405610 bytes)
2020-10-30 06:28:43 End of local task; Time Taken: 2.101 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1603952366868_0035, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0035/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0035
Hadoop job information for Stage-4: number of mappers: 2; number of reducers: 0
2020-10-30 06:28:51,663 Stage-4 map = 0%, reduce = 0%
2020-10-30 06:29:08,084 Stage-4 map = 30%, reduce = 0%, Cumulative CPU 30.52 sec
2020-10-30 06:29:19,150 Stage-4 map = 61%, reduce = 0%, Cumulative CPU 32.23 sec
2020-10-30 06:29:21,300 Stage-4 map = 75%, reduce = 0%, Cumulative CPU 33.71 sec
2020-10-30 06:29:24,393 Stage-4 map = 100%, reduce = 0%, Cumulative CPU 38.24 sec
MapReduce Total cumulative CPU time: 38 seconds 240 msec
Ended Job = job_1603952366868_0035
Moving data to directory hdfs://localhost:9000/user/hive/warehouse/outtable
MapReduce Jobs Launched:
Stage-Stage-4: Map: 2 Cumulative CPU: 38.24 sec HDFS Read: 457369370 HDFS Write: 11640818 SUCCESS
Total MapReduce CPU Time Spent: 38 seconds 240 msec
OK
Time taken: 50.801 seconds
hive> insert overwrite local directory '/home/jaisekhar/tapi/outdata' row format delimited fields terminated by ',' select * from outtable;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = jaisekhar_20201030063019_97abf186-61a3-47b9-bbda-7fde9f1ea674
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1603952366868_0036, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0036/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0036
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2020-10-30 06:30:26,955 Stage-1 map = 0%, reduce = 0%
2020-10-30 06:30:34,437 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 4.19 sec
MapReduce Total cumulative CPU time: 38 seconds 240 msec
Ended Job = job_1603952366868_0036
Moving data to directory hdfs://localhost:9000/user/hive/warehouse/outtable
MapReduce Jobs Launched:
Stage-Stage-4: Map: 2 Cumulative CPU: 38.24 sec HDFS Read: 457369370 HDFS Write: 11640818 SUCCESS
Total MapReduce CPU Time Spent: 38 seconds 240 msec
OK
Time taken: 50.801 seconds
hive> insert overwrite local directory '/home/jaisekhar/tapi/outdata' row format delimited fields terminated by ',' select * from outtable;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = jaisekhar_20201030073931_8dc5c22d-0da3-4921-8945-61633365e4e8
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1603952366868_0037, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0037/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0037
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2020-10-30 07:39:39,566 Stage-2 map = 0%, reduce = 0%
2020-10-30 07:39:47,128 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 4.48 sec
2020-10-30 07:39:53,504 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 6.76 sec
MapReduce Total cumulative CPU time: 6 seconds 760 msec
Ended Job = job_1603952366868_0037
Stage-5 is selected by condition resolver.
Stage-1 is filtered out by condition resolver.
```

- The below query shows the user's name whose tweets has maximum number of retweet count. In this dataset, Earnings Whispers tweet has more tweet count.

```
hive> select user.name, retweet_count from tweets_raw where retweet_count in (select max(retweet_count) from tweets);
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = jaisekhar_20201030073931_8dc5c22d-0da3-4921-8945-61633365e4e8
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1603952366868_0037, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0037/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0037
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2020-10-30 07:39:39,566 Stage-2 map = 0%, reduce = 0%
2020-10-30 07:39:47,128 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 4.48 sec
2020-10-30 07:39:53,504 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 6.76 sec
MapReduce Total cumulative CPU time: 6 seconds 760 msec
Ended Job = job_1603952366868_0037
Stage-5 is selected by condition resolver.
Stage-1 is filtered out by condition resolver.
```

```

[jaisekarh@hadoppj:~]$ binhive xterm -jaisekarh@project01 ~-/hive
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1603952366868_0037, Tracking URL = http://project01:8088/proxy/application_1603952366868_0037/
Kill Command = /home/jaisekarh/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0037
Hadoop job information for Stage-0: number of mappers: 1; number of reducers: 1
2020-10-30 07:39:35,650 Stage-0 map = 100%, reduce = 0%
2020-10-30 07:39:47,128 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 4.48 sec
2020-10-30 07:39:53,504 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 6.76 sec
MapReduce Total cumulative CPU time: 6 seconds 766 msec
Ended Job = job_1603952366868_0037
Stage-0 is selected by condition resolver.
Stage-1 is filtered out by condition resolver.
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/jaisekarh/hive/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/jaisekarh/hadoop-2.7.3/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
2020-10-30 07:40:02 Starting to launch local task to process map join; maximum memory = 477626368
2020-10-30 07:40:02 Dump the side-table for tag: 1 with group count: 1 into file: file:/tmp/jaisekarh/1219b18d-06d9-4391-8622-9e6e48671738/hive_2020-10-30_07-39-31_293_8745835599822287755
5-1-local-10005/HashTable-Stage-3/MapJoin-mapfile71--.hashtable
2020-10-30 07:40:03 Uploaded 1 File to: file:/tmp/jaisekarh/1219b18d-06d9-4391-8622-9e6e48671738/hive_2020-10-30_07-39-31_293_8745835599822287755-1-local-10005/HashTable-Stage-3/MapJoin
-mapfile71--.hashtable (288 bytes)
2020-10-30 07:40:03 End of local task; Time Taken: 0.891 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 3 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1603952366868_0038, Tracking URL = http://project01:8088/proxy/application_1603952366868_0038/
Kill Command = /home/jaisekarh/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0038
Hadoop job information for Stage-0: number of mappers: 2; number of reducers: 0
2020-10-30 07:40:11,676 Stage-0 map = 0%, reduce = 0%
2020-10-30 07:40:31,328 Stage-0 map = 61%, reduce = 0%, Cumulative CPU 22.63 sec
2020-10-30 07:40:34,548 Stage-0 map = 75%, reduce = 0%, Cumulative CPU 28.05 sec
2020-10-30 07:40:36,734 Stage-0 map = 100%, reduce = 0%, Cumulative CPU 29.85 sec
MapReduce Total cumulative CPU time: 29 seconds 856 msec
Ended Job = job_1603952366868_0038
MapReduce Jobs Launched:
Stage-Stage-0: Map: 1 Reduce: 1 Cumulative CPU: 6.76 sec HDFS Read: 36235131 HDFS Write: 116 SUCCESS
Stage-Stage-0: Map: 2 Cumulative CPU: 29.85 sec HDFS Read: 457367714 HDFS Write: 208 SUCCESS
Total MapReduce CPU Time Spent: 36 seconds 610 msec
OK
Earnings Whispers      775
Time taken: 67.545 seconds, Fetched: 1 row(s)
hive> 

```

- Using this query, we are getting User's name and their total number of tweets in the descending order

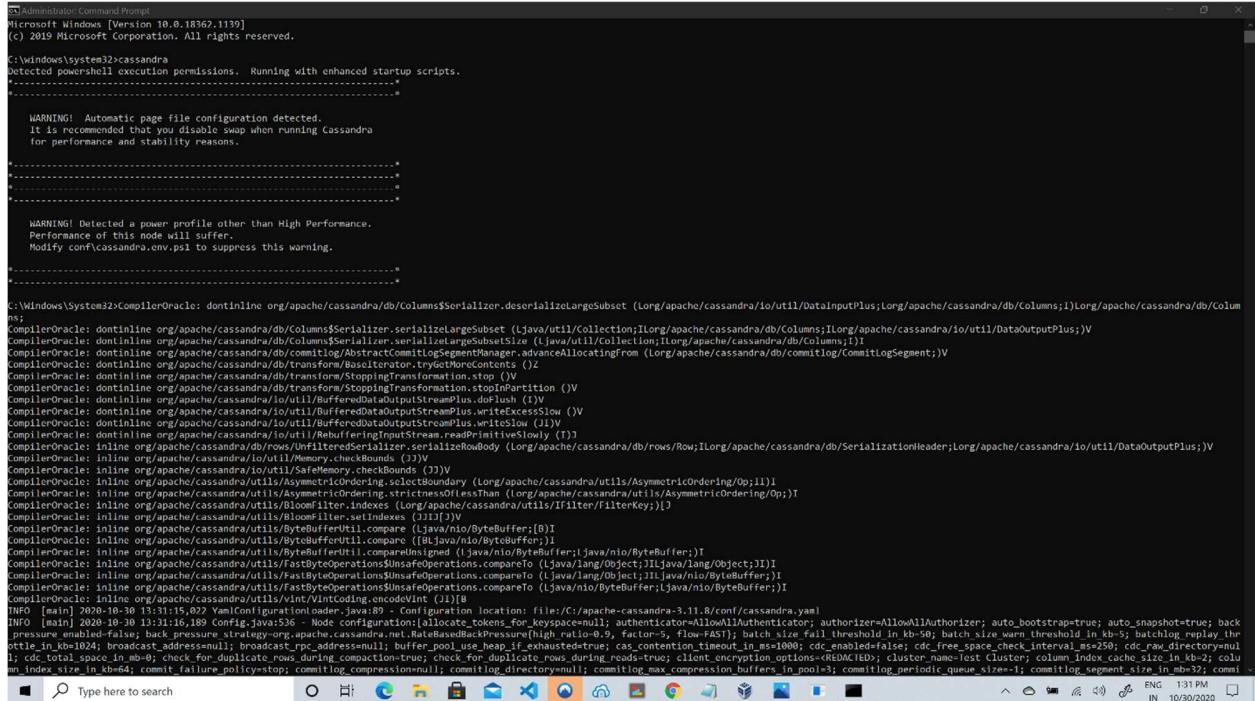
```

[jaisekarh@hadoppj:~]$ binhive xterm -jaisekarh@project01 ~-/hive
hive> select user.name,count(*) as count from tweets_raw group by user.name order by count DESC limit 10;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = jaisekarh_20201030075001_6df17d34-ba0b-4dc6-97c2-7c7c630aa97c
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 2
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1603952366868_0041, Tracking URL = http://project01:8088/proxy/application_1603952366868_0041/
Kill Command = /home/jaisekarh/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0041
Hadoop job information for Stage-0: number of mappers: 2; number of reducers: 2
2020-10-30 07:50:08,021 Stage-1 map = 0%, reduce = 0%
2020-10-30 07:50:27,576 Stage-1 map = 40%, reduce = 0%, Cumulative CPU 22.66 sec
2020-10-30 07:50:28,622 Stage-1 map = 67%, reduce = 0%, Cumulative CPU 23.58 sec
2020-10-30 07:50:30,725 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 25.73 sec
2020-10-30 07:50:39,389 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 32.46 sec
MapReduce Total cumulative CPU time: 32 seconds 460 msec
Ended Job = job_1603952366868_0041
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1603952366868_0042, Tracking URL = http://project01:8088/proxy/application_1603952366868_0042/
Kill Command = /home/jaisekarh/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0042
Hadoop job information for Stage-0: number of mappers: 1; number of reducers: 1
2020-10-30 07:50:53,241 Stage-2 map = 0%, reduce = 0%
2020-10-30 07:50:59,635 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 2.31 sec
2020-10-30 07:51:06,016 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 4.66 sec
MapReduce Total cumulative CPU time: 4 seconds 660 msec
Ended Job = job_1603952366868_0042
MapReduce Jobs Launched:
Stage-Stage-0: Map: 2 Reduce: 2 Cumulative CPU: 32.46 sec HDFS Read: 457373148 HDFS Write: 753692 SUCCESS
Stage-Stage-0: Map: 1 Reduce: 1 Cumulative CPU: 4.66 sec HDFS Read: 759318 HDFS Write: 403 SUCCESS
Total MapReduce CPU Time Spent: 37 seconds 120 msec
OK
The Right Stock Alerts 2055
The Tribe of Benjamin 1786
TIN-Tech Bloggers 785
AppleRetweetBot 731
Sam rothstein 579

```

## • Cassandra Analysis:

Cassandra is started in command prompt administrator mode.

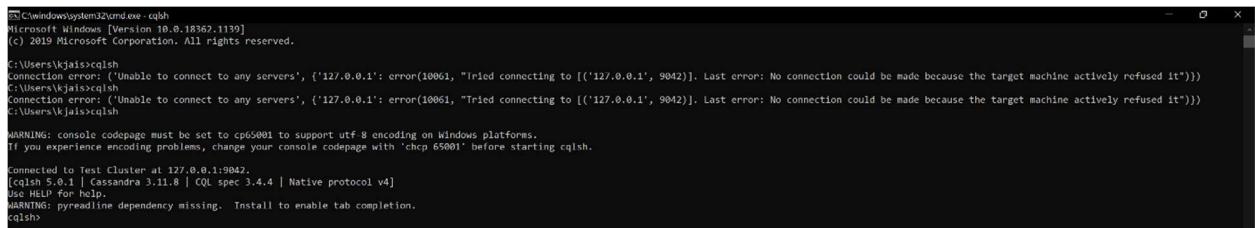


```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.18362.1139]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cassandra
Detected powershell execution permissions. Running with enhanced startup scripts.
*-----*
*-----*
WARNING! Automatic page file configuration detected.
It is recommended that you disable swap when running Cassandra
for performance and stability reasons.
*-----*
*-----*
WARNING! Detected a power profile other than High Performance.
Performance of this node will suffer.
Modifiy config/cassandra-env.ps1 to suppress this warning.
*-----*
*-----*

C:\Windows\System32>CompilerOracle: dominline org/apache/cassandra/db/Columns$Serializer.deserializeLargeSubset (Long/apache/cassandra/io/util/DataInputPlus;Long/apache/cassandra/db/Columns;I)Long/apache/cassandra/db/Column.h
CompilerOracle: dominline org/apache/cassandra/db/Collection$Serializer.serializeLargeSubset ([Ljava/util/Collection;ILorg/apache/cassandra/db/Columns;ILorg/apache/cassandra/io/util/DataOutputPlus;)V
CompilerOracle: dominline org/apache/cassandra/db/Columns$Serializer.serializeLargeSubsetSize (I[Ljava/util/Collection;ILorg/apache/cassandra/db/Columns;I)I
CompilerOracle: dominline org/apache/cassandra/db/commitlog/AbstractCommitLogSegmentManager.advanceAllocatingFrom (Long/apache/cassandra/db/commitlog/CommitLogSegment;)V
CompilerOracle: dominline org/apache/cassandra/db/transform/BaseIterator.tryGetMoreContents ()Z
CompilerOracle: dominline org/apache/cassandra/db/transform/StoppingTransformation.stop ()V
CompilerOracle: dominline org/apache/cassandra/db/transform/StoppingTransformation.stopPartition ()V
CompilerOracle: dominline org/apache/cassandra/db/Util$BufferedDataOutputStreamPlus.flush (I)V
CompilerOracle: dominline org/apache/cassandra/io/util/BufferedDataOutputStreamPlus.writeExcessSlow ()V
CompilerOracle: dominline org/apache/cassandra/io/util/BufferedDataOutputStreamPlus.writeSlow (J)V
CompilerOracle: dominline org/apache/cassandra/io/util/RebufferingInputStream.readPrimitiveSlowly (?)V
CompilerOracle: inline org/apache/cassandra/db/Rows$UnfilteredSerializer.serializeRowBody (Long/apache/cassandra/db/Rows;Row;ILorg/apache/cassandra/db/SerializationLeader;Long/apache/cassandra/io/util/DataOutputPlus;)V
CompilerOracle: inline org/apache/cassandra/db/Rows$UnfilteredSerializer.serializeRowHeader (Long/apache/cassandra/db/Rows;Row;ILorg/apache/cassandra/db/SerializationLeader;Long/apache/cassandra/io/util/DataOutputPlus;)V
CompilerOracle: inline org/apache/cassandra/io/util/SafeMemory.checkBounds (J)V
CompilerOracle: inline org/apache/cassandra/utils/AsymmetricOrdering.selectBoundary (Long/apache/cassandra/utils/AsymmetricOrdering/O;II)
CompilerOracle: inline org/apache/cassandra/utils/AsymmetricOrdering.strictnessOfLessThan (ILorg/apache/cassandra/utils/AsymmetricOrdering/O;)P
CompilerOracle: inline org/apache/cassandra/utils/BloomFilter.indexes (Long/apache/cassandra/utils/IFilter/FilterKey;)[I
CompilerOracle: inline org/apache/cassandra/utils/BloomFilter.setIndex (III)V
CompilerOracle: inline org/apache/cassandra/utils/ByteBufferUtil.compare (ILjava/nio/ByteBuffer;ILjava/nio/ByteBuffer;B)I
CompilerOracle: inline org/apache/cassandra/utils/ByteBufferUtil.compareUnsigned (ILjava/nio/ByteBuffer;ILjava/nio/ByteBuffer;)T
CompilerOracle: inline org/apache/cassandra/utils/FastByteOperations.compareTo (Ljava/lang/Object;ILjava/lang/Object;JI)I
CompilerOracle: inline org/apache/cassandra/utils/FastByteOperations$UnsafeOperations.compareTo (Ljava/lang/Object;ILjava/nio/ByteBuffer;)I
CompilerOracle: inline org/apache/cassandra/utils/FastByteOperations$UnsafeOperations.compare (Ljava/lang/Object;ILjava/nio/ByteBuffer;)I
CompilerOracle: inline org/apache/cassandra/utils/FastByteOperations$UnsafeOperations.compare (Ljava/lang/Object;ILjava/nio/ByteBuffer;)I
CompilerOracle: inline org/apache/cassandra/utils/FastByteOperations$UnsafeOperations.compare (Ljava/lang/Object;ILjava/nio/ByteBuffer;)I
INFO [main] 2020-10-30 13:31:15,072 YamlConfigurationCoder.java:536 Configuration location: file:/C:/apache-cassandra-3.11.8/con/cassandra.yaml
INFO [main] 2020-10-30 13:31:16,189 Config.java:536 - Node configuration: allocate_tokens_for_keyspace=null; authenticator=AllowAllAuthenticator; authorizer=AllowAllAuthorizer; auto_bootstrap=true; auto_snapshot=true; back_pressure_enabled=false; back_pressure_strategy=org.apache.cassandra.net.RateBasedBackPressure[rhigh_ratio=0.9, factor=5, flow=FAST]; batch_size_warn_threshold_in_kb=50; batch_size_warn_threshold_in_kb=5; batch_size_warn_threshold_in_kb=5; batch_size_warn_threshold_in_kb=5; batch_size_warn_threshold_in_kb=5; batch_size_warn_threshold_in_kb=5; broadcast_address=null; broadcast_address=null; buffer_pool_use_heap_if_exhausted=true; cas_contention_timeout_in_ms=1000; cdc_enabled=false; cdc_free_space_check_interval_ms=250; cdc_raw_directory=null; cdc_total_space_in_mb=0; check_for_duplicate_rows_during_compaction=true; check_for_duplicate_rows_during_compaction=true; client_encryption_options=<REMOVED>; cluster_name=lost Cluster; column_index_cache_size_in_kb=2; column_index_size_in_kb=64; commit_failure_policy=top; committing_log_compression=null; committing_directory=null; committing_max_compression_buffers_in_pool=5; committing_perodic_queue_size_in_mb=12; committing_segment_size_in_mb=12; commit
```

Started cqlsh.



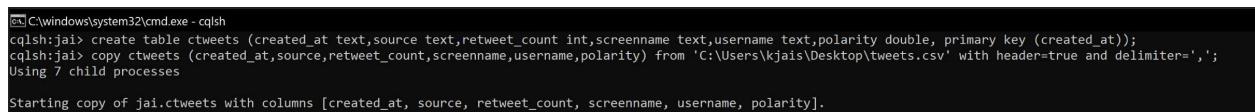
```
C:\Windows\system32>cmd.exe - cqlsh
Microsoft Windows [Version 10.0.18362.1139]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\kjais>cqlsh
Connection error: ('unable to connect to any servers', {'127.0.0.1': error(10061, "tried connecting to [('127.0.0.1', 9042)]. Last error: No connection could be made because the target machine actively refused it")})
C:\Users\kjais>cqlsh
Connection error: ('unable to connect to any servers', {'127.0.0.1': error(10061, "tried connecting to [('127.0.0.1', 9042)]. Last error: No connection could be made because the target machine actively refused it")})

WARNING: console codepage must be set to cp65001 to support utf-8 encoding on Windows platforms.
If you experience encoding problems, change your console codepage with 'chcp 65001' before starting cqlsh.

Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.8 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
WARNING: pyreadline dependency missing. Install to enable tab completion.
cqlsh>
```

Created table in Cassandra with required fields and copied the data from the tweets.csv into the ctweets table.



```
C:\Windows\system32>cmd.exe - cqlsh
cqlsh:1> create table ctweets (created_at text,source text,retweet_count int,screenname text,username text,polarity double, primary key (created_at));
cqlsh:1> copy ctweets (created_at,source,retweet_count,screenname,username,polarity) from 'C:\Users\kjais\Desktop\tweets.csv' with header=true and delimiter=';';
Using 7 child processes

Starting copy of jai.ctweets with columns [created_at, source, retweet_count, screenname, username, polarity].
```

- The below query gives the count of number of most positive tweets which means the polarity values is more than 3.

```
cqlsh:jai> select count(*) as positive from ctweets where polarity > 3 allow filtering;
-
positive
-----
414
(1 rows)

Warnings :
Aggregation query used without partition key
```

- The below query gives the count of number of positive tweets whose polarity value is more than 0.

```
cqlsh:jai> select count(*) as positive from ctweets where polarity > 0 allow filtering;
-
positive
-----
12601
(1 rows)

Warnings :
Aggregation query used without partition key
```

- The below query gives the count of total negative tweets whose polarity value is less than 0.

```
cqlsh:jai> select count(*) as negative from ctweets where polarity < 0 allow filtering;
-
negative
-----
4896
(1 rows)

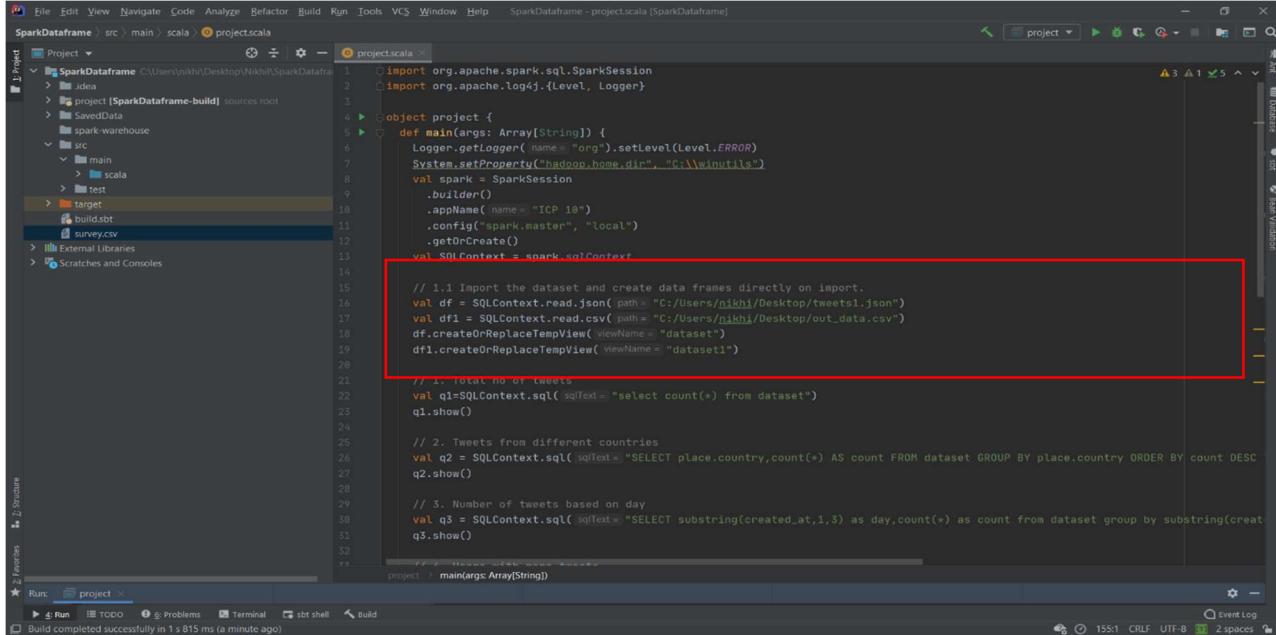
Warnings :
Aggregation query used without partition key
```

## ● SPARK ANALYSIS

Below are the queries for analysis using spark.

### Loading the data

We created two data frames reading data from tweets1.json file which is the twitter data that is extracted and out\_data.csv file is the data with polarity values.

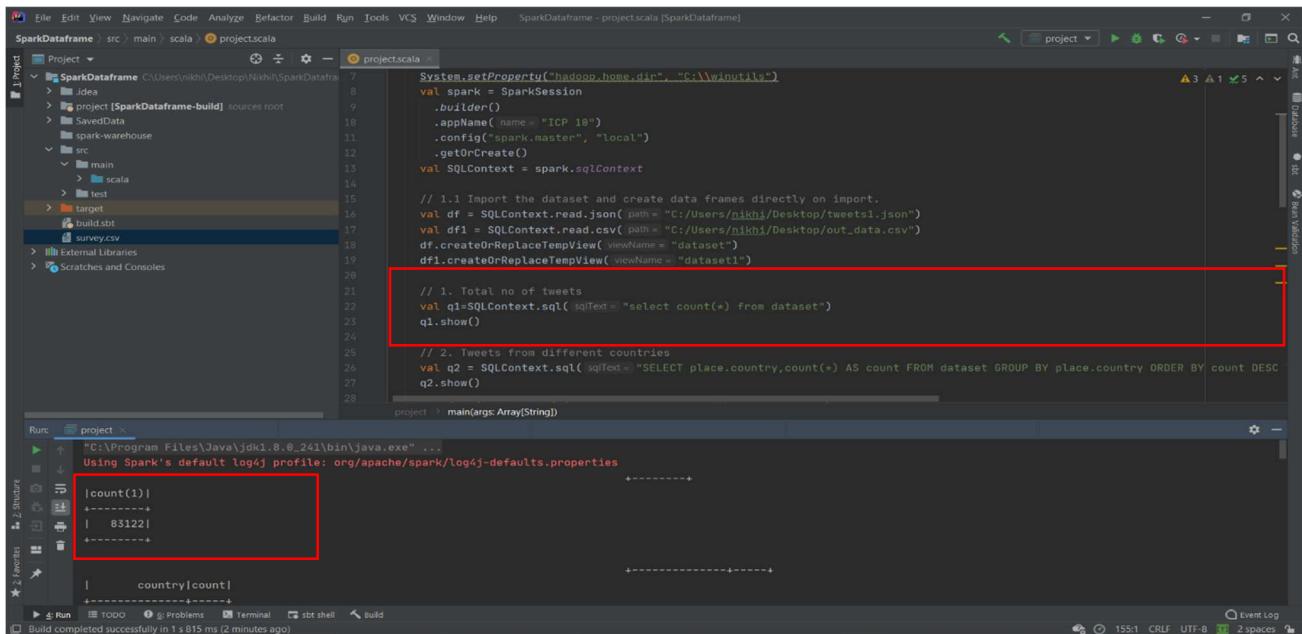


The screenshot shows the IntelliJ IDEA interface with the project 'SparkDataframe' open. The code editor displays a Scala script named 'project.scala'. The code imports org.apache.spark.sql.SparkSession and org.apache.log4j.{Level, Logger}. It defines a main function that sets the log level to ERROR, sets the hadoop.home.dir to C:\winutils, creates a SparkSession, and reads data from tweets1.json and out\_data.csv into datasets df and df1 respectively. The code is highlighted with a red rectangle around the data loading section.

```
1  import org.apache.spark.sql.SparkSession
2  import org.apache.log4j.{Level, Logger}
3
4  object project {
5    def main(args: Array[String]): Unit = {
6      Logger.getLogger("org").setLevel(Level.ERROR)
7      System.setProperty("hadoop.home.dir", "C:\\winutils")
8      val spark = SparkSession
9        .builder()
10       .appName("ICP 10")
11       .config("spark.master", "local")
12       .getOrCreate()
13       val SQLContext = spark.sqlContext
14
15      // 1. Import the dataset and create data frames directly on import.
16      val df = SQLContext.read.json(path = "C:/Users/nikhil/Desktop/tweets1.json")
17      val df1 = SQLContext.read.csv(path = "C:/Users/nikhil/Desktop/out_data.csv")
18      df.createOrReplaceTempView(viewName = "dataset")
19      df1.createOrReplaceTempView(viewName = "dataset1")
20
21      // 2. Total no of tweets
22      val q1 = SQLContext.sql(sqlText = "select count(*) from dataset")
23      q1.show()
24
25      // 3. Tweets from different countries
26      val q2 = SQLContext.sql(sqlText = "SELECT place.country, count(*) AS count FROM dataset GROUP BY place.country ORDER BY count DESC")
27      q2.show()
28
29      // 4. Number of tweets based on day
30      val q3 = SQLContext.sql(sqlText = "SELECT substring(created_at,1,3) as day, count(*) as count from dataset group by substring(created_at,1,3)")
31      q3.show()
32
33    }
34  }
```

1. The below simple query shows number of records in the data frame dataset.

```
val q1=SQLContext.sql("select count(*) from dataset")
q1.show()
```



The screenshot shows the IntelliJ IDEA interface with the project 'SparkDataframe' open. The code editor displays the same Scala script 'project.scala'. The code is highlighted with a red rectangle around the data loading section. Below the code editor, the terminal window shows the output of the command 'q1.show()'. The output shows a single row with the value 831221, indicating the total number of records in the dataset.

```
1  val q1=SQLContext.sql("select count(*) from dataset")
2  q1.show()
3
4
5 +-----+
6 |count(1)|
7 +-----+
8 | 831221|
9 +-----+
```

## 2. The query displays the tweets count from different countries

```
val q2 = SQLContext.sql("SELECT place.country,count(*) AS count FROM dataset GROUP BY place.country ORDER BY count DESC limit 10")
q2.show()
```

The screenshot shows the IntelliJ IDEA interface with the code editor open. A red box highlights the section of code that performs the query. Below the code editor, the run results window shows the output of the query, which lists countries and their tweet counts.

country count
null 82745
United States  303
Canada  9
Australia  7
Argentina  6
Japan  6
Mexico  6
United Kingdom  6
Indonesia  5
India  4

## 3. The query displays the count of tweets based on a day

```
val q3 = SQLContext.sql("SELECT substring(created_at,1,3) as day,count(*) as count from dataset group by substring(created_at,1,3)")
q3.show()
```

The screenshot shows the IntelliJ IDEA interface with the code editor open. A red box highlights the section of code that performs the query. Below the code editor, the run results window shows the output of the query, which lists days of the week and their tweet counts.

day count
Sun  6791
Mon  18171
Thu  9581
Sat  6463
Wed  8656
Tue  21789
Fri  11751

#### 4. The query displays the user with more number of tweets

```
val q4=SQLContext.sql("SELECT count(*) as count, user.name from dataset where user.name is not null group by user.name order by count desc limit 10")
q4.show()
```

count	name
2855	The Right Stock A...
1706	The Tribe of Benj...
785	TIN-Tech Bloggers
731	AppleRetweetBot
579	Sam rothstein
564	Joe Gambiste
558	Music News 360
548	PsychoTrader
477	TickWatcher
467	Towelite

#### 5. The query displays the users with more follower count

```
val q5 = SQLContext.sql("SELECT user.name, max(user.followers_count) as followers_count FROM dataset WHERE text like '%AAPL%' group by user.name order by followers_count desc limit 15");
q5.show()
```

name	followers_count
Reuters	22528183
MarketWatch	3810849
CNBC	3883863
Reuters Business	2179179
Jim Cramer	1423163
Yahoo Finance	986524
zerohedge	817684
TheStreet	736448
Stocktwits	723657
9to5Mac.com	626829
AppleInsider	519664

6. The query displays the tweet count based on months as we extracted the tweets only from October month only that month tweet count is shown.

```
val q6 = SQLContext.sql("SELECT substring(created_at,5,3) as month, count(*) as count from dataset group by substring(created_at,5,3)");
q6.show()
```

```
// 6. Number of tweets based on months
val q6 = SQLContext.sql("SELECT substring(created_at,5,3) as month, count(*) as count from dataset group by substring(created_at,5,3)");
q6.show()
```

month count
Oct 83122

7. The query displays the tweets from different locations in USA

```
val q7=SQLContext.sql("SELECT user.location,count(*) as count FROM dataset WHERE place.country='United States' AND user.location is not null GROUP BY user.location ORDER BY count DESC LIMIT 15");
q7.show()
```

```
// 7. Number of tweets based on different locations in USA
val q7=SQLContext.sql("SELECT user.location,count(*) as count FROM dataset WHERE place.country='United States' AND user.location is not null GROUP BY user.location ORDER BY count DESC LIMIT 15");
q7.show()
```

location count
Los Angeles, CA  28
MA   NC   22
New York, NY  16
Richmond, VA  18
Los Angeles  8
Houston, TX  7
Gaithersburg, MD  6
New York, USA  6
Louisiana  5
Chicago, IL  5

## 8. The query displays the polarity value of top 10 tweets

```
val q8=SQLContext.sql("select _c5 as polarity from dataset1 limit 10")
q8.show()
```

The screenshot shows the IntelliJ IDEA interface with a Scala project named "SparkDataframe". The code editor displays a file named "project.scala" containing several SQL queries. A red box highlights the code for finding the top 10 tweets by polarity:

```
// 8. Polarity of tweets
val q8=SQLContext.sql(sqlText= "select _c5 as polarity from dataset1 limit 10")
q8.show()
```

The resulting output is shown in a separate window, also highlighted with a red box:

polarity
0.5
0.0
0.0
0.0
0.0
0.0
-2.0
0.0
0.0

## 9. The query displays the tweets with users screen name having positive polarity

```
val q9=SQLContext.sql("select _c3 as screenname,_c5 as polarity from dataset1 where _c5>0.0")
q9.show()
```

The screenshot shows the IntelliJ IDEA interface with a Scala project named "SparkDataframe". The code editor displays a file named "project.scala" containing several SQL queries. A red box highlights the code for finding tweets with users having positive polarity:

```
// 9. User Screen name with positive polarity
val q9=SQLContext.sql(sqlText= "select _c3 as screenname,_c5 as polarity from dataset1 where _c5>0.0")
q9.show()
```

The resulting output is shown in a separate window, also highlighted with a red box:

screenname polarity
house_money_ 0.5
Official_Forex 2.0
aaforsel 1.5
ShortingIsFun 2.0
FaisamTrader 2.0
Luke0onay 2.0
manupaga 2.0
AssetReset 3.0
soloalami 2.0
Dividend_Dollar 2.0
BeltwayBreg 2.0
OptionAlarm 1.0

10. The query displays the data with name and number of tweets they tweeted

```
val q10=SQLContext.sql("select user.name, count(*) as count from dataset group by user.name order by count desc limit 10")
q10.show()
```

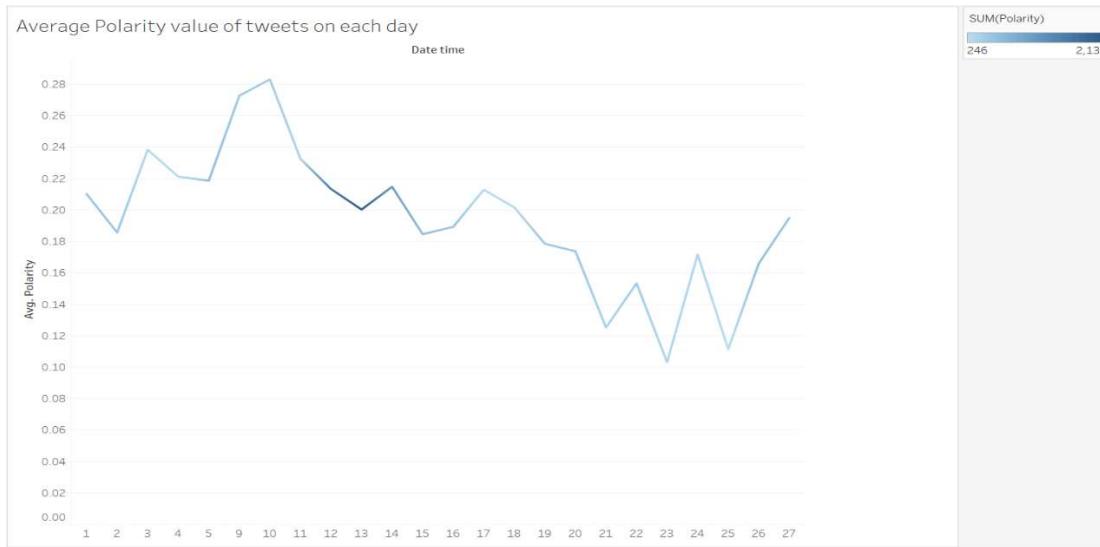
The screenshot shows an IDE interface with a Scala file open. The code implements several SQL queries on a dataset. A red box highlights the code for question 10, which performs a group-by operation on the user names and counts the number of tweets, ordering the results by count in descending order and limiting the output to 10 rows. Another red box highlights the resulting output table, which lists 10 user names along with their tweet counts.

name	count
The Right Stock A...	2855
The Tribe of Benj...	1786
TIN-Tech Bloggers	785
AppleRetweetBot	731
Sam rothstein	579
Joe Gambiste	564
Music News 368	558
PsychoTrader	548
TickWatcher	477
Towelie	467

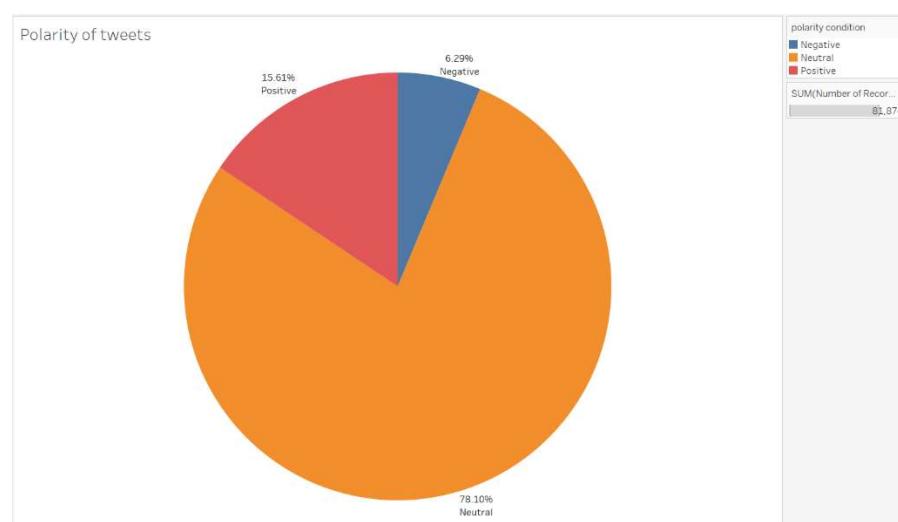
## Visualization & Result Evaluation:

We used Tableau for the visualization

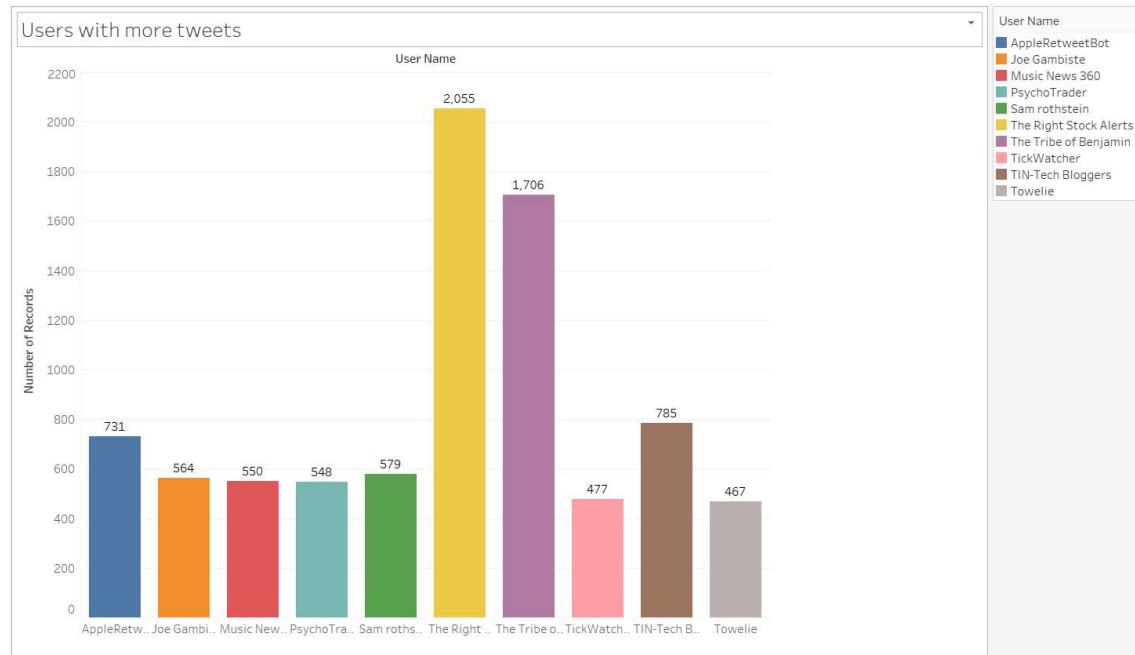
- The Average polarity of tweets based on keyword **AAPL** on each day of October month. From October 5<sup>th</sup> to 10<sup>th</sup> there is a gradual increase in the polarity value. The highest average polarity value is recorded on October 10<sup>th</sup>. From 10<sup>th</sup> of the month polarity value gradually decreases.



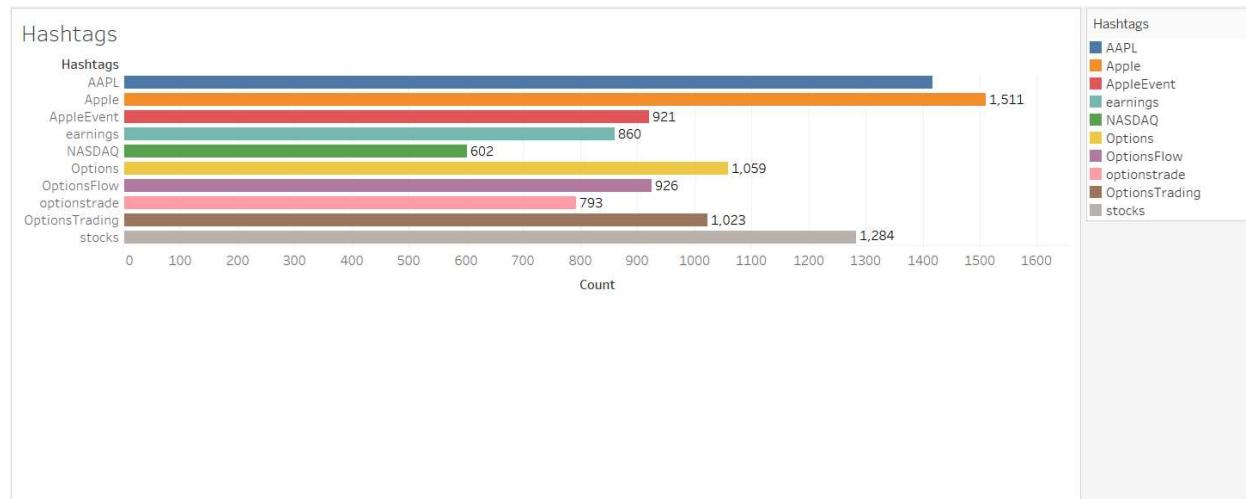
- Polarity value of each tweet is calculated by using sentiment analysis on each tweet. We get values in the range between -5 to +5. If the polarity value lies between -5 and 0 it is Negative. If the polarity value lies between 0 and 5 it is Positive. If the polarity value is 0 and 1 it is Neutral. The data shows 78% tweets are neutral and remaining 22% of tweets are positive and negative.



- Data shows the top 10 users with a greater number of tweets in the month of October. User with username **The Right Stock Alerts** has more tweets on AAPL stock in the month of October.



- Data shows that Apple is the hashtag that has highest wordcount in the month of October

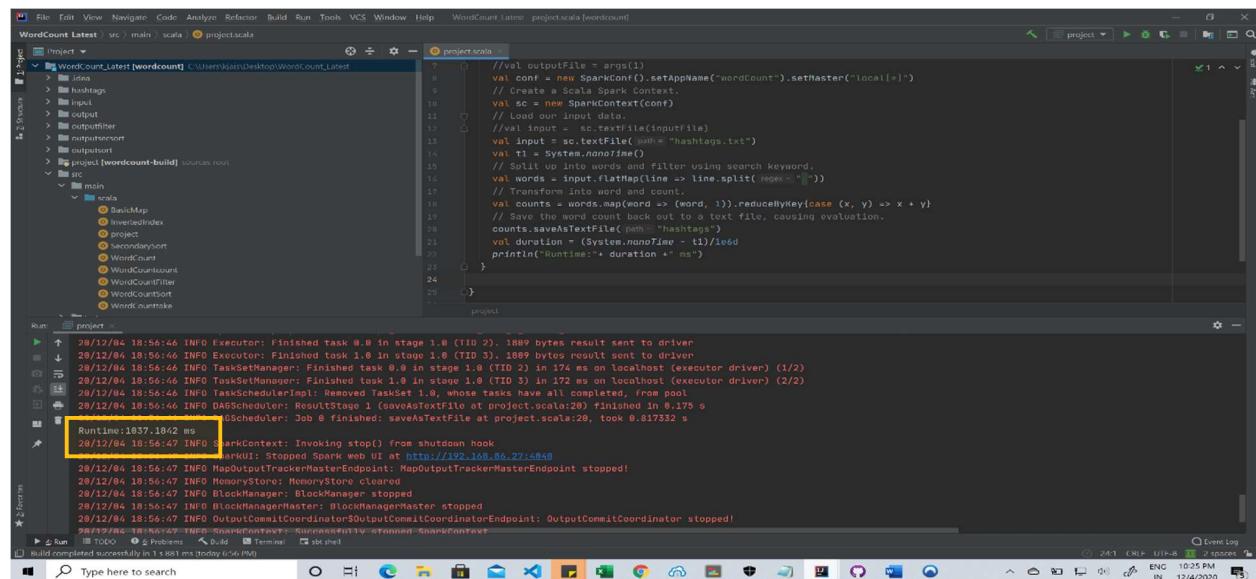


# Technical Analysis

## MapReduce vs Apache Spark

In the below we ran the wordcount program in both apache spark and mapreduce. The runtime for the wordcount in apache spark is 1037 ms and runtime in Hadoop mapreduce is 2790 ms (It can be more). By this we can say Apache Spark is way faster than Mapreduce in means of processing the data.

Apache Spark:

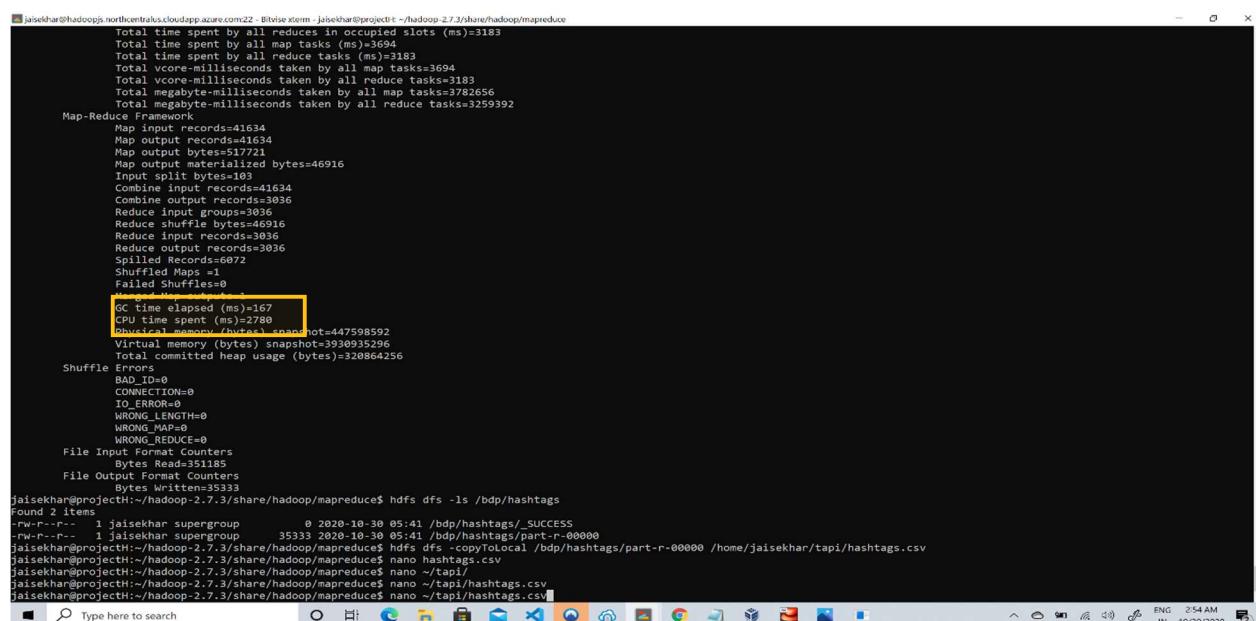


```
project.scala
7 //val outputfile = args(1)
8 val conf = new SparkConf().setAppName("wordCount").setMaster("local[*]")
9 // Create a Scala Spark Context.
10 val sc = new SparkContext(conf)
11 // Load input file into rdd
12 //val input = sc.textFile(path + "hashtags.txt")
13 val input = sc.textFile("hashtags.txt")
14 val t1 = System.nanoTime()
15 // Split up into words and filter using search keyword.
16 val words = input.flatMap(line => line.split(" "))
17 // Transform into word and count.
18 val counts = words.map(word => (word, 1)).reduceByKey((x, y) => x + y)
19 // Save the word count back out to a text file, causing evaluation.
20 counts.saveAsTextFile("hashtags")
21 val duration = (System.nanoTime() - t1)/1e9d
22 println("Runtime: " + duration + " ns")
23 }
```

Run

```
29/12/04 18:56:26 INFO Executor: Finished task 0.0 in stage 1.0 ((TID 2), 1889 bytes result sent to driver
29/12/04 18:56:46 INFO Executor: Finished task 1.0 in stage 1.0 ((TID 3), 1889 bytes result sent to driver
29/12/04 18:56:46 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 2) in 174 ms on localhost (executor driver) (1/2)
29/12/04 18:56:46 INFO TaskSetManager: Finished task 1.0 in stage 1.0 ((TID 3) in 172 ms on localhost (executor driver) (2/2)
29/12/04 18:56:46 INFO TaskSchedulerImpl: Removed TasksSet 1.0, whose tasks have all completed, from pool
29/12/04 18:56:47 INFO DAGScheduler: ResultStage 1 (saveAsTextFile at project.scala:20) finished in 0.179 s
29/12/04 18:56:47 INFO DAGScheduler: DScheduler: Job 0 finished: saveAsTextFile at project.scala:20, took 0.817332 s
Runtime:1037.1842 ms
29/12/04 18:56:47 INFO DAGScheduler: Invoking stop() from shutdown hook
29/12/04 18:56:47 INFO DAGScheduler: Stopped Spark web UI at http://192.168.86.27:4040
29/12/04 18:56:47 INFO OutputCommitCoordinator: OutputTrackerMasterEndpoint stopped!
29/12/04 18:56:47 INFO MemoryStore: MemoryStore cleared
29/12/04 18:56:47 INFO BlockManager: BlockManager stopped
29/12/04 18:56:47 INFO BlockManagerMaster: BlockManagerMaster stopped
29/12/04 18:56:47 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
```

Hadoop Mapreduce:

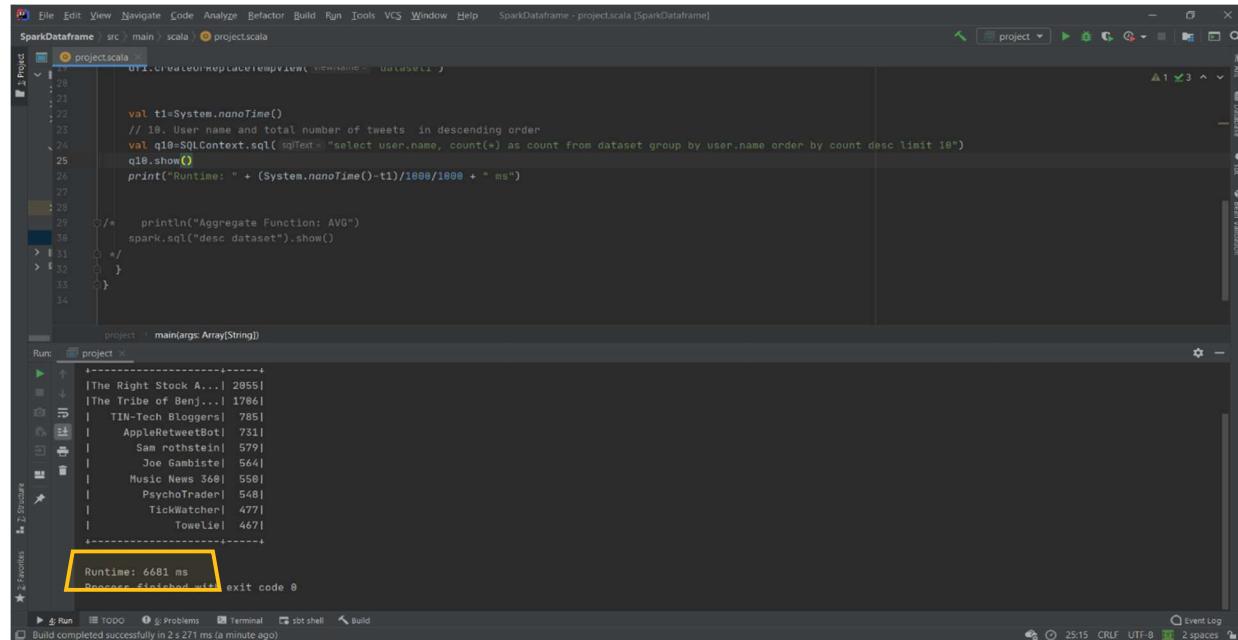


```
jaisekhar@hadoop:~/hadoop-2.7.3/share/hadoop/mapreduce$ hdfs dfs -ls /bdp/hashtags
Total time spent by all reduces in occupied slots (ms)=3183
Total time spent by all map tasks (ms)=3694
Total time spent by all reduce tasks (ms)=3183
Total vcore-milliseconds taken by all map tasks=3694
Total vcore-milliseconds taken by all reduce tasks=3183
Total megabyte-milliseconds taken by all map tasks=3782656
Total megabyte-milliseconds taken by all reduce tasks=3259392
Map-Reduce Framework
  Map input records=41634
  Map output records=41634
  Map output bytes=517721
  Map output materialized bytes=46916
  Input split bytes=103
  Combiner input records=41634
  Combine output records=3036
  Reduce input groups=3036
  Reduce shuffle bytes=46916
  Reduce input records=3036
  Reduce output records=3036
  Spilled Records=6072
  Shuffled Maps=3036
  Failed Shuffles=0
  Merged Map outputs=3036
  GC time elapsed (ms)=167
  CPU time spent (ms)=2780
  Virtual memory (bytes) snapshot=447598592
  Virtual memory (bytes) total=3930935296
  Total committed heap usage (bytes)=320864256
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_PARTITION=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=351185
File Output Format Counters
  Bytes Written=35333
jaisekhar@hadoop:~/hadoop-2.7.3/share/hadoop/mapreduce$ hdfs dfs -ls /bdp/hashtags
Found 2 items
-rw-r--r-- 1 jaisekhar supergroup          0 2020-10-30 05:41 /bdp/hashtags/_SUCCESS
-rw-r--r-- 1 jaisekhar supergroup  5333 2020-10-30 05:41 /bdp/hashtags/part-r-00000
jaisekhan@projectH:/~hadoop-2.7.3/share/hadoop/mapreduce$ hdfs dfs -copyToLocal /bdp/hashtags/part-r-00000 /home/jaisekhan/tapi/hashtags.csv
jaisekhan@projectH:/~hadoop-2.7.3/share/hadoop/mapreduce$ nano hashtags.csv
jaisekhan@projectH:/~hadoop-2.7.3/share/hadoop/mapreduce$ nano -v tapi/hashtags.csv
jaisekhan@projectH:/~hadoop-2.7.3/share/hadoop/mapreduce$ nano -v tapi/hashtags.csv
```

## Hive vs Apache Spark

In the below snippet the comparison was drawn between processing time of Hive and Apache Spark using QL. Apache spark took only 6681 ms where as Hive took 37000 ms. By this we can say Apache spark is very faster than Hive.

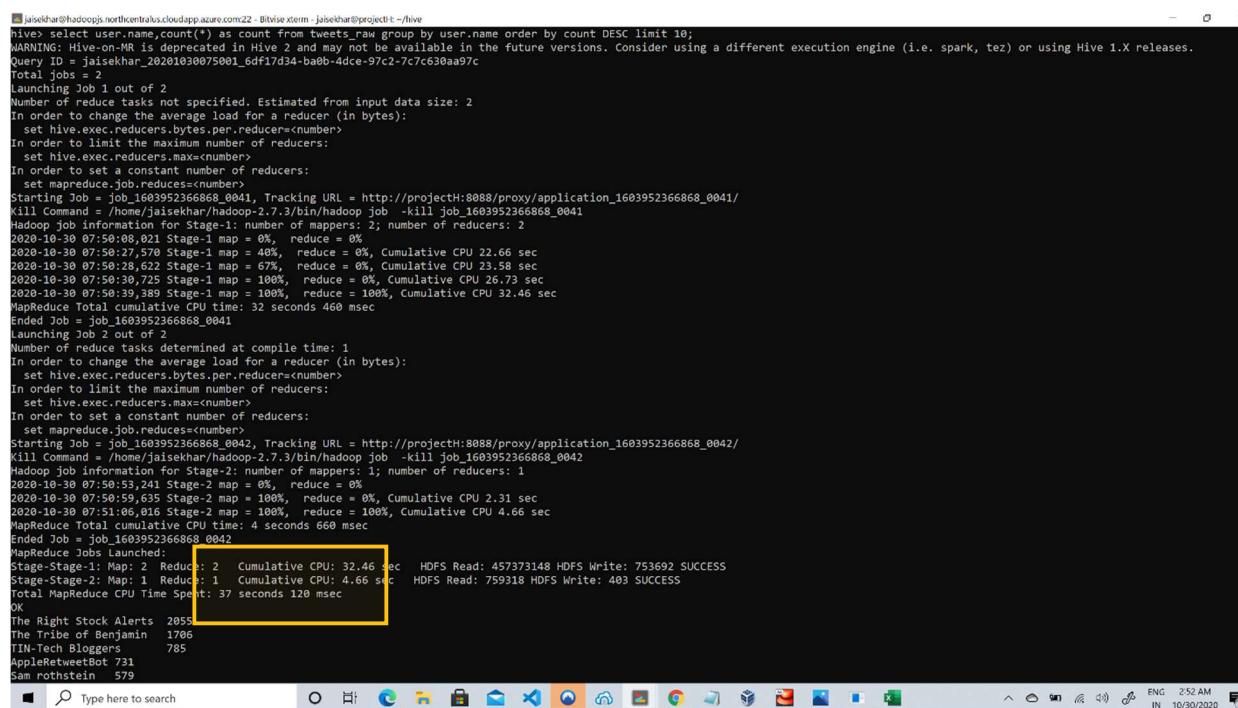
### Apache Spark:



```
val t1=System.nanoTime()
// 10. User name and total number of tweets in descending order
val q10=sqlContext.sql("select user.name, count(*) as count from dataset group by user.name order by count desc limit 10")
q10.show()
print("Runtime: " + (System.nanoTime()-t1)/1000/1000 + " ms")
```

Runtime: 6681 ms  
Process finished with exit code 0

### Hive:

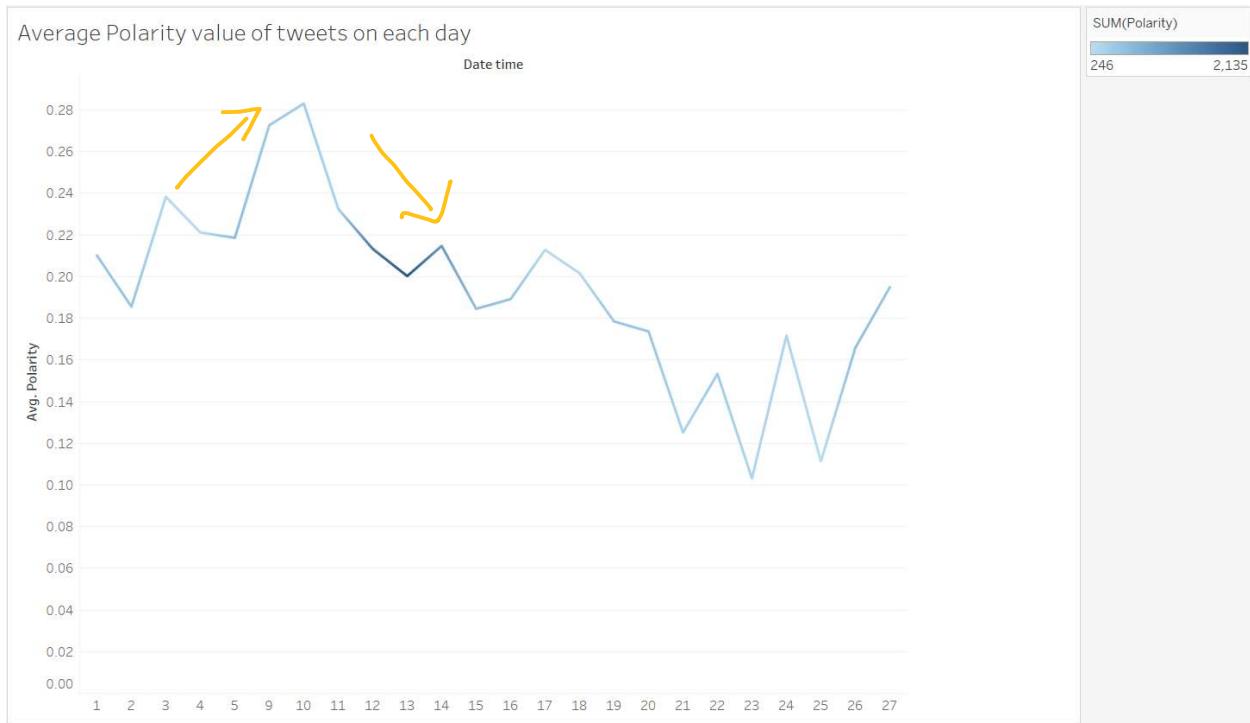


```
hive> select user.name,count(*) as count from tweets_raw group by user.name order by count DESC limit 10;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = jaisekhar_20201030075001_6df17d34-ba0b-4dc6-97c2-7c7c630aa97c
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 2
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1603952366868_0041, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0041/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0041
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 2
2020-10-30 07:50:05,021 Stage-1 map = 0%, reduce = 0%
2020-10-30 07:50:27,570 Stage-1 map = 40%, reduce = 0%, Cumulative CPU 22.66 sec
2020-10-30 07:50:28,627 Stage-1 map = 40%, reduce = 0%, Cumulative CPU 23.58 sec
2020-10-30 07:50:30,725 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 26.73 sec
2020-10-30 07:50:39,389 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 32.46 sec
MapReduce Total cumulative CPU time: 32 seconds 460 msec
Ended Job = job_1603952366868_0041
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1603952366868_0042, Tracking URL = http://projectH:8088/proxy/application_1603952366868_0042/
Kill Command = /home/jaisekhar/hadoop-2.7.3/bin/hadoop job -kill job_1603952366868_0042
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2020-10-30 07:50:53,241 Stage-2 map = 0%, reduce = 0%
2020-10-30 07:50:59,635 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 2.31 sec
2020-10-30 07:51:06,016 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 4.66 sec
MapReduce Total cumulative CPU time: 4 seconds 660 msec
Ended Job = job_1603952366868_0042
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2 Reduce: 2 Cumulative CPU: 32.46 sec HDFS Read: 457373148 HDFS Write: 753692 SUCCESS
Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 4.66 sec HDFS Read: 759318 HDFS Write: 403 SUCCESS
Total MapReduce CPU Time Spent: 37 seconds 120 msec
OK
The Right Stock Alerts 2055
The Tribe of Benjamin 1786
TIN-Tech Bloggers 785
AppleRetweetBot 731
Sam rothstein 579
```

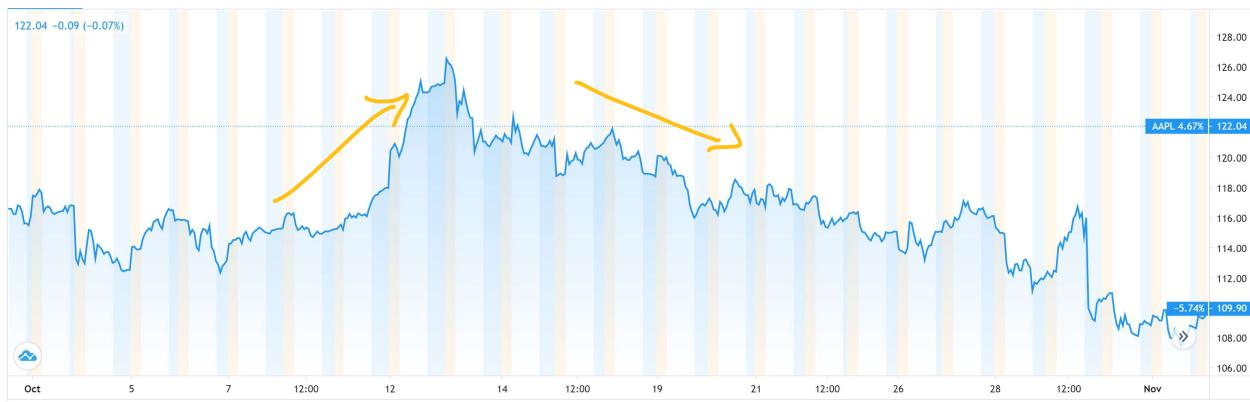
## Stock Analysis: (Important)

The below two graphical representations are the trends obtained from analysis and other is the real stock market trend of the ticker AAPL stock in the month of October. Here, we can clearly see the identical trend between both the real and analytical trend which gives early signs for the investment. By this we can say that stock can be predictable with this system.

### Our System: (Tableau Representation)



## Real Stock: (Trading View)



## Conclusion

Using this system, we successfully analyzed the data extracted from the twitter social opinions based on stock ticker with the help of hadoop ecosystem. In the above results, we able to get the trend after analysis which can be used for predictions of movement of stock. This trend might help for the investors in future investments. These trends are giving early signs for the investments which might help to gain the profits. Right now, this system might be only for one stock and for one month but it can be extended for future usge. Thus, we can say that we successfully achieved our goal using this system.

## Future work

In this system, we analysed only AAPL stock ticker, which we extracted based on this keyword from the twitter for one month. In future, we can extend this range of extraction on keywords on time frames accordingly which might be dynamic in nature. Since our system is based on historical data but we can also try to analyze the live data with proper models.

## Project Management

- **Work completed:**

1. Extracted tweets using python and twitter API for the dataset.
2. Loading the data into the HDFS.
3. Sentiment Analysis done on every tweet in the dataset.
4. Wordcount on hashtags using mapreduce.
5. Creation of table and loading of data into Hive.
6. Analysis of data using Hive
7. Analysis of data using Cassandra
8. Visualization using Tableau.
9. Initialization and loading the data into Apache Spark.
10. Analysis of data using SparkSQL
11. Technical analysis by comparing Apache Spark, Mapreduce and Hive
12. Final comparison between our system trend and real trend of stock (success)

- **Contribution:**

**Jaisekhar Koya : 25%**

Extracting tweets using Python. Done sentimental analysis using AFINN dictionary which gives polarity value. Extracted tweets were stored in HDFS. Created and loaded data to hive. Performed few queries in Hive and also Cassandra. Drawn the comparison between Apache Spark and Mapreduce. Contributed in drawing the comparison between trends of system and real.

### **Sri Sai Nikhil Kantipudi:** 25%

Helped in doing sentimental analysis on every tweet in the dataset. Done wordcount on extracted hashtags using MapReduce. Contributed in loading and analysing the data in Hive. Performed few queries in Hive and also in Cassandra. Analyzed the data using Spark SQL. Also contributed to work for comparing system vs real trends.

### **Sai Rohith Guntupally:** 25%

Tweets extracted and parsed for hashtags using python. Done hdfs operations on the extracted dataset. Performed few queries on the dataset using hive and Cassandra. Part of visualization in tableau. Wrote few queries in apache spark. Drawn the comparison between Apache Spark and Hive.

### **Aarthi Nagireddy:** 25%

Sentiment Analysis using dictionary system. Hive initialization and loading of data of the dataset. Performed few queries in Hive and Cassandra for data analysis. Part of Visualization work in tableau. Initialization and loading the data into Apache Spark. Wrote few queries in apache spark.

## Story Approach

### Chapter – 1: Life

- **Who** are the people or communities in need of help?

Investors who play a lot with stock market will get helped with this system which makes their stock selection and timing effective.

- **What** problem happened to them?

Half of the investors are failing in stock selection with ineffective analyses.

- **When** did the problem take place?

During investing in stock market on particular stocks by investors many people will loose their investment because of inaccurate approaches.

- **Where** means two things: 1) The environment and settings that the people or the community is living in, and 2) the place/location where the problem take place.

Only in stock market this problem exists. There were different kinds of stock trading's. But, when it comes to the market we got only one stock market with different exchanges.

- **Why** means the possible causes and/or origin of the problem.

People always wants to earn or save their money by buying shares in good company. But their selection of company or stock might be bad because of their process of analyses.

- **How:** If you would like, you can add a dimension of how. How did it happen? Sometimes, the answer to how can be covered by what, when and where.

For example, if we take AAPL stock which belongs to apple. X likes the apple products so wants to invest in the company. So X bought the shares of AAPL stock. But, what X don't know is, at the same time Apple facing BendGate issue and stock just decreasing which bought loss to X. This system helps in analysing the user opinions and helps in selection.

## Chapter -2: Data

- **Who** is the data set about? **Who** were sampled in this data set? **Who** were over sampled or under sampled? Are they representative of the main characters in Assignment 1? Is there any identifiable information or is there any risk of disclose identifiable information? This is fundamentally about the sampling issue, and anonymity.

The dataset is about the users who tweeted with the keyword 'AAPL' (in this case) in the twitter. There are many numbers of users who tweets in twitter. Thus, it help us in analyzing more data. This dataset contains only public available information . It doesn't contain any undisclosable information. These users are the main representatives stated in the assignment 1.

- **What** events, activities, behaviors, and observations etc. are recorded by the data set? Does the data set record the targeted events, activities, behaviors, etc. in Assignment 1? This is fundamentally about the variables.

The dataset contains the tweets extracted from the twitter which contains user information, tweet text, hashtags, created time and their entity information. This are the important variables used in the project. This covers all the required variables stated in the assignment 1.

- **When** did the event, activity, behavior, and observation, etc. take place? **When** were the data collected? Is it longitudinal or cross-sectional? Are they real time data? How old or fresh are the data? To what extent generalization can be made across time to inform Assignment 1? This is fundamentally about the temporal structure of the data set, and the external validity of the data set across time.

The data is generated when the user tweets about AAPL stock in the twitter. There will be a greater number of users who tweets same. The data in our dataset consists of all the tweets which contains AAPL keyword in Oct 1<sup>st</sup> to Oct 28<sup>th</sup>. Hence this is the old data not the real time data. This is permanent data which might need to extract more data in future for further analysis of other data range. It satisfied the abstract stated in assignment 1.

- **Where** did the event, activity, behavior, and observation, etc. take place? **Where** were the data collected if the information is available? What does the geographical coverage of the data set look like? Does the data set contain geographical information (GIS)? Is this a local, regional, national, or global data set? To what extent generalization can be made across settings to inform Assignment 1? This is fundamentally about geographic variables in the data set, and the external validity of the data set across settings.

The dataset consists of tweets which are extracted from twitter. Twitter consists of huge amount of data. It is an online social media platform. It has more number of users. This data set contains geographical information of the tweeted location. This dataset doesn't have geographical limitations. This contains the data as stated in Assignment 1.

- **Why** did the event, activity, behavior, or observation etc. take place? **Why** were the data collected?

Twitter is one of the important sources for the data. Many number of people puts their opinion in the twitter. In the same way, many companies or investors puts their news and opinions in twitter which effects in stock trend. Hence we are using their opinions(tweets) to analyze the stock movement.

- **How:** If you would like, you can add a dimension of how. How did it happen? Sometimes, the answer to how can be covered by what, when and where.

Most of this part was already covered above. But to be precise, we are collecting the tweets from twitter and doing sentiment analysis to find whether their opinion is positive or negative and do the analysis on the whole data using hadoop ecosystem.

- **Calibration 1:** Every aspect covered in Assignment 1 aligns with the Assignment 2. We have chosen right dataset in Assignment 2 for analysis as mentioned in the Assignment 1. And, this is the relevant dataset to the project.
- **Calibration 2:** Experimented different models for our system using various NLP techniques. Aligned the data according to the process of using NLP techniques which is sentiment analysis. Applied accurately on the parsed data from twitter dataset.

## Chapter -3: The Scientist and AI

- **Who:** In this story, there are three main characters: 1) the people/the community who needs help, 2) the data scientist (that is you), and 3) AI. How much does the data scientist understand Assignment 1 (domain) and Assignment 2 (data)?

The student/data scientist will understand the domain chapter as it will resemble the complete idea of the project which needs to done can also be considered as requirements technically. Domain knowledge is required to develop the project in later stages. Also, after getting the information related to the domain the data scientist needs to work on the data to process and analyze based on the requirements which stated in the chapter 2 Data.

- **What** models and analysis did the data scientist and AI apply to fulfill the need of the people or the community?
  - Can the data scientist estimate and select data for their goals from Assignment 1? Can they map data sets from Assignment 2 onto appropriate ML models?
  - Can the data scientist connect Story 1 with ML models/stories about what a ML model can do? To perform good ML research, what in-depth knowledge and experience with ML algorithms and ML stories does a data scientist need?

Data scientist selects the data according to the specified goals in the chapter 1. Using the same data, data scientist will gather the data as specified in the goals of chapter 2. Then works on the data using the domain knowledge. In this specific project, the data scientist needs the have knowledge about the hadoop ecosystem.

- **When** has to do with the iterations (Calibration 2). How much time did it take for experimentation? How efficient is the modeling/algorithm? Can the data scientist determine the acceptance level of the model (validation with accuracy and runtime performance) considering the targeted users?

Various technologies were used in this project. For every analysis done using one technology in hadoop ecosystem we will be having different run time performances and accurancies. In this project we dealt only NLP technique which is sentimental analysis and data analysis with different technologies in hadoop ecosystem.

- **Where** has to do with the learning environment. Where did this experiential learning process take place? For example, it was part of an online Deep Learning course.

It was part of the Big Data Programming course offered in University of Missouri-Kansas City and also few online courses for NLP Techniques.

- **How:** If you would like, you can add a dimension of how. How did it happen? Sometimes, the answer to how can be covered by what, when and where.

The data scientist can use data in the way it is required for the analysis or based on the tool. New data or data source can be added for further analysis as per required nothing static in this process.

## Chapter 4: Users

- **Who:** the main character is the targeted user or audience?

The project is targeting the investors who is involved in the stock market. The analysis comes from this project will help them in picking the right stocks.

- **What** can the application do? What does the visualization show?

The analysis shows the one month analysis on the stock ticker's data collected from the twitter. It provides one month movement of stock using the social opinions by using different tools in hadoop ecosystem.

- **When** can the user use the application/visualization?

The investor is user here. The investor uses this analysis or visualization during the investment in stock market.

- **Where** will the visualization and applications be deployed, for example, mobile phones, the web, or IoT devices?

The visualization as of now is deployed in the tableau using the data retrieved after the analysis in the hadoop ecosystem

- **Why** is the visualization or application useful to the user?

This analysis helps the investor to pick the right stock to invest in the stock market and gain profits based on its historical movements and social opinions.

- **How** will the people/the community use this application or visualization to make changes?

The investor uses this analysis or visualization for any stock. They can change the stock ticker according to their requirement in the future.

## Chapter 5: The society

- **Who** will be impacted? **Who** were sampled? **Who** were over sampled or under sampled? **Who** were the data scientists (yourself and your collaborators)?

The investor will get impacted using system. They get right stocks to pick after the results of the analysis. The data scientists here are the student who performs the analysis using the hadoop ecosystem. They just get benefited.

- **What** are the social and cultural impacts? **What** are the concerns about data privacy, security, and fairness?

The investors only get impacted as stated above. We are using the data which is publicly available only. We are not dealing with any impacting privacy or security or fairness of data.

- **When** will the social and cultural impacts take place? **When** should people be concerned about data privacy, security, and fairness?

During the investment in the stock market the investors might use this application which might impact in their investment. We are using the data which is publicly available only. We are not dealing with any impacting privacy or security or fairness of data.

- **Where** will the social and cultural impacts take place? **Where** will data privacy, security, and fairness issues, like data breach, and evaluative bias, likely to happen?

The impacts will take place during the investment in the stock market. It wont breach the data privacy not the security nor the fairness issues. Even, this system is not evaluative bias.

- **Why** are the social and cultural impacts important or consequential to the people and/or the community? **Why** should we be concerned about data privacy, security, and fairness issues?

This is very important in investors community. Because their daily routine is to pick the right stocks and do the investment. This system gives the analysis about the stock which helps them in the picking the right stock. It won't breach the data privacy not the security nor the fairness issues. Even, this system is not evaluative bias.

- How can we address these societal issues in ML using a community-in-the-loop approach?

Right now, this system is using NLP techniques to process the data which is sentimental analysis on the data collected from the twitter. This helps to retrieve the social opinions on large scale with out any biased in the results.

## References

- ANALYSIS AND PREDICTION OF STOCK MARKET USING BIG DATA TECHNOLOGY  
[https://www.academia.edu/36679481/ANALYSIS\\_AND\\_PREDICTION\\_OF\\_STOCK\\_MARKET\\_USING\\_BIG\\_DATA TECHNOLOGY](https://www.academia.edu/36679481/ANALYSIS_AND_PREDICTION_OF_STOCK_MARKET_USING_BIG_DATA TECHNOLOGY)
- Stock Market Prediction on Bigdata Using Machine Learning Algorithm  
<http://ijesc.org/upload/b91a9a994c4d79a72f5542393ca9469d.Stock%20Market%20Prediction%20on%20Bigdata%20Using%20Machine%20Learning%20Algorithm.pdf>

GITHUB Link:

<https://github.com/jaisekhar/Stock-Market-Analysis-using-Hadoop-Ecosystem>