Quantitative Analyses for Customer Satisfaction and Brand Imaging (ReadMe)

The following are the steps that we have followed for this project.

1. Data Scraping from different platforms
2. Code or steps for data ingest
3. Data cleaning for the scraped data and profiling code
4. Data analytics over the cleaned data.

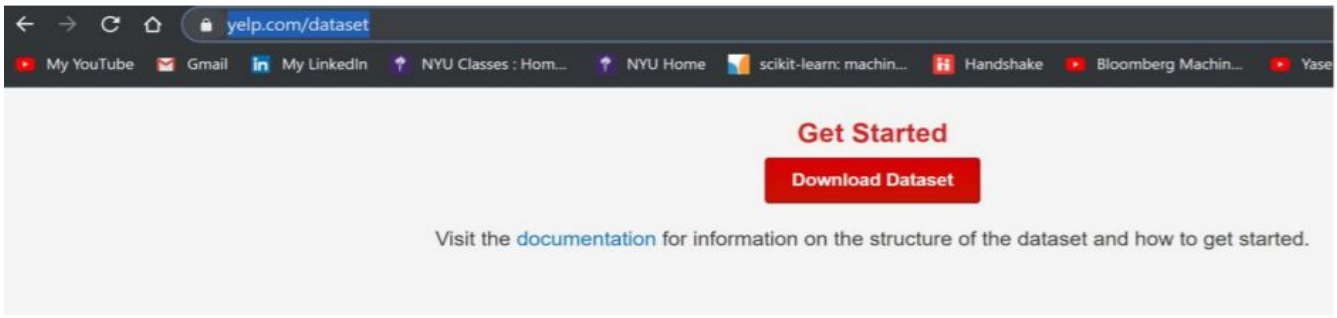
Additionally, the challenges that we faced are also mentioned in each step.

1. Data Scraping

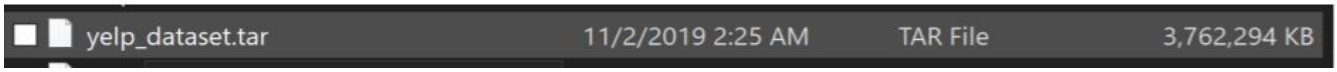
We have scrapped the data from 3 different sources: Yelp, Twitter and Reddit.

YELP :

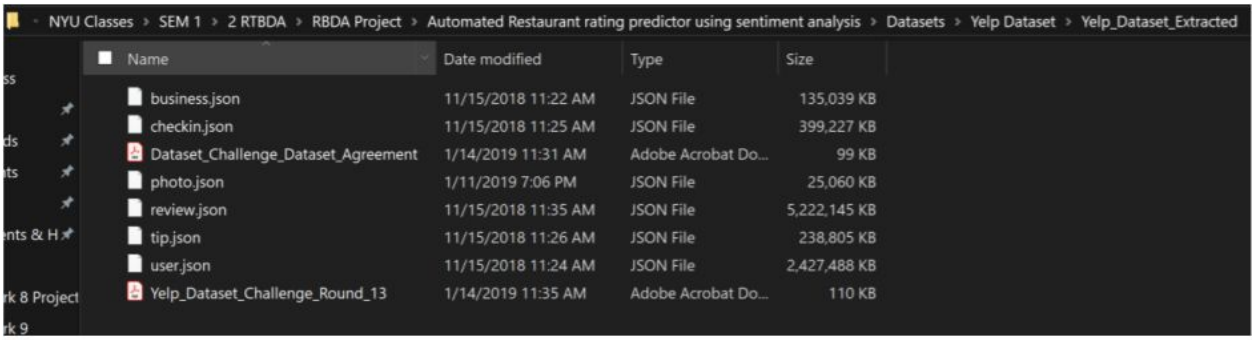
For yelp, we got the data available from there website: → https://www.yelp.com/dataset/download. The total size of the data is **8GB**. Data had to be further cleaned for the analyses.



- The data is available in .tar format



The data files were extracted using the command on command prompt at the folder destination as:
tar -xvf yelp_dataset.tar
Yelp Extracted datafiles in JSON format:



TWITTER:

For this, the creation of a twitter development account is necessary which will provide *consumer_key, consumer_secret, access_key and access_secret,* which is necessary to establish a connection with Twitter. These values are unique for every user.

For data extraction, python script is used which is placed under this location: */user/ss13449/project/twitter_scrap.py*

Below is the screenshot for the code:

```
#!/usr/bin/env python3
#Author: Sonal Sharma
import tweepy
from tweepy import Stream
from tweepy import OAuthHandler
from tweepy.streaming import StreamListener

#consumer API key, consumer API secret key, access token and access token secret
consumer_key = "xx"
consumer_secret = "xx"
access_key = "xx"
access_secret = "xx"

class listener(StreamListener):
    def on_data(self,tweets):
        #creating .csv file to store extracted tweets
        f = open("twitter_starbucks.json" , "a")
        f.write(tweets + "\n")
        f.close()
        return(True)

#Accessing twitter with tweepy library for python with help of tokens and keys

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_key, access_secret)
stream = Stream(auth,listener())
stream.filter(track=["Starbucks"])
#stream.filter(track=["MacDonalds"])
```

Twitter data for McDonalds and Starbucks was extracted using the commands as per the attached python file. (Reference: Twitter Assignment)

Through this script, we were able to scrap the data from twitter for different restaurants(of approx size **1.1 GB**)

REDDIT:

Setting up an account over Reddit is necessary for data scraping and which will provide unique values for the fields like *client_id, client_secret,user_agent, username, and password.* These values are required to successfully establish a connection with Reddit. Python script for data scraping from Reddit is placed under:

/user/ss13449/project/reddit_scrap.py

Below is the screenshot for the code:

```python
#Reddit Data Extraction Code in Python:
import praw
import datetime as dt

reddit = praw.Reddit(client_id='xxxxxxxxxxxxxxxxxx', \
                     client_secret='xxxxxxxxxxxxxxxxxx' \
                     user_agent='xxxxxxxxxxxxxxxxxx', \
                     username='xxxxxxxxxxxxxxxxxx', \
                     password='xxxxxxxxxxxxxxxxxx')

subreddit = reddit.subreddit('SearchWords')
top_subreddit = subreddit.top() ## Similarily we have .top(),.new(),.hot(),.controversial() and .gilded()
top_subreddit = subreddit.top(limit=2000)

for submission in top_subreddit:
    date_new = get_date(submission)
    print("{},\"{}\",{}".format(submission.id,submission.title,date_new))

def get_date(submission):
    time = submission.created
    return datetime.datetime.fromtimestamp(time)

## Ref: http://www.storybench.org/how-to-scrape-reddit-with-python/
```

The extracted two files are:

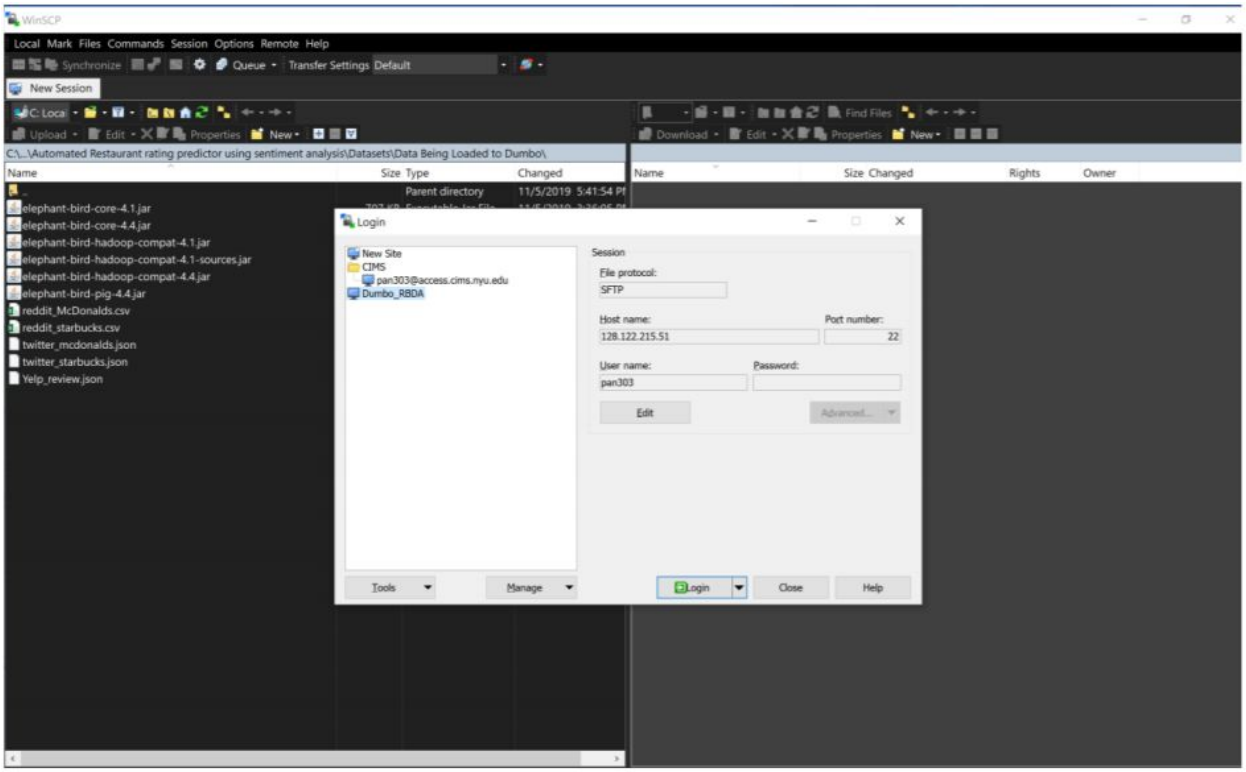
The data size of the data that we extracted from Reddit is around **1GB.**

2. Code or Steps for Data Ingest

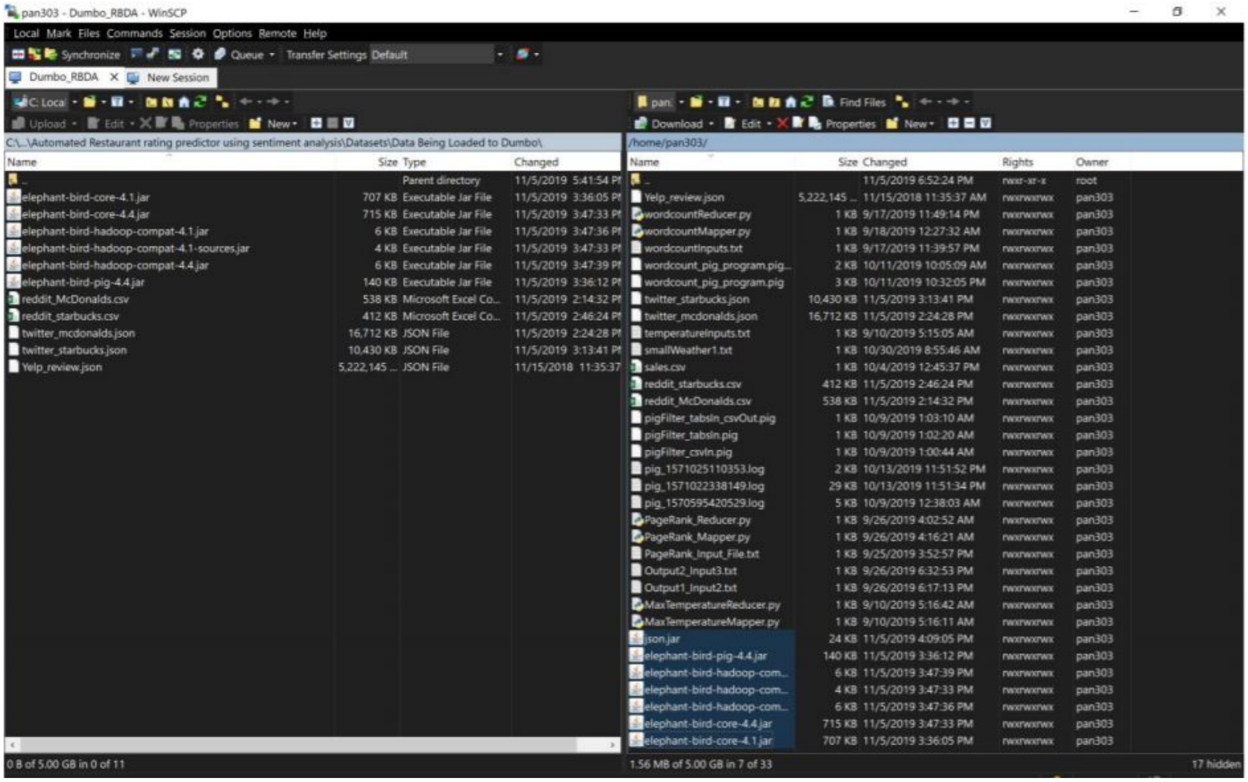
WinSCP (Windows Secure Copy), a popular SFTP & FTP Client was used to copy the files from a local computer to the dumbo cluster.

Note: WinSCP can be used smoothly for file transfer if the NYU network is used. In case of non-NYU network connection, it is advisable to use the NYU VPN (i.e. Cisco AnyConnect Secure Mobility Client)

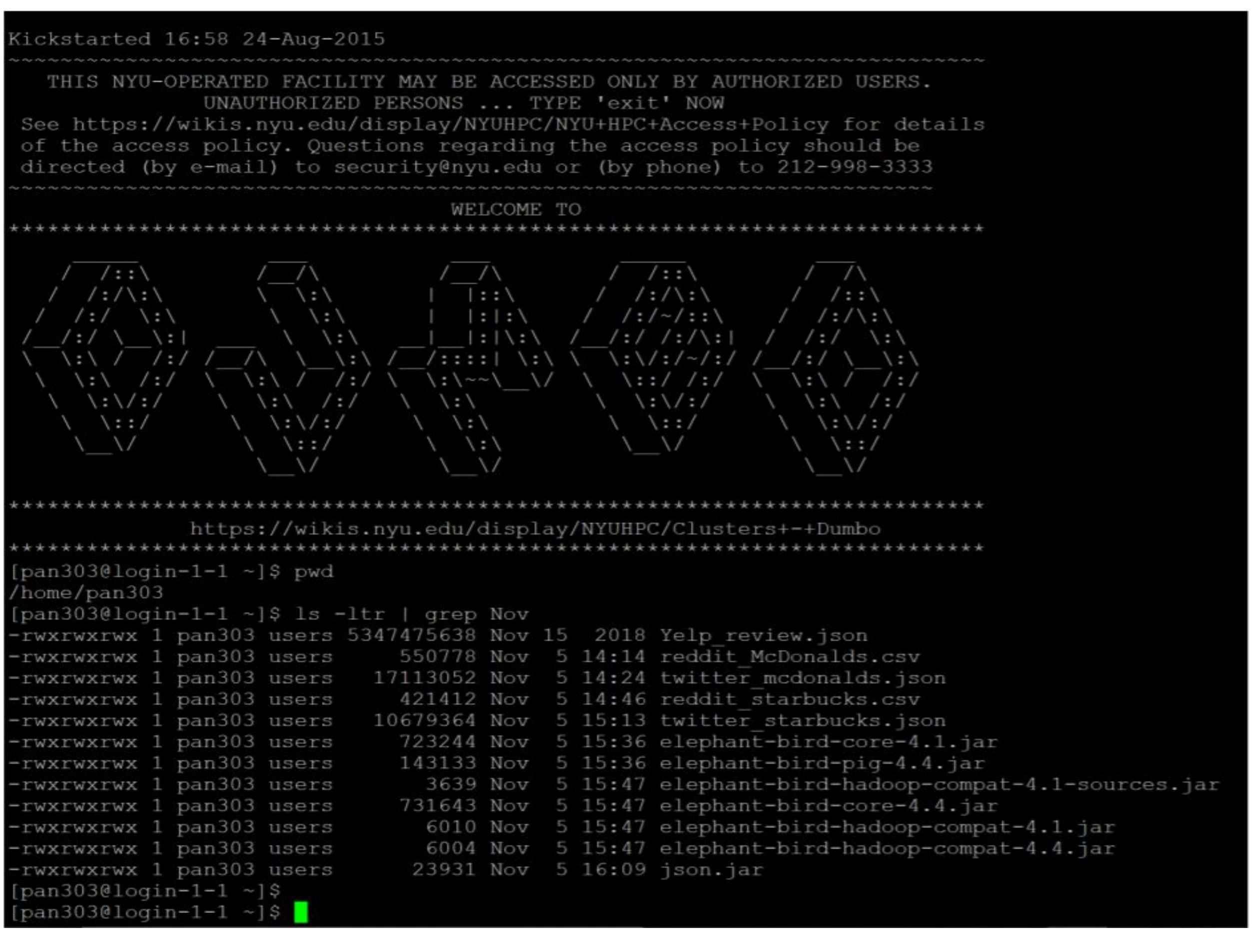
Step 1: Starting the WinSCP & connecting to NYU Dumbo Cluster as a remote host.



Step 2: Transferring the datafiles & related jar files to the local user directory first as follows:



Step 3: Setting up the necessary file permissions


Kickstarted 16:58 24-Aug-2015
THIS NYU-OPERATED FACILITY MAY BE ACCESSED ONLY BY AUTHORIZED USERS.
UNAUTHORIZED PERSONS ... TYPE 'exit' NOW
See https://wikis.nyu.edu/display/NYUHPC/NYU+HPC+Access+Policy for details
of the access policy. Questions regarding the access policy should be
directed (by e-mail) to security@nyu.edu or (by phone) to 212-998-3333
WELCOME TO
https://wikis.nyu.edu/display/NYUHPC/Clusters+-+Dumbo
[pan303@login-1-1 ~]$ pwd
/home/pan303
[pan303@login-1-1 ~]$ ls -ltr | grep Nov
-rwxrwxrwx 1 pan303 users 5347475638 Nov 15 2018 Yelp_review.json
-rwxrwxrwx 1 pan303 users   550778 Nov 5 14:14 reddit_McDonalds.csv
-rwxrwxrwx 1 pan303 users   17113052 Nov 5 14:24 twitter_mcdonalds.json
-rwxrwxrwx 1 pan303 users   421412 Nov 5 14:46 reddit_starbucks.csv
-rwxrwxrwx 1 pan303 users   10679364 Nov 5 15:13 twitter_starbucks.json
-rwxrwxrwx 1 pan303 users   723244 Nov 5 15:36 elephant-bird-core-4.1.jar
-rwxrwxrwx 1 pan303 users   143133 Nov 5 15:36 elephant-bird-pig-4.4.jar
-rwxrwxrwx 1 pan303 users   3639 Nov 5 15:47 elephant-bird-hadoop-compat-4.1-sources.jar
-rwxrwxrwx 1 pan303 users   731643 Nov 5 15:47 elephant-bird-core-4.4.jar
-rwxrwxrwx 1 pan303 users   6010 Nov 5 15:47 elephant-bird-hadoop-compat-4.1.jar
-rwxrwxrwx 1 pan303 users   6004 Nov 5 15:47 elephant-bird-hadoop-compat-4.4.jar
-rwxrwxrwx 1 pan303 users   23931 Nov 5 16:09 json.jar
[pan303@login-1-1 ~]$
[pan303@login-1-1 ~]$

Step 4: Loading the datafiles from user folders to the HDFS using the command:

>hdfs dfs -put <filename> <shared_directory_in_HDFS>

Here, Shared Directory is: /user/ss13449/project/

```
[pan303@login-1-1 ~]$
[pan303@login-1-1 ~]$ hdfs dfs -ls /user/ss13449/project/
Found 12 items
-rwxrwxrwx+  3 pan303  users 5347475638 2019-11-05 18:59 /user/ss13449/project/Yelp_review.json
drwxrwxr-x+  - ss13449 users          0 2019-11-05 16:10 /user/ss13449/project/jars
drwxr-xr-x+  - ss13449 users          0 2019-11-05 20:23 /user/ss13449/project/output
drwxr-xr-x+  - ss13449 users          0 2019-11-05 20:28 /user/ss13449/project/output1
drwxr-xr-x+  - kg2644  users          0 2019-11-05 22:47 /user/ss13449/project/output5
-rw-r--r--+  3 ss13449 users       7293 2019-11-05 18:58 /user/ss13449/project/projectinput.json
-rw-r--r--+  3 ss13449 users       3707 2019-11-05 19:10 /user/ss13449/project/projectinputyelp.json
-rw-r--r--+  3 pan303  users     550778 2019-11-05 16:02 /user/ss13449/project/reddit_McDonalds.csv
-rw-r--r--+  3 pan303  users     421412 2019-11-05 16:02 /user/ss13449/project/reddit_starbucks.csv
-rw-r--r--+  3 kg2644  users        270 2019-11-05 22:35 /user/ss13449/project/reddit_starbucks_mapper.py
-rwxrwxrwx+  3 pan303  users   17113052 2019-11-05 14:27 /user/ss13449/project/twitter_mcdonalds.json
-rwxrwxrwx+  3 pan303  users   10679364 2019-11-05 15:14 /user/ss13449/project/twitter_starbucks.json
[pan303@login-1-1 ~]$
[pan303@login-1-1 ~]$ pwd
/home/pan303
[pan303@login-1-1 ~]$
```

3. **Data Cleaning (ETL) and profiling code:**

Data cleaning has constituted the biggest time-consuming part of this project. Below are the mapper scripts for different data sources which are executed using Hadoop pipelines:

Datasource 1: **Yelp (JSON format)**

Below are the cleaning scripts for the Yelp dataset.

(mapper_clean_yelp.py, mapper_clean_yelpbusiness.py)

These mapper files are present at the below location: */user/ss13449/project/*

> cat /user/ss13449/project/mapper_clean_yelp.py

```
[kg2644@login-1-1 ~]$ hdfs dfs -cat /user/ss13449/project/mapper_clean_yelp.py
#!/usr/bin/env python2.6.6
# -*- coding: utf-8 -*-
import re
import sys
import json
import time
reload(sys)
sys.setdefaultencoding('utf-8')
rows=0
err=0
for line in sys.stdin:
        try:
                d=json.loads(line)
                key=str(d['review_id'])
                datestamp=time.strftime('%m-%d-%Y',time.strptime(d['date'],'%Y-%m-%d %H:%M:%S'))
                z = lambda x: re.compile('\#').sub('', re.compile('RT @[\w_]+:').sub('@', x).strip())
                text_removedRT=z(d['text'])
                text_removedLinks= re.sub(r'''(?i)\b((?:https?://|www\d{0,3}[.]|[a-z0-9.\-]+[.][a-z]{2,4}/)(?:[^\s()<>]+|\(([^\s()<>]+|(\([^\s()<>]+\)))*\))+(?:\(([^\s()<>]+|(\([^\s()<>]+\)))*\)|[^\s`!()\[\]{};:'".,<>?«»“”‘’]))''',"",text_removedRT)
                review=re.sub('[!:#`,.\'\"~$@%?/\r\n\t\[\]\(\)\{\}]','', text_removedLinks.encode('ascii','ignore').lower())
                businessid=str(d['business_id'])
                rating=int(d['stars'])
                print '%s,%s,"%s",%s,%d' % (key,datestamp,review,businessid,rating)
                rows=rows+1
        except:
                pass
```

Use the below command for code execution:

hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-streaming.jar -D mapreduce.job.reduces=0 -files hdfs://dumbo/user/ss13449/project/mapper_clean_yelp.py -mapper "python mapper_clean_yelp.py" -input /user/ss13449/project/data/Yelp.json -output /user/ss13449/project/cleaned_data/data_yelp/

Output files are created at the below locations:

/user/ss13449/project/cleaned_data/data_yelp/

```
[kg2644@login-1-1 ~]$ hdfs dfs -ls /user/ss13449/project/cleaned_data/data_yelp/
Found 33 items
-rw-r--r--+  3 ss13449 users          0 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/_SUCCESS
-rw-r--r--+  3 ss13449 users  108175090 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00000
-rw-r--r--+  3 ss13449 users  108213925 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00001
-rw-r--r--+  3 ss13449 users  108268893 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00002
-rw-r--r--+  3 ss13449 users  108835035 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00003
-rw-r--r--+  3 ss13449 users  108047005 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00004
-rw-r--r--+  3 ss13449 users  108105836 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00005
-rw-r--r--+  3 ss13449 users  108251852 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00006
-rw-r--r--+  3 ss13449 users  108851757 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00007
-rw-r--r--+  3 ss13449 users  108305903 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00008
-rw-r--r--+  3 ss13449 users  108313385 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00009
-rw-r--r--+  3 ss13449 users  108369786 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00010
-rw-r--r--+  3 ss13449 users  109019951 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00011
-rw-r--r--+  3 ss13449 users  108130500 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00012
-rw-r--r--+  3 ss13449 users  108167610 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00013
-rw-r--r--+  3 ss13449 users  108238440 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00014
-rw-r--r--+  3 ss13449 users  108910274 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00015
-rw-r--r--+  3 ss13449 users  108059102 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00016
-rw-r--r--+  3 ss13449 users  108040583 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00017
-rw-r--r--+  3 ss13449 users  108165756 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00018
-rw-r--r--+  3 ss13449 users  108761436 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00019
-rw-r--r--+  3 ss13449 users  108161845 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00020
-rw-r--r--+  3 ss13449 users  108222081 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00021
-rw-r--r--+  3 ss13449 users  108299044 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00022
-rw-r--r--+  3 ss13449 users  108892359 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00023
-rw-r--r--+  3 ss13449 users  108349925 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00024
-rw-r--r--+  3 ss13449 users  108392921 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00025
-rw-r--r--+  3 ss13449 users  108532300 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00026
-rw-r--r--+  3 ss13449 users  109098981 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00027
-rw-r--r--+  3 ss13449 users  108157420 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00028
-rw-r--r--+  3 ss13449 users  108148653 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00029
-rw-r--r--+  3 ss13449 users  108309079 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00030
-rw-r--r--+  3 ss13449 users   98514129 2019-11-16 19:52 /user/ss13449/project/cleaned_data/data_yelp/part-00031
[kg2644@login-1-1 ~]$
```

From the above screenshot, we can see that we have successfully executed our file.

> *cat /user/ss13449/project/mapper_clean_yelpbusiness.py*

```
[kg2644@login-1-1 ~]$ hdfs dfs -cat /user/ss13449/project/mapper_clean_yelpbusiness.py
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys
import json
import re
reload(sys)
sys.setdefaultencoding('utf-8')

for line in sys.stdin:
        try:
                d = json.loads(line)
                Business_id = str(d['business_id'])
                City = str(d['city']).encode('ascii','ignore')
                State =str(d['state'])
                Business_name=str(d['name']).encode('ascii','ignore')
                print '%s,%s,%s,%s' % (Business_id,Business_name,City,State)
        except:
                pass
[kg2644@login-1-1 ~]$
```

Use the below command for code execution:
hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-streaming.jar -D mapreduce.job.reduces=0 -files hdfs://dumbo/user/ss13449/project/ mapper_clean_yelpbusiness.py"
-mapper "python mapper_clean_yelpbusiness.py" -input
/user/ss13449/project/data/business.json -output /user/ss13449/project/cleaned_data/data_yelp_business/

Output files are created at the below locations:
/user/ss13449/project/cleaned_data/data_yelp_business

```
[kg2644@login-1-1 ~]$ hdfs dfs -ls /user/ss13449/project/cleaned_data/data_yelp_business
Found 3 items
-rw-r--r--+  3 ss13449 users          0 2019-11-14 17:16 /user/ss13449/project/cleaned_data/data_yelp_business/_SUCCESS
-rw-r--r--+  3 ss13449 users    5388078 2019-11-14 17:16 /user/ss13449/project/cleaned_data/data_yelp_business/part-00000
-rw-r--r--+  3 ss13449 users    5394547 2019-11-14 17:16 /user/ss13449/project/cleaned_data/data_yelp_business/part-00001
[kg2644@login-1-1 ~]$
```

Datasource 2: **Twitter (JSON Format)**
Below are the cleaning scripts for Twitter.
/user/ss13449/project/mapper_clean_twitter_mcdonalds.py
/user/ss13449/project/mapper_clean_twitter_starbucks.py

> cat */user/ss13449/project/mapper_clean_twitter_mcdonalds.py*
Below is the code for the above file :

```
[kg2644@login-1-1 ~]$ hdfs dfs -cat /user/ss13449/project/mapper_clean_twitter_mcdonalds.py
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import sys
import csv
import re
import datetime
import time
import csv
from csv import reader
#from importlib import reload
reload(sys)
sys.setdefaultencoding('utf-8')

for line in reader(sys.stdin):
        d=line
        key = d[0]
        try:
                d[2]=d[2].rstrip('\n')
                datestamp=time.strftime('%m-%d-%Y',time.strptime(d[2],'%m/%d/%Y %H:%M'))
        except:
                datestamp=''
        z = lambda x: re.compile('\#').sub('', re.compile('RT @[\w_]+:').sub('@', x).strip())
        text_removedRT=z(d[1])
        text_removedLinks= re.sub(r'''(?i)\b((?:https?://|www\d{0,3}[.]|[a-z0-9.\-]+[.][a-z]{2,4}/)(?:[^\s()<>]+|\(([^\s()<>]+|(\([^\s()<>]+\)))*\))+(?:\(([^\s()<>]+|(\([^\s()<>]+\)))*\
)|[^\s`!()\[\]{};:'".,<>?«»“”‘’]))''',"",text_removedRT)
        review=re.sub('[!#:`,.\'\"-*$@%?/\r\t\n\[\]\(\)\{\}]','', text_removedLinks.encode('ascii','ignore').lower())
                print '%s : %s : "%s" : McDonalds :  :  :  : ' % (key,datestamp,review)
[kg2644@login-1-1 ~]$
```

Use the below command for code execution:
hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-streaming.jar -D mapreduce.job.reduces=0 -files hdfs://dumbo/user/ss13449/project/mapper_clean_twitter_mcdonalds.py -mapper "python mapper_clean_twitter_mcdonalds.py" -input /user/ss13449/project/data/twitter_mcdonalds.json -output /user/ss13449/project/cleaned_data/data_twitter_mcdonalds/

Output files are created at the below locations:
/user/ss13449/project/cleaned_datadata_twitter_mcdonalds/

```
[kg2644@login-1-1 ~]$ hdfs dfs -ls /user/ss13449/project/cleaned_data/data_twitter_mcdonalds/
Found 3 items
-rw-r--r--+  3 kg2644 users          0 2019-11-17 22:00 /user/ss13449/project/cleaned_data/data_twitter_mcdonalds/_SUCCESS
-rw-r--r--+  3 kg2644 users     129787 2019-11-17 22:00 /user/ss13449/project/cleaned_data/data_twitter_mcdonalds/part-00000
-rw-r--r--+  3 kg2644 users     131813 2019-11-17 22:00 /user/ss13449/project/cleaned_data/data_twitter_mcdonalds/part-00001
[kg2644@login-1-1 ~]$
```

> *cat /user/ss13449/project/mapper_clean_twitter_starbucks.py*
Below is the code :

```
[kg2644@login-1-1 ~]$ hdfs dfs -cat /user/ss13449/project/mapper_clean_twitter_starbucks.py
#!/usr/bin/env python2.6.6
# -*- coding: utf-8 -*-
import re
import sys
import json
import time

for line in sys.stdin:
        try:
                d=json.loads(line)
                key=str(d['id_str'])
                datestamp=time.strftime('%m-%d-%Y',time.strptime(d['created_at'],'%a %b %d %H:%M:%S +0000 %Y'))
                z = lambda x: re.compile('\#').sub('', re.compile('RT @[\w_]+:').sub('@', x).strip())
                text_removedRT=z(d['text'])
                text_removedLinks= re.sub(r'''(?i)\b((?:https?://|www\d{0,3}[.]|[a-z0-9.\-]+[.][a-z]{2,4}/)(?:[^\s()<>]+|\(([^\s()<>]+|(\([^\s()<>]+\)))*\))+(?:\(([^\s()<>
]+|(\([^\s()<>]+\)))*\)|[^\s`!()\[\]{};:'".,<>?«»“”‘’]))''',"",text_removedRT)
                review=re.sub('[!#`:,.\'\"~$@%?/\t\r\n\[\]\(\)\{\}]','', text_removedLinks.encode('ascii','ignore').lower())
                business_name = str(d['user']['name'])
                print '%s : %s : "%s" : Starbucks :  :  :  : ' % (key,datestamp,review)
        except:
                pass
[kg2644@login-1-1 ~]$
```

For code execution, run the below command:
hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-streaming.jar -D mapreduce.job.reduces=0 -files hdfs://dumbo/user/ss13449/project/mapper_clean_twitter_starbucks.py -mapper "python mapper_clean_twitter_starbucks.py" -input /user/ss13449/project/data/twitter_starbucks.json -output /user/ss13449/project/cleaned_data/data_twitter_starbucks/

Output files are created at the below locations:
/user/ss13449/project/cleaned_datadata_twitter_starbucks/

```
[kg2644@login-1-1 ~]$ hdfs dfs -ls /user/ss13449/project/cleaned_data/data_twitter_starbucks/
Found 3 items
-rw-r--r--+  3 kg2644 users          0 2019-11-17 21:59 /user/ss13449/project/cleaned_data/data_twitter_starbucks/_SUCCESS
-rw-r--r--+  3 kg2644 users     114227 2019-11-17 21:59 /user/ss13449/project/cleaned_data/data_twitter_starbucks/part-00000
-rw-r--r--+  3 kg2644 users     112960 2019-11-17 21:59 /user/ss13449/project/cleaned_data/data_twitter_starbucks/part-00001
[kg2644@login-1-1 ~]$
```

Datasource 3: **Reddit (csv format)**
Below are the cleaning scripts for Reddit:
/user/ss13449/project/mapper_clean_reddit_mcdonalds.py
/user/ss13449/project/mapper_clean_reddit_starbucks.py

cat /user/ss13449/project/mapper_clean_reddit_mcdonalds.py
Below is the code for the above file :

```
[kg2644@login-1-1 ~]$ hdfs dfs -cat /user/ss13449/project/mapper_clean_reddit_mcdonalds.py
#!/usr/bin/python 2.6.6
# -*- coding: utf-8 -*-

import re
import time
import csv
from csv import reader
import sys
from datetime import datetime
reload(sys)
sys.setdefaultencoding('utf-8')

for line in reader(sys.stdin) :
        d=line
        key = d[0]
        try:
                d[2]=d[2].rstrip('\n')
                datestamp=time.strftime('%m-%d-%Y',time.strptime(d[2],'%m/%d/%Y %H:%M'))
        except:
                if len(d[2])!= 0:
                        continue
                datestamp=''
        z = lambda x: re.compile('\#').sub('', re.compile('RT @[\w_]+:').sub('@', x).strip())
        text_removedRT=z(d[1])
        text_removedLinks= re.sub(r'''(?i)\b((?:https?://|www\d{0,3}[.]|[a-z0-9.\-]+[.][a-z]{2,4}/)(?:[^\s()<>]+|\(([^\s()<>]+|(\([^\s()<>]+\)))*\))+(?:\(([^\s()<>]+|(\([^\s()<>]+\)))*\
)|[^\s`!()\[\]{};:'".,<>?«»“”‘’]))''',"",text_removedRT)
        review=re.sub('[!#:`,.\'\"~*$@%?/\r\t\n\[\]\(\)\{\}]','', text_removedLinks.encode('ascii','ignore').lower())
        print '%s : %s : "%s" : Starbucks :  :  :  :  ' % (key,datestamp,review)
[kg2644@login-1-1 ~]$
```

Use the below command for code execution:
hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-streaming.jar -D mapreduce.job.reduces=0 -files hdfs://dumbo/user/ss13449/project/mapper_clean_reddit_mcdonalds.py -mapper "python mapper_clean_reddit_mcdonalds.py" -input /user/ss13449/project/data/reddit_McDonalds.csv -output /user/ss13449/project/cleaned_data/data_reddit_mcdonalds/

Output files are created at the below locations:
/user/ss13449/project/cleaned_data/data_reddit_mcdonalds

```
[kg2644@login-1-1 ~]$ hdfs dfs -ls /user/ss13449/project/cleaned_data/data_reddit_mcdonalds/
Found 3 items
-rw-r--r--+  3 ss13449 users          0 2019-11-25 23:20 /user/ss13449/project/cleaned_data/data_reddit_mcdonalds/_SUCCESS
-rw-r--r--+  3 ss13449 users     315646 2019-11-25 23:20 /user/ss13449/project/cleaned_data/data_reddit_mcdonalds/part-00000
-rw-r--r--+  3 ss13449 users     318815 2019-11-25 23:20 /user/ss13449/project/cleaned_data/data_reddit_mcdonalds/part-00001
[kg2644@login-1-1 ~]$
```

cat /user/ss13449/project/mapper_clean_reddit_starbucks.py
Below is the code for the above file :

```
[kg2644@login-1-1 ~]$ hdfs dfs -cat /user/ss13449/project/mapper_clean_reddit_starbucks.py
#!/usr/bin/python 2.6.6
# -*- coding: utf-8 -*-

import re
import time
import csv
from csv import reader
import sys
reload(sys)
from datetime import datetime
sys.setdefaultencoding('utf-8')


for line in reader(sys.stdin) :
        d=line
        key = d[0]
        try:
                d[2]=d[2].rstrip('\n')
                date = datetime.strptime(d[2],"%Y-%m-%d %H:%M:%S")
                date_new = date.strftime('%d-%m-%Y')
                #datestamp=time.strftime('%m-%d-%Y',time.strptime(d[2],'%Y-%m-%d %H:%M'))
        except:
                date_new=''
        z = lambda x: re.compile('\#').sub('', re.compile('RT @[\w_]+:').sub('@', x).strip())
        text_removedRT=z(d[1])
        text_removedLinks= re.sub(r'''(?i)\b((?:https?://|www\d{0,3}[.]|[a-z0-9.\-]+[.][a-z]{2,4}/)(?:[^\s()<>]+|\(([^\s()<>]+|(\([^\s()<>]+\)))*\))+(?:\(([^\s()<>]+|(\([^\s()<>]+\)))*\
)|[^\s`!()\[\]{};:'".,<>?«»“”‘’]))''',' ',text_removedRT)
        review=re.sub('[!#:`,.\'\"-*$@%?/\;\t\n\[\]\(\)\{\}]','', text_removedLinks.encode('ascii','ignore').lower())
        print '%s : %s : "%s" : Starbucks :  :  :  : ' % (key,date_new,review)
[kg2644@login-1-1 ~]$
```

Use below command for code execution:
hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-streaming.jar -D mapreduce.job.reduces=0 -files hdfs://dumbo/user/ss13449/project/mapper_clean_reddit_starbucks.py -mapper "python mapper_clean_reddit_starbucks.py" -input /user/ss13449/project/data/reddit_starbucks.txt -output /user/ss13449/project/cleaned_data/data_reddit_starbucks

Output files are created at the below locations:
/user/ss13449/project/cleaned_data/data_reddit_starbucks

```
[kg2644@login-1-1 ~]$ hdfs dfs -ls /user/ss13449/project/cleaned_data/data_reddit_starbucks
Found 3 items
-rw-r--r--+  3 ss13449 users          0 2019-11-25 22:16 /user/ss13449/project/cleaned_data/data_reddit_starbucks/_SUCCESS
-rw-r--r--+  3 ss13449 users     248505 2019-11-25 22:16 /user/ss13449/project/cleaned_data/data_reddit_starbucks/part-00000
-rw-r--r--+  3 ss13449 users     248467 2019-11-25 22:16 /user/ss13449/project/cleaned_data/data_reddit_starbucks/part-00001
[kg2644@login-1-1 ~]$
```

So, till now we have cleaned the data that we have scrapped from Yelp, Twitter and Reddit. The data we have extracted contains the following columns like *userID, date, review, businessid, businessname, stars, city, state*

Data profiling:
Reddit Data Profiling using MapReduce:
mapper_profile.py has been used to for the Column Profiling of the data.

Command for Reddit Data Profiling:
hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-streaming.jar -D mapreduce.job.reduces=0 -files hdfs://dumbo/user/ss13449/project/mapper_profile.py -mapper "python mapper_profile.py" -input /user/ss13449/project/cleaned_data/reddit -output /user/ss13449/project/reddit_profile*

```
pan303@login-2-1:~
[pan303@login-2-1 ~]$ hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-streaming.jar \
> -D mapreduce.job.reduces=0 \
> -files hdfs://dumbo/user/ss13449/project/mapper_profile.py \
> -mapper "python mapper_profile.py" \
> -input /user/ss13449/project/output_reddit_cleaned_data/part-00000 \
> -output /user/ss13449/project/output_reddit_cleaned_data_profile
packageJobJar: [] [/opt/cloudera/parcels/CDH-5.15.2-1.cdh5.15.2.p0.3/jars/hadoop-streaming-2.6.0-cdh5.15.2.jar] /tmp/streamjob6521407989188451773.jar tmpDir=null
19/11/06 21:42:19 INFO mapred.FileInputFormat: Total input paths to process : 1
19/11/06 21:42:40 INFO hdfs.DFSClient: Exception in createBlockOutputStream
java.io.IOException: Bad connect ack with firstBadLink as 10.0.255.234:50010
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.createBlockOutputStream(DFSOutputStream.java:1785)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.nextBlockOutputStream(DFSOutputStream.java:1683)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:790)
19/11/06 21:42:40 WARN hdfs.DFSClient: Abandoning BP-1256044058-128.122.215.50-1440607644284:blk_1148829710_75201059
19/11/06 21:42:40 WARN hdfs.DFSClient: Excluding datanode DatanodeInfoWithStorage[10.0.255.234:50010,DS-ade6d62a-9e7d-4814-b6b5-68aa5683d6c2,DISK]
19/11/06 21:42:40 INFO mapreduce.JobSubmitter: number of splits:2
19/11/06 21:42:41 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1569350662793_36853
19/11/06 21:42:41 INFO impl.YarnClientImpl: Submitted application application_1569350662793_36853
19/11/06 21:42:41 INFO mapreduce.Job: The url to track the job: http://babar.es.its.nyu.edu:8088/proxy/application_1569350662793_36853/
19/11/06 21:42:41 INFO mapreduce.Job: Running job: job_1569350662793_36853
19/11/06 21:42:47 INFO mapreduce.Job: Job job_1569350662793_36853 running in uber mode : false
19/11/06 21:42:47 INFO mapreduce.Job:  map 0% reduce 0%
19/11/06 21:42:52 INFO mapreduce.Job:  map 100% reduce 0%
19/11/06 21:42:57 INFO mapreduce.Job: Job job_1569350662793_36853 completed successfully
19/11/06 21:42:57 INFO mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=315172
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=210387
                HDFS: Number of bytes written=1096
                HDFS: Number of read operations=10
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=4
        Job Counters
                Launched map tasks=2
                Data-local map tasks=2
                Total time spent by all maps in occupied slots (ms)=23392
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=5848
                Total vcore-milliseconds taken by all map tasks=5848
                Total megabyte-milliseconds taken by all map tasks=23953408
        Map-Reduce Framework
                Map input records=2490
                Map output records=50
                Input split bytes=246
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=83
                CPU time spent (ms)=1770
                Physical memory (bytes) snapshot=712323072
                Virtual memory (bytes) snapshot=7453806592
                Total committed heap usage (bytes)=2379218944
```

```
[pan303@login-2-1 ~]$ hdfs dfs -cat /user/ss13449/project/output_reddit_cleaned_data_profile/part-00000

Total rows of Review Data: 1232
Column Profiles for Twitter Data:
1) ID
   Data Type: String
   Max String Length: 6
   Max String Size(bytes): 46
   Range: Doesn't Apply
   Contains Null: No
   is_primary_key: Yes
2) Date
   Data Type: String
   Max String Length: 10
   Max String Size(bytes): 50
   Range: Doesn't Apply
   Contains Null: No
   is_primary_key: No
3) Review
   Data Type: String
   Max String Length: 279
   Max String Size(bytes): 319
   Range: Doesn't Apply
   Contains Null: No
   is_primary_key: No

[pan303@login-2-1 ~]$
[pan303@login-2-1 ~]$ date
Wed Nov  6 22:35:55 EST 2019
[pan303@login-2-1 ~]$
```

Yelp Data Profiling using MapReduce:

Command:

*hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-streaming.jar *
*-D mapreduce.job.reduces=0 *
*-files hdfs://dumbo/user/ss13449/project/mapper_profile.py *
*-mapper "python mapper_profile.py" *
*-input /user/ss13449/project/cleaned_data/yelp *
-output /user/ss13449/project/yelp_profile

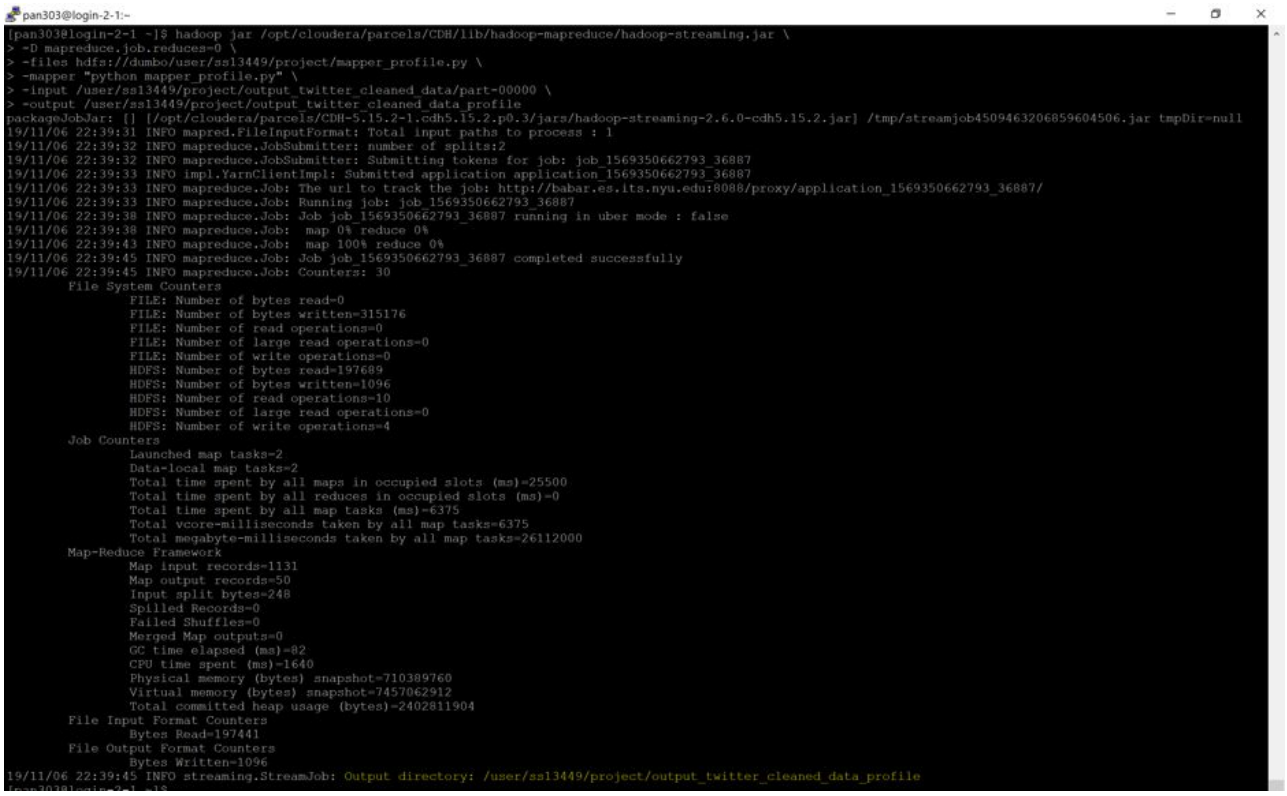
```
[pan303@login-2-1 ~]$ hdfs dfs -cat /user/ss13449/project/yelp_out/part-00000
Total rows of Review Data: 157296
Column Profiles for Twitter Data:
1) ID
   Data Type: String
   Max String Length: 22
   Max String Size(bytes): 62
   Range: Doesn't Apply
   Contains Null: No
   is_primary_key: Yes
2) Date
   Data Type: String
   Max String Length: 10
   Max String Size(bytes): 50
   Range: Doesn't Apply
   Contains Null: No
   is_primary_key: No
3) Review
   Data Type: String
   Max String Length: 5419
   Max String Size(bytes): 6459
   Range: Doesn't Apply
   Contains Null: No
   is_primary_key: No
[pan303@login-2-1 ~]$
[pan303@login-2-1 ~]$
[pan303@login-2-1 ~]$  hdfs dfs -ls /user/ss13449/project/yelp_out
Found 1 items
-rw-r--r--+  3 ss13449 users        526 2019-11-06 23:37 /user/ss13449/project/yelp_out/part-00000
[pan303@login-2-1 ~]$
[pan303@login-2-1 ~]$
```

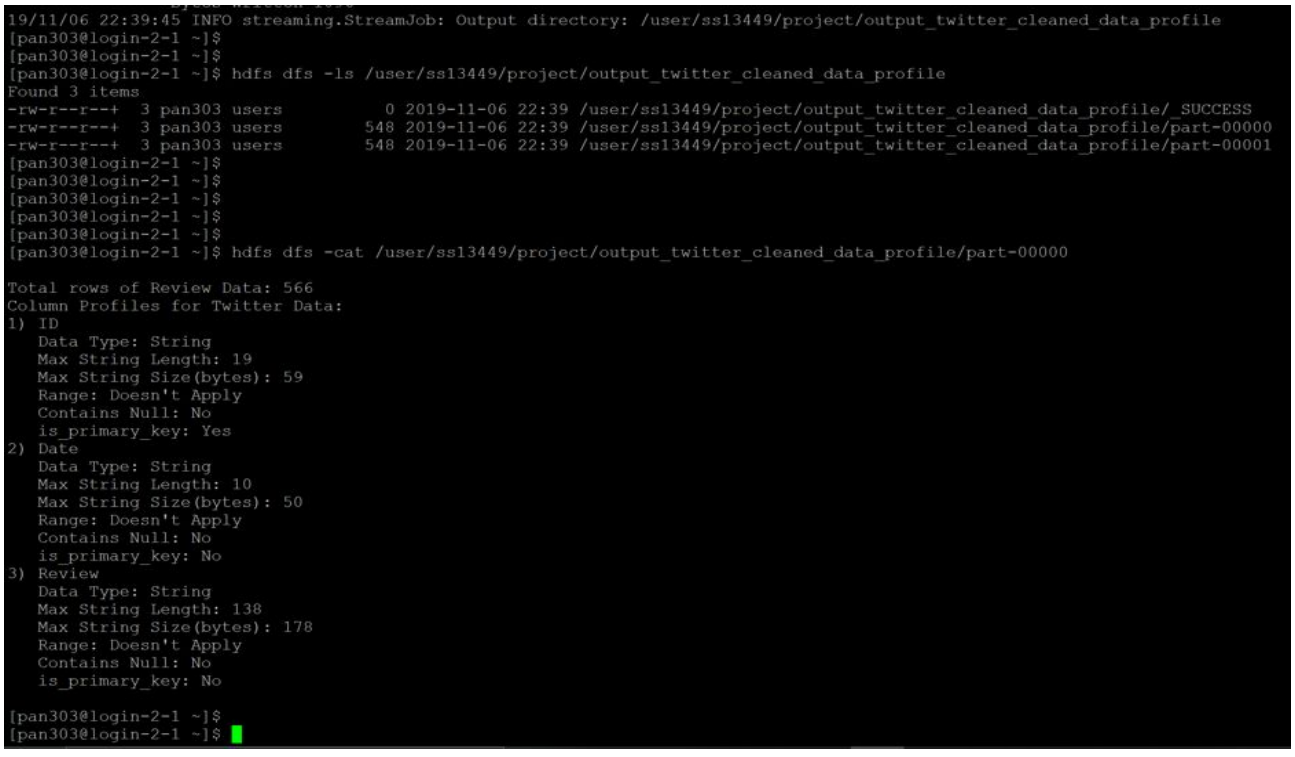
Data Profiling for Twitter data using MapReduce:

mapper_profile.py has been used to for the Column Profiling of the data.

Command:

*hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-streaming.jar *
*-D mapreduce.job.reduces=0 *
*-files hdfs://dumbo/user/ss13449/project/mapper_clean_yelp.py *
*-mapper "python mapper_clean_.py" *
-input /user/ss13449/project/cleaned_data/twitter *
-output /user/ss13449/project/twitter_profile





4. Data Analysis

Below are some of the questions that we have answered through in Data Analysis part are:

1. Finding the polarity of the reviews for Starbucks and McDonalds (using sentiment analysis).
2. Finding the average polarity of a particular restaurant for any other restaurant (in our case- McDonalds, Starbucks) and comparing them with the actual ratings. This may help in verifying the ratings on social platforms with ratings from reviews.
3. Finding the frequent topics about which reviewers are talking about. This may help in restaurants to analyze their brand image and take appropriate actions and business decisions upon user frequent topics.

4. Try to compare the average polarity of any restaurant over the different platforms like Yelp, Twitter, Reddit and see on which platform they are popular and are most likely liked by people.
5. Compare the user ratings and average polarity ratings for different states and compare the results.
6. Compare the average user ratings and average polarity ratings weekly, monthly and quarterly, to see how the restaurant's performance.

Below are the key challenges that we encountered while dealing with this.

1. We will be using some of the *NLTK libraries like TextBlob, stopwords*, *ngrams* but these libraries were not present on the node. We were not aware of this and our code was getting failed every time. So, we contact HPC and Wensheng from the HPC team cooperated with us and getting the NLTK libraries installed over the node.
2. Though the libraries were installed, but we could not easily run it over the node. A python script was needed to run it and we created a *python wrapper script* and use it with the mapper code in the pipeline and were successfully able to execute our codes.
3. We also faced the issues with internal(*manage table*) table and we were not able to fetch the data from manage table to text file, whereas it becomes easier for us to fetch the data from *external tables to text files*.
   > But after a lot of research over the internet and consulting with TAs we found out that external tables are a better way to transfer data from hive to hdfs.

So, we have probably used the below technologies for our project:

1. Hive
2. Impala
3. Map-Reducer

Table Creation and Merging through Hive For YELP:

create external table Yelp_data(id string, date_stamp string, reviews string, business_id string,stars float) row format delimited fields terminated by ',' LINES TERMINATED BY '\n' location '/user/ss13449/project/cleaned_data/data_yelp/';

-- Here an external table Yelp_data has been created on HIVE from the cleaned data which is present at the location : /user/ss13449/project/cleaned_data/data_yelp/

```
0: jdbc:hive2://babar.es.its.nyu.edu:10000/> describe yelp_data;
INFO  : Compiling command(queryId=hive_20191130184848_c32c9f97-4485-4a9d-b2cf-56df9232f8f5): describe yelp_data
INFO  : Semantic Analysis Completed
INFO  : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:col_name, type:string, comment:from deserializer), FieldSchema(name:data_type, type:string, comment:from deserializer), FieldSchema(name:comment, type:string, comment:from deserializer)], properties:null)
INFO  : Completed compiling command(queryId=hive_20191130184848_c32c9f97-4485-4a9d-b2cf-56df9232f8f5); Time taken: 0.159 seconds
INFO  : Executing command(queryId=hive_20191130184848_c32c9f97-4485-4a9d-b2cf-56df9232f8f5): describe yelp_data
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hive_20191130184848_c32c9f97-4485-4a9d-b2cf-56df9232f8f5); Time taken: 0.009 seconds
INFO  : OK
+--------------+------------+----------+--+
|   col_name   | data_type  | comment  |
+--------------+------------+----------+--+
| id           | string     |          |
| date_stamp   | string     |          |
| reviews      | string     |          |
| business_id  | string     |          |
| stars        | float      |          |
+--------------+------------+----------+--+
5 rows selected (0.256 seconds)
```

create external table Yelp_business_data(business_id string, business_name string, city string, state string) row format delimited fields terminated by ',' LINES TERMINATED BY '\n' location '/user/ss13449/project/cleaned_data/data_yelp_business/';

-- Here an external table Yelp_business_data has been created on HIVE from the cleaned data which is present at the location : /user/ss13449/project/cleaned_data/data_yelp_business/

```
0: jdbc:hive2://babar.es.its.nyu.edu:10000/> describe yelp_business_data;
INFO  : Compiling command(queryId=hive_20191130184949_9cda9b61-9908-41ab-bb71-b2d5a80dc411): describe yelp_business_data
INFO  : Semantic Analysis Completed
INFO  : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:col_name, type:string, comment:from deserializer), FieldSchema(name:data_type, type:string, comment:from deserializer), FieldSchema(name:comment, type:string, comment:from deserializer)], properties:null)
INFO  : Completed compiling command(queryId=hive_20191130184949_9cda9b61-9908-41ab-bb71-b2d5a80dc411); Time taken: 0.277 seconds
INFO  : Executing command(queryId=hive_20191130184949_9cda9b61-9908-41ab-bb71-b2d5a80dc411): describe yelp_business_data
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hive_20191130184949_9cda9b61-9908-41ab-bb71-b2d5a80dc411); Time taken: 0.008 seconds
INFO  : OK
+---------------+------------+----------+--+
|    col_name   | data_type  | comment  |
+---------------+------------+----------+--+
| business_id   | string     |          |
| business_name | string     |          |
| city          | string     |          |
| state         | string     |          |
+---------------+------------+----------+--+
4 rows selected (0.403 seconds)
0: jdbc:hive2://babar.es.its.nyu.edu:10000/>
```

create external table Yelp_review(id string, date_stamp date,reviews string,business_id string,business_name string,stars float,city string,state string)
STORED AS TEXTFILE
LOCATION 'hdfs://dumbo/user/ss13449/project/cleaned_data/tabledata/yelp';
-- Here an empty table Yelp_review has been created where the joined data from both the tables Yelp_data and Yelp_business_data will be stored and can also store them in Text File format

INSERT OVERWRITE TABLE Yelp_review select id, cast(to_date(from_unixtime(unix_timestamp(date_stamp,'dd-MM-yyyy'))) as date),reviews,yelp_data.business_id,yelp_business_data.business_name,stars,city,state from yelp_data JOIN yelp_business_data ON yelp_data.business_id=yelp_business_data.business_id;
-- Here we will overwrite the empty table Yelp_review that we have created and will store the required data into it. One thing here is to note that we have make the JOIN condition on business_id, because it is the only unique identifier in both the tables.

Taking Data in Text File from Hive:

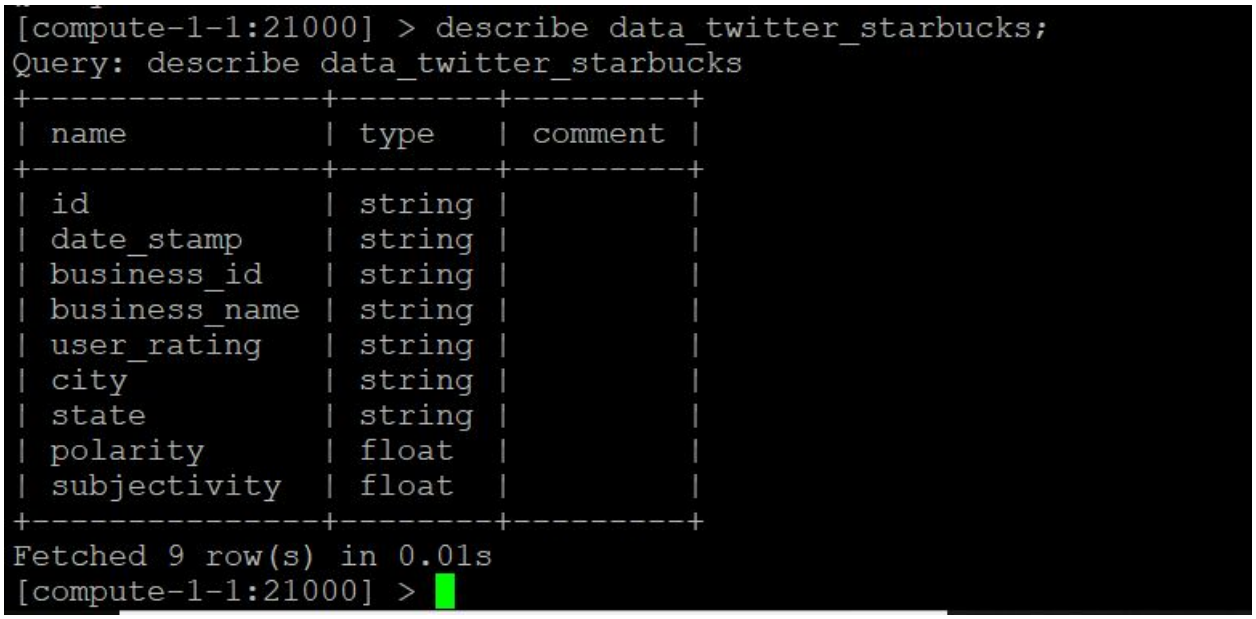
INSERT OVERWRITE DIRECTORY '/user/ss13449/project/cleaned_data/yelp' ROW FORMAT DELIMITED FIELDS TERMINATED BY ':' SELECT * FROM yelp_review;
-- We have finally overwrite our textfiles from the data which is present in the tables yelp_review, also see the table which is created in Hive.

```
0: jdbc:hive2://babar.es.its.nyu.edu:10000/> describe yelp_review;
INFO  : Compiling command(queryId=hive_20191130184444_bed60a5f-18a5-4ed0-863f-a3d7eb52a784): describe yelp_review
INFO  : Semantic Analysis Completed
INFO  : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:col_name, type:string, comment:from deserializer), FieldSchema(name:data_type, type:string, comment:from deserialize
r), FieldSchema(name:comment, type:string, comment:from deserializer)], properties:null)
INFO  : Completed compiling command(queryId=hive_20191130184444_bed60a5f-18a5-4ed0-863f-a3d7eb52a784); Time taken: 0.153 seconds
INFO  : Executing command(queryId=hive_20191130184444_bed60a5f-18a5-4ed0-863f-a3d7eb52a784): describe yelp_review
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hive_20191130184444_bed60a5f-18a5-4ed0-863f-a3d7eb52a784); Time taken: 0.008 seconds
INFO  : OK
+---------------+------------+----------+--+
|    col_name   | data_type  | comment  |
+---------------+------------+----------+--+
| id            | string     |          |
| date_stamp    | date       |          |
| reviews       | string     |          |
| business_id   | string     |          |
| business_name | string     |          |
| stars         | float      |          |
| city          | string     |          |
| state         | string     |          |
+---------------+------------+----------+--+
8 rows selected (0.309 seconds)
0: jdbc:hive2://babar.es.its.nyu.edu:10000/>
```

Table Creation in Impala for Twitter Data :

Similarly, we have created a table for Twitter just like we have created for Yelp, but this time we have created the tables in IMPALA.

create external table data_twitter_starbucks(id string, date_stamp string, business_id string, business_name string, user_rating string, city string, state string, polarity float, subjectivity float) row format delimited fields terminated by ':' location '/user/ss13449/project/cleaned_data/analysed_data/analysed_data_twitter_starbucks';



create external table data_twitter_mcdonalds(id string, date_stamp string, business_id string, business_name string, user_rating string, city string, state string, polarity float, subjectivity float) row format delimited fields terminated by ':' location '/user/ss13449/project/cleaned_data/analysed_data/analysed_data_twitter_mcdonalds/';

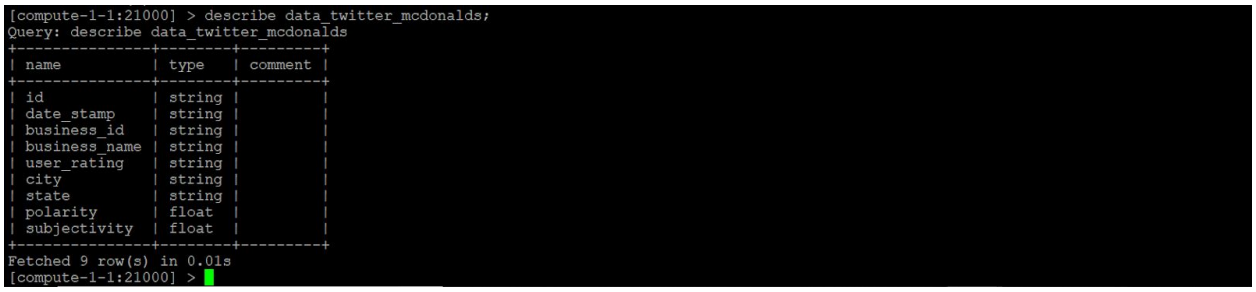


Table Creation in Impala for Reddit Data :

Similarly, we have created a table for Reddit just like we have created for Yelp, but this time we have created the tables in IMPALA.

create external table data_reddit_starbucks(id string, date_stamp string, business_id string, business_name string, user_rating string, city string, state string, polarity float, subjectivity float) row format delimited fields terminated by ':' location '/user/ss13449/project/cleaned_data/analysed_data/analysed_data_reddit_starbucks';

create external table data_reddit_mcdonalds(id string, date_stamp string, business_id string, business_name string, user_rating string, city string, state string, polarity float, subjectivity float) row format delimited fields terminated by ':' location '/user/ss13449/project/cleaned_data/analysed_data/analysed_data_reddit_mcdonalds';

```
[compute-1-1:21000] > describe data_reddit_mcdonalds;
Query: describe data_reddit_mcdonalds
+---------------+--------+---------+
| name          | type   | comment |
+---------------+--------+---------+
| id            | string |         |
| date_stamp    | string |         |
| business_id   | string |         |
| business_name | string |         |
| user_rating   | string |         |
| city          | string |         |
| state         | string |         |
| polarity      | float  |         |
| subjectivity  | float  |         |
+---------------+--------+---------+
Fetched 9 row(s) in 0.01s
[compute-1-1:21000] >
```

Now we will create an empty table data_for_analyses and will put the Reddit and Twitter data into this table. Below are the commands that we have used for this:

Insert into Table data_for_analyses select id, date_stamp, business_id, business_name, cast(user_rating as float), city, state, polarity, subjectivity from data_reddit_starbucks;

Insert into Table data_for_analyses select id, date_stamp, business_id, business_name, cast(user_rating as float), city, state, polarity, subjectivity from data_reddit_mcdonalds;

Insert into Table data_for_analyses select id, date_stamp, business_id, business_name, cast(user_rating as float), city, state, polarity, subjectivity from data_twitter_starbucks;

Insert into Table data_for_analyses select id, date_stamp, business_id, business_name, cast(user_rating as float), city, state, polarity, subjectivity from data_twitter_mcdonalds;

Till now we have created the tables and have placed all the cleaned data into it.
Now, let's run our analytics code, which will basically give us the polarity for each review. To run each analytics code using Hadoop pipeline, we will use the python Wrapper script which is present at the below location:

/user/ss13449/project/python_wrapper.sh

```
[kg2644@login-2-1 ~]$ hdfs dfs -cat /user/ss13449/project/python_wrapper.sh
#!/bin/bash
export NLTK_DATA=/share/apps/python/3.6.5/data/nltk
source /etc/profile.d/modules.sh
module purge
module load python/gnu/3.6.5
python $*
[kg2644@login-2-1 ~]$
```

Analytics Run for YELP :

hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-streaming.jar -D mapreduce.job.reduces=0 -files "python_wrapper.sh,analytics.py" -mapper "python_wrapper.sh analytics.py" -input /user/ss13449/project/cleaned_data/yelp -output /user/ss13449/project/cleaned_data/analysed_data

Analytics code:->

```
s [kg2644@login-2-1 ~]$ hdfs dfs -cat /user/ss13449/project/analytics.py
#!/user/bin/env python 3.6.5

import sys
import textblob
from textblob import TextBlob

try:
        for line in sys.stdin:
                lines = line.strip().split(':')
                blob = TextBlob(lines[2])
                print(lines[0],':',lines[1],':',lines[3],':',lines[4],':',lines[5],':',lines[6],':',lines[7],':',blob.sentiment[0],':',blob.sentiment[1])
except:
        pass
[kg2644@login-2-1 ~]$
```

Below is the screenshot, after successfully executing the above command:-->

```
[kg2644@login-2-1 ~]$ hdfs dfs -ls /user/ss13449/project/cleaned_data/analysed_data
Found 48 items
-rw-r--r--+  3 impala  users      97968 2019-11-26 05:10 /user/ss13449/project/cleaned_data/analysed_data/5b4d3e88db0967f8-a430dca00000000_1361128615_data.0.
-rw-r--r--+  3 impala  users      96004 2019-11-26 05:10 /user/ss13449/project/cleaned_data/analysed_data/5b4d3e88db0967f8-a430dca00000001_1970647877_data.0.
-rw-r--r--+  3 impala  users     175469 2019-11-25 23:34 /user/ss13449/project/cleaned_data/analysed_data/67462653dbe2c7fb-7121313600000000_834729423_data.0.
-rw-r--r--+  3 impala  users     174211 2019-11-25 23:34 /user/ss13449/project/cleaned_data/analysed_data/67462653dbe2c7fb-7121313600000001_2090129110_data.0.
-rw-r--r--+  3 impala  users     180078 2019-11-26 05:05 /user/ss13449/project/cleaned_data/analysed_data/704a60f6d367340a-87e8c32800000000_1640151534_data.0.
-rw-r--r--+  3 impala  users     186847 2019-11-26 05:05 /user/ss13449/project/cleaned_data/analysed_data/704a60f6d367340a-87e8c32800000001_1552240890_data.0.
-rw-r--r--+  3 impala  users     175469 2019-11-26 05:08 /user/ss13449/project/cleaned_data/analysed_data/8c468419201f1876-f46ea93800000000_909437756_data.0.
-rw-r--r--+  3 impala  users     174211 2019-11-26 05:08 /user/ss13449/project/cleaned_data/analysed_data/8c468419201f1876-f46ea93800000001_909437756_data.0.
-rw-r--r--+  3 ss13449 users          0 2019-11-17 00:50 /user/ss13449/project/cleaned_data/analysed_data/_SUCCESS
drwxrwxr-x+  - impala  users          0 2019-11-26 05:10 /user/ss13449/project/cleaned_data/analysed_data/_impala_insert_staging
drwxr-xr-x+  - ss13449 users          0 2019-11-25 23:21 /user/ss13449/project/cleaned_data/analysed_data/analysed_data_reddit_mcdonalds
drwxr-xr-x+  - ss13449 users          0 2019-11-25 22:30 /user/ss13449/project/cleaned_data/analysed_data/analysed_data_reddit_starbucks
drwxrwxrwx+  - kg2644  users          0 2019-11-17 22:04 /user/ss13449/project/cleaned_data/analysed_data/analysed_data_twitter_mcdonalds
drwxrwxrwx+  - kg2644  users          0 2019-11-17 22:01 /user/ss13449/project/cleaned_data/analysed_data/analysed_data_twitter_starbucks
-rw-r--r--+  3 impala  users      64620 2019-11-26 05:09 /user/ss13449/project/cleaned_data/analysed_data/d3489dd702e242c9-60135deb00000000_863814299_data.0.
-rw-r--r--+  3 impala  users      63939 2019-11-26 05:09 /user/ss13449/project/cleaned_data/analysed_data/d3489dd702e242c9-60135deb00000001_2021977668_data.0.
-rw-r--r--+  3 ss13449 users   28816679 2019-11-17 00:49 /user/ss13449/project/cleaned_data/analysed_data/part-00000
-rw-r--r--+  3 ss13449 users   28882119 2019-11-17 00:49 /user/ss13449/project/cleaned_data/analysed_data/part-00001
-rw-r--r--+  3 ss13449 users   28644504 2019-11-17 00:50 /user/ss13449/project/cleaned_data/analysed_data/part-00002
-rw-r--r--+  3 ss13449 users   28376240 2019-11-17 00:49 /user/ss13449/project/cleaned_data/analysed_data/part-00003
-rw-r--r--+  3 ss13449 users   28498548 2019-11-17 00:49 /user/ss13449/project/cleaned_data/analysed_data/part-00004
-rw-r--r--+  3 ss13449 users   28287635 2019-11-17 00:49 /user/ss13449/project/cleaned_data/analysed_data/part-00005
-rw-r--r--+  3 ss13449 users   27910164 2019-11-17 00:49 /user/ss13449/project/cleaned_data/analysed_data/part-00006
-rw-r--r--+  3 ss13449 users   28569315 2019-11-17 00:49 /user/ss13449/project/cleaned_data/analysed_data/part-00007
-rw-r--r--+  3 ss13449 users   28200017 2019-11-17 00:49 /user/ss13449/project/cleaned_data/analysed_data/part-00008
-rw-r--r--+  3 ss13449 users   28025696 2019-11-17 00:49 /user/ss13449/project/cleaned_data/analysed_data/part-00009
-rw-r--r--+  3 ss13449 users   28591428 2019-11-17 00:49 /user/ss13449/project/cleaned_data/analysed_data/part-00010
-rw-r--r--+  3 ss13449 users   28338877 2019-11-17 00:49 /user/ss13449/project/cleaned_data/analysed_data/part-00011
-rw-r--r--+  3 ss13449 users   27921502 2019-11-17 00:49 /user/ss13449/project/cleaned_data/analysed_data/part-00012
-rw-r--r--+  3 ss13449 users   28674013 2019-11-17 00:49 /user/ss13449/project/cleaned_data/analysed_data/part-00013
-rw-r--r--+  3 ss13449 users   28630896 2019-11-17 00:49 /user/ss13449/project/cleaned_data/analysed_data/part-00014
-rw-r--r--+  3 ss13449 users   28487630 2019-11-17 00:49 /user/ss13449/project/cleaned_data/analysed_data/part-00015
-rw-r--r--+  3 ss13449 users   27774275 2019-11-17 00:49 /user/ss13449/project/cleaned_data/analysed_data/part-00016
-rw-r--r--+  3 ss13449 users   27608637 2019-11-17 00:49 /user/ss13449/project/cleaned_data/analysed_data/part-00017
-rw-r--r--+  3 ss13449 users   28649696 2019-11-17 00:49 /user/ss13449/project/cleaned_data/analysed_data/part-00018
-rw-r--r--+  3 ss13449 users   28640113 2019-11-17 00:49 /user/ss13449/project/cleaned_data/analysed_data/part-00019
-rw-r--r--+  3 ss13449 users   23582580 2019-11-17 00:49 /user/ss13449/project/cleaned_data/analysed_data/part-00020
-rw-r--r--+  3 ss13449 users   24214316 2019-11-17 00:49 /user/ss13449/project/cleaned_data/analysed_data/part-00021
-rw-r--r--+  3 ss13449 users   24401108 2019-11-17 00:49 /user/ss13449/project/cleaned_data/analysed_data/part-00022
-rw-r--r--+  3 ss13449 users   19861952 2019-11-17 00:48 /user/ss13449/project/cleaned_data/analysed_data/part-00023
-rw-r--r--+  3 ss13449 users   17110169 2019-11-17 00:48 /user/ss13449/project/cleaned_data/analysed_data/part-00024
-rw-r--r--+  3 ss13449 users   15648609 2019-11-17 00:48 /user/ss13449/project/cleaned_data/analysed_data/part-00025
-rw-r--r--+  3 ss13449 users   15701248 2019-11-17 00:48 /user/ss13449/project/cleaned_data/analysed_data/part-00026
-rw-r--r--+  3 ss13449 users   15571149 2019-11-17 00:48 /user/ss13449/project/cleaned_data/analysed_data/part-00027
-rw-r--r--+  3 ss13449 users   15625769 2019-11-17 00:48 /user/ss13449/project/cleaned_data/analysed_data/part-00028
-rw-r--r--+  3 ss13449 users   11432389 2019-11-17 00:47 /user/ss13449/project/cleaned_data/analysed_data/part-00029
-rw-r--r--+  3 ss13449 users   11295448 2019-11-17 00:47 /user/ss13449/project/cleaned_data/analysed_data/part-00030
-rw-r--r--+  3 ss13449 users    7183301 2019-11-17 00:47 /user/ss13449/project/cleaned_data/analysed_data/part-00031
[kg2644@login-2-1 ~]$
```

We have to make the ':' separated files, which will make it easier for us to make the processing going forward easier.

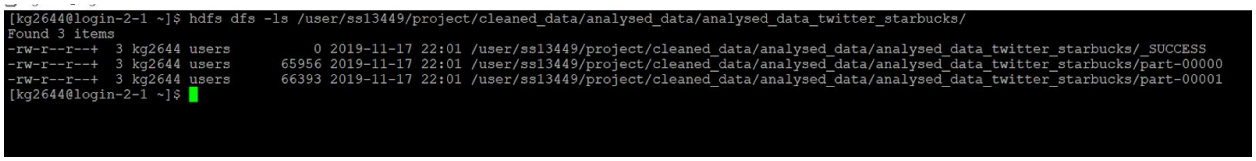
Analytics Run for Twitter:

Our analytics code will remain the same for all the analysis that we will be doing, only the input path gets changed in the command line.

Starbucks:

hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-streaming.jar -D mapreduce.job.reduces=0 -files "python_wrapper.sh,analytics.py" -mapper "python_wrapper.sh analytics.py" -input /user/ss13449/project/cleaned_data/data_twitter_starbucks -output /user/ss13449/project/cleaned_data/analysed_data/analysed_data_twitter_starbucks/

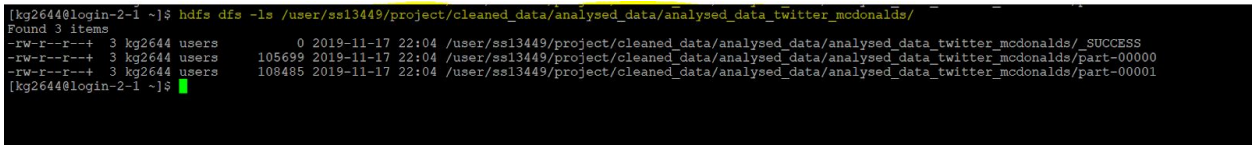
Below is the screenshot, after successfully executing the above command:



McDonald's:

hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-streaming.jar -D mapreduce.job.reduces=0 -files "python_wrapper.sh,analytics.py" -mapper "python_wrapper.sh analytics.py" -input /user/ss13449/project/cleaned_data/data_twitter_mcdonalds -output /user/ss13449/project/cleaned_data/analysed_data/analysed_data_twitter_mcdonalds/

Below is the screenshot, after successfully executing the above command:



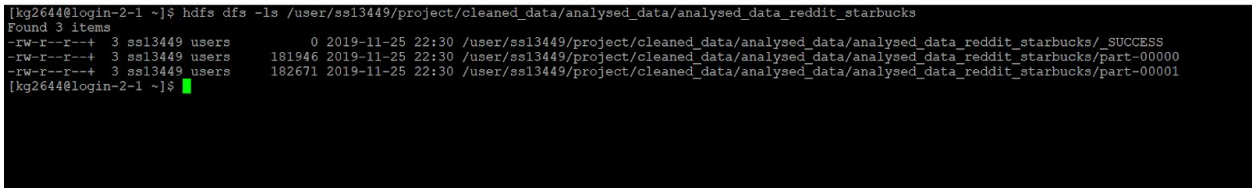
Analytics Run for Reddit:

Our analytics code will remain the same for all the analysis that we will be doing, only the input path gets changed in the command line.

Starbucks:

hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-streaming.jar -D mapreduce.job.reduces=0 -files "python_wrapper.sh,analytics.py" -mapper "python_wrapper.sh analytics.py" -input /user/ss13449/project/cleaned_data/data_reddit_starbucks/ -output /user/ss13449/project/cleaned_data/analysed_data/analysed_data_reddit_starbucks

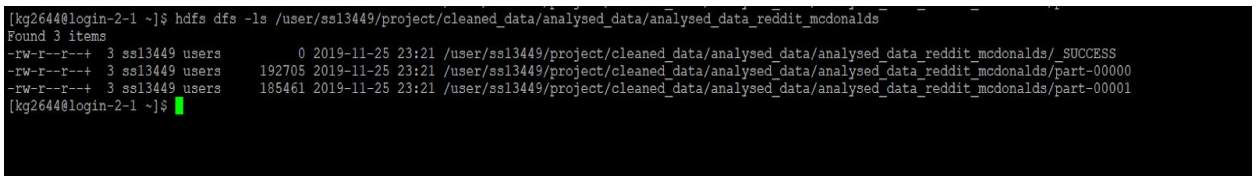
Below is the screenshot, after successfully executing the above command:



McDonalds:

hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-streaming.jar -D mapreduce.job.reduces=0 -files "python_wrapper.sh,analytics.py" -mapper "python_wrapper.sh analytics.py" -input /user/ss13449/project/cleaned_data/data_reddit_mcdonalds/ -output /user/ss13449/project/cleaned_data/analysed_data/analysed_data_reddit_mcdonalds

Below is the screenshot, after successfully executing the above command:



***Analyses 1 & 2*:** Our first and second analyses is to find the average rating(that we will get through polarity) for Starbucks and McDonald's for all the data that we have gathered and compare them with the actual average rating.

***Solution*:** We will be using Impala for faster processing of the data. Below are the commands that we will be using:

Select avg(user_rating) as average_user_rating, avg((polarity*5)+5)/2 as average_polarity from data_for_analyses where business_name like "%Starbucks%" and user_rating is not null;

Here we have used math to compute the rating. Polarity results after sentiment analysis comes in between [-1,+1] and so to push it in between [0,5], first multiply by 5, then add 5 and divide the ratings by 2. This will give us the rating in the range [0,5].

```
[compute-1-1:21000] > Select avg(user_rating) as average_user_rating, avg((polarity*5)+5)/2 as average_polarity from data_for_analyses where business_name like "%Starbucks%" and user_rating is not null;
Query: Select avg(user_rating) as average_user_rating, avg((polarity*5)+5)/2 as average_polarity from data_for_analyses where business_name like "%Starbucks%" and user_rating is not null
Query submitted at: 2019-11-30 18:16:48 (Coordinator: http://compute-1-1.local:25000)
Query progress can be monitored at: http://compute-1-1.local:25000/query_plan?query_id=b04bd27f2fdea224:19c51fa00000000
+--------------------+--------------------+
| average_user_rating | average_polarity  |
+--------------------+--------------------+
| 3.084090344998759  | 2.827585428025279 |
+--------------------+--------------------+
Fetched 1 row(s) in 0.73s
```

Select avg(user_rating) as average_user_rating, avg((polarity*5)+5)/2 as average_polarity from data_for_analyses where business_name like "%McDonalds%" and user_rating is not null;

```
[compute-1-1:21000] > Select avg(user_rating) as average_user_rating, avg((polarity*5)+5)/2 as average_polarity from data_for_analyses where business_name like "%McDonalds%" and user_rating is not null;
Query: Select avg(user_rating) as average_user_rating, avg((polarity*5)+5)/2 as average_polarity from data_for_analyses where business_name like "%McDonalds%" and user_rating is not null
Query submitted at: 2019-11-30 18:16:53 (Coordinator: http://compute-1-1.local:25000)
Query progress can be monitored at: http://compute-1-1.local:25000/query_plan?query_id=2c45d2f63922a98a:3d71f6ec00000000
+--------------------+--------------------+
| average_user_rating | average_polarity  |
+--------------------+--------------------+
| 2.125              | 2.510612693991349 |
+--------------------+--------------------+
Fetched 1 row(s) in 0.31s
```

From the above, we can see that for Starbucks the average user rating comes out to be 3.0 whereas the average polarity rating comes out to be 2.82, and for Mcdonald's average user rating is 2.1 whereas average polarity comes out to be 2.5. We can see that the average polarity results are close to the average user ratings, keeping the error margin of +/- 0.5.

Analyses 3: This involves making a word count program, to see which are those particular items that are most and least liked by the people (as in their reviews). This will basically help the restaurants in letting them know which are the things they are good at and in which they are not?

Solution: We have prepared the below code for the word count analysis. We have also made use of stopwords, as we don't want to count the common words.

STARBUCKS :
The code is present at location:

/user/ss13449/project/analytics_starbucks.py

```
[ss13449@login-1-1 ~]$ hdfs dfs -cat /user/ss13449/project/analytics_starbucks.py
#!/usr/bin/env/python 3.6.5

import sys, re
from nltk.corpus import stopwords
from nltk.util import ngrams
businessName = 'starbucks'
twograms = dict()
stopword = set(stopwords.words('english'))
for line in sys.stdin:
        d = line.strip(" ").split(':')
        if businessName in d[3].lower() or businessName in d[4].lower():
                s = d[2].lower().replace('"', '')
                tokens = [token for token in s.split(" ") if token != ""]
                val = list(ngrams(tokens, 2))
                for i in val:
                        x,y = i
                        if x not in stopword and y not in stopword:
                                tup = x.replace('\'','') + " " + y.replace('\'','')
                                if tup in twograms:
                                        twograms[tup] += 1
                                else:
                                        twograms[tup] = 1
print("\n".join("{!r}:{!r}".format(k, v) for k, v in twograms.items()))
#print(sorted(twograms.items(), key=lambda x: x[1], reverse=True))
```

Command used to calculate the word count program for Starbucks data is:

hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-streaming.jar -D mapreduce.job.reduces=0 -files "python_wrapper.sh,analytics_starbucks.py" -mapper "python_wrapper.sh analytics_starbucks.py" -input /user/ss13449/project/cleaned_data/all_data/ -output /user/ss13449/project/cleaned_data/word_count_starbucks

Below is the result for the same:

```
[kg2644@login-1-1 ~]$ hdfs dfs -ls  /user/ss13449/project/cleaned_data/word_count_starbucks
Found 41 items
-rw-r--r--+  3 kg2644 users          0 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/_SUCCESS
-rw-r--r--+  3 kg2644 users     251425 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00000
-rw-r--r--+  3 kg2644 users     254969 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00001
-rw-r--r--+  3 kg2644 users     198073 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00002
-rw-r--r--+  3 kg2644 users     214348 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00003
-rw-r--r--+  3 kg2644 users     190359 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00004
-rw-r--r--+  3 kg2644 users     215360 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00005
-rw-r--r--+  3 kg2644 users     196796 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00006
-rw-r--r--+  3 kg2644 users     248437 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00007
-rw-r--r--+  3 kg2644 users     225696 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00008
-rw-r--r--+  3 kg2644 users     217065 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00009
-rw-r--r--+  3 kg2644 users     210765 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00010
-rw-r--r--+  3 kg2644 users     216429 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00011
-rw-r--r--+  3 kg2644 users     194126 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00012
-rw-r--r--+  3 kg2644 users     235470 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00013
-rw-r--r--+  3 kg2644 users     213260 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00014
-rw-r--r--+  3 kg2644 users     214739 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00015
-rw-r--r--+  3 kg2644 users     222239 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00016
-rw-r--r--+  3 kg2644 users     206330 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00017
-rw-r--r--+  3 kg2644 users     210840 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00018
-rw-r--r--+  3 kg2644 users     217729 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00019
-rw-r--r--+  3 kg2644 users     192200 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00020
-rw-r--r--+  3 kg2644 users     225289 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00021
-rw-r--r--+  3 kg2644 users     216775 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00022
-rw-r--r--+  3 kg2644 users     141690 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00023
-rw-r--r--+  3 kg2644 users     117826 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00024
-rw-r--r--+  3 kg2644 users     144724 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00025
-rw-r--r--+  3 kg2644 users     105121 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00026
-rw-r--r--+  3 kg2644 users     133878 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00027
-rw-r--r--+  3 kg2644 users     141548 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00028
-rw-r--r--+  3 kg2644 users      76930 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00029
-rw-r--r--+  3 kg2644 users     105135 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00030
-rw-r--r--+  3 kg2644 users      47248 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00031
-rw-r--r--+  3 kg2644 users      88859 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00032
-rw-r--r--+  3 kg2644 users      88561 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00033
-rw-r--r--+  3 kg2644 users      84317 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00034
-rw-r--r--+  3 kg2644 users     108932 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00035
-rw-r--r--+  3 kg2644 users          4 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00036
-rw-r--r--+  3 kg2644 users          4 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00037
-rw-r--r--+  3 kg2644 users      65943 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00038
-rw-r--r--+  3 kg2644 users      64256 2019-12-01 18:44 /user/ss13449/project/cleaned_data/word_count_starbucks/part-00039
[kg2644@login-1-1 ~]$
```

McDonalds:-

Code is present at location:

/user/ss13449/project/analytics_mcdonalds.py and below is the code for the same.

```
[ss13449@login-1-1 ~]$ hdfs dfs -cat /user/ss13449/project/analytics_mcdonalds.py
#!/usr/bin/env/python 3.6.5

import sys, re
from nltk.corpus import stopwords
from nltk.util import ngrams
businessName = 'mcdonalds'
twograms = dict()
stopword = set(stopwords.words('english'))
for line in sys.stdin:
        d = line.strip(" ").split(':')
        if businessName in d[3].lower() or businessName in d[4].lower():
                s = d[2].lower().replace('"', '')
                tokens = [token for token in s.split(" ") if token != ""]
                val = list(ngrams(tokens, 2))
                for i in val:
                        x,y = i
                        if x not in stopword and y not in stopword:
                                tup = x.replace('\'','') + " " + y.replace('\'','')
                                if tup in twograms:
                                        twograms[tup] += 1
                                else:
                                        twograms[tup] = 1
print("\n".join("{!r}:{!r}".format(k, v) for k, v in twograms.items()))
#print(sorted(twograms.items(), key=lambda x: x[1], reverse=True))
```

Command used to calculate the word count program for McDonald's data is:

hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-streaming.jar -D mapreduce.job.reduces=0 -files "python_wrapper.sh,analytics_mcdonalds.py" -mapper "python_wrapper.sh analytics_mcdonalds.py" -input /user/ss13449/project/cleaned_data/all_data/ -output /user/ss13449/project/cleaned_data/word_count_mcdonalds

Below is the result for the same:-->

```
[kg2644@login-1-1 ~]$ hdfs dfs -ls /user/ss13449/project/cleaned_data/word_count_mcdonalds
Found 41 items
-rw-r--r--+  3 kg2644 users          0 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/_SUCCESS
-rw-r--r--+  3 kg2644 users      10323 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00000
-rw-r--r--+  3 kg2644 users       2351 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00001
-rw-r--r--+  3 kg2644 users       7547 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00002
-rw-r--r--+  3 kg2644 users       3144 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00003
-rw-r--r--+  3 kg2644 users       5403 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00004
-rw-r--r--+  3 kg2644 users       2297 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00005
-rw-r--r--+  3 kg2644 users       2862 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00006
-rw-r--r--+  3 kg2644 users      10123 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00007
-rw-r--r--+  3 kg2644 users       7139 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00008
-rw-r--r--+  3 kg2644 users       4740 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00009
-rw-r--r--+  3 kg2644 users       6771 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00010
-rw-r--r--+  3 kg2644 users       3679 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00011
-rw-r--r--+  3 kg2644 users       2685 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00012
-rw-r--r--+  3 kg2644 users       3403 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00013
-rw-r--r--+  3 kg2644 users       3617 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00014
-rw-r--r--+  3 kg2644 users       2937 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00015
-rw-r--r--+  3 kg2644 users       1351 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00016
-rw-r--r--+  3 kg2644 users       3068 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00017
-rw-r--r--+  3 kg2644 users       3031 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00018
-rw-r--r--+  3 kg2644 users       5227 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00019
-rw-r--r--+  3 kg2644 users       2163 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00020
-rw-r--r--+  3 kg2644 users       1308 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00021
-rw-r--r--+  3 kg2644 users        356 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00022
-rw-r--r--+  3 kg2644 users       1577 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00023
-rw-r--r--+  3 kg2644 users       1858 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00024
-rw-r--r--+  3 kg2644 users        895 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00025
-rw-r--r--+  3 kg2644 users       2149 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00026
-rw-r--r--+  3 kg2644 users        922 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00027
-rw-r--r--+  3 kg2644 users       3677 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00028
-rw-r--r--+  3 kg2644 users       2978 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00029
-rw-r--r--+  3 kg2644 users        637 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00030
-rw-r--r--+  3 kg2644 users        534 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00031
-rw-r--r--+  3 kg2644 users          4 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00032
-rw-r--r--+  3 kg2644 users          4 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00033
-rw-r--r--+  3 kg2644 users          4 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00034
-rw-r--r--+  3 kg2644 users          4 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00035
-rw-r--r--+  3 kg2644 users      47229 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00036
-rw-r--r--+  3 kg2644 users      46569 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00037
-rw-r--r--+  3 kg2644 users          4 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00038
-rw-r--r--+  3 kg2644 users          4 2019-12-01 18:46 /user/ss13449/project/cleaned_data/word_count_mcdonalds/part-00039
```

Analyses of Top topics in Impala:

Data is imported into the impala for sorting. The following command is used:

create external table fc_starbucks(feature string, count int) row format delimited fields terminated by ':' location '/user/ss13449/project/cleaned_data/word_count_starbucks/';

create external table fc_mcdonalds(feature string, count int) row format delimited fields terminated by ':' location '/user/ss13449/project/cleaned_data/word_count_mcdonalds/';

To combine all topics and finding all top reviews:

select trim(feature) as features,sum(count) as frequency from fc_starbucks where count > 1 group by features order by sum(count) desc limit 10;

select trim(feature) as features,sum(count) as frequency from fc_mcdonalds where count > 1 group by features order by sum(count) desc limit 10;

Following results are obtained:
Starbucks:

```
[compute-1-1:21000] > select trim(feature) as features,sum(count) as frequency from fc_starbucks where count > 1 group by features order by sum(count) desc limit 10;
Query: select trim(feature) as features,sum(count) as frequency from fc_starbucks where count > 1 group by features order by sum(count) desc limit 10
Query submitted at: 2019-12-01 22:13:12 (Coordinator: http://compute-1-1.local:25000)
Query progress can be monitored at: http://compute-1-1.local:25000/query_plan?query_id=8140634f71f895d7:2e758a8b00000000
+----------------------+-----------+
| features             | frequency |
+----------------------+-----------+
| 'drive thru'         | 2948      |
| 'customer service'   | 1868      |
| 'every time'         | 802       |
| 'starbucks location' | 720       |
| 'parking lot'        | 619       |
| 'iced coffee'        | 560       |
| 'green tea'          | 537       |
| 'worst starbucks'    | 478       |
| 'ive ever'           | 473       |
| 'dont know'          | 447       |
+----------------------+-----------+
Fetched 10 row(s) in 0.42s
[compute-1-1:21000] >
```

McDonalds:

```
[compute-1-1:21000] > select trim(feature) as features,sum(count) as frequency from fc_mcdonalds where count > 1 group by features order by sum(count) desc limit 20;
Query: select trim(feature) as features,sum(count) as frequency from fc_mcdonalds where count > 1 group by features order by sum(count) desc limit 20
Query submitted at: 2019-12-01 22:12:29 (Coordinator: http://compute-1-1.local:25000)
Query progress can be monitored at: http://compute-1-1.local:25000/query_plan?query_id=6348e5f0c0ea0243:4c3a395700000000
+-------------------------+-----------+
| features                | frequency |
+-------------------------+-----------+
| 'ice cream'             | 74        |
| 'delicious ice'         | 46        |
| 'drive thru'            | 42        |
| 'mcdonalds ceo'         | 30        |
| 'outta hand'            | 25        |
| 'advertising getting'   | 25        |
| 'mcdonalds advertising' | 25        |
| 'getting outta'         | 25        |
| 'lol thats'             | 24        |
| 'thats accurate'        | 24        |
| 'cream machine'         | 19        |
| 'mcdonalds sprite'      | 19        |
| 'mcdonalds money'       | 18        |
| 'become grandparents'   | 18        |
| 'sudden got'            | 18        |
| 'parents become'        | 18        |
| 'ceo fired'             | 18        |
| 'told zeke'             | 17        |
| 'itz_mizdee told'       | 17        |
| 'got food'              | 17        |
+-------------------------+-----------+
Fetched 20 row(s) in 0.42s
```

Analyses 4: Try to compare the average polarity of different restaurants over different platforms like Yelp, Reddit and Twitter and see on which platforms they are more popular/or are more likely to be liked by the people.

Solution:
For Starbucks: We will be running all our queries on IMPALA.

1. Over Reddit, average polarity comes out to be -->

```
[compute-1-1:21000] > select count(polarity) as Count ,avg((polarity*5)+5)/2 as average_polarity  from data_reddit_starbucks;
Query: select count(polarity) as Count ,avg((polarity*5)+5)/2 as average_polarity  from data_reddit_starbucks
Query submitted at: 2019-11-30 21:03:14 (Coordinator: http://compute-1-1.local:25000)
Query progress can be monitored at: http://compute-1-1.local:25000/query_plan?query_id=7d40a77a9317967d:f9ae772b00000000
+-------+------------------+
| count | average_polarity |
+-------+------------------+
| 4988  | 2.618489110807198 |
+-------+------------------+
Fetched 1 row(s) in 0.11s
[compute-1-1:21000] >
```

2. Over Twitter, average polarity comes out to be:

```
[compute-1-1:21000] > select count(polarity) as Count ,avg((polarity*5)+5)/2 as average_polarity  from data_twitter_starbucks;
Query: select count(polarity) as Count ,avg((polarity*5)+5)/2 as average_polarity  from data_twitter_starbucks
Query submitted at: 2019-11-30 21:03:53 (Coordinator: http://compute-1-1.local:25000)
Query progress can be monitored at: http://compute-1-1.local:25000/query_plan?query_id=df4e9584f226ff99:a94956ef00000000
+-------+------------------+
| count | average_polarity  |
+-------+------------------+
| 1531  | 2.6675566508277 16 |
+-------+------------------+
Fetched 1 row(s) in 0.11s
[compute-1-1:21000] >
```

3. Over Yelp, average polarity comes out to be:

```
[compute-1-1:21000] > select count(polarity) as Count ,avg((polarity*5)+5)/2 as average_polarity  from data_yelp where business_name like '%Starbucks%';
Query: select count(polarity) as Count ,avg((polarity*5)+5)/2 as average_polarity  from data_yelp where business_name like '%Starbucks%'
Query submitted at: 2019-11-30 21:05:04 (Coordinator: http://compute-1-1.local:25000)
Query progress can be monitored at: http://compute-1-1.local:25000/query_plan?query_id=3b4edbeb59f5daa2:9e2e73a400000000
+-------+------------------+
| count | average_polarity  |
+-------+------------------+
| 20145 | 2.827585428025279 |
+-------+------------------+
Fetched 1 row(s) in 0.31s
[compute-1-1:21000] >
```

For McDonalds: We will be running all our queries on IMPALA.

1. Over Reddit, average polarity comes out to be:

```
[compute-1-1:21000] >  select count(polarity) as Count,avg((polarity*5)+5)/2 as average_polarity from data_reddit_mcdonalds;
Query: select count(polarity) as Count,avg((polarity*5)+5)/2 as average_polarity from data_reddit_mcdonalds
Query submitted at: 2019-11-30 20:26:20 (Coordinator: http://compute-1-1.local:25000)
Query progress can be monitored at: http://compute-1-1.local:25000/query_plan?query_id=fc498e5ac30fab3a:9258616100000000
+-------+------------------+
| count | average_polarity  |
+-------+------------------+
| 4973  | 2.676237589526467 |
+-------+------------------+
Fetched 1 row(s) in 0.22s
```

2. Over Twitter, average polarity comes out to be:

```
[compute-1-1:21000] >  select count(polarity) as Count,avg((polarity*5)+5)/2 as average_polarity from data_twitter_mcdonalds;
Query: select count(polarity) as Count,avg((polarity*5)+5)/2 as average_polarity from data_twitter_mcdonalds
Query submitted at: 2019-11-30 20:26:28 (Coordinator: http://compute-1-1.local:25000)
Query progress can be monitored at: http://compute-1-1.local:25000/query_plan?query_id=ed42f7eb4884d126:6e66901b00000000
+-------+------------------+
| count | average_polarity  |
+-------+------------------+
| 2588  | 2.582368420240764 |
+-------+------------------+
Fetched 1 row(s) in 0.11s
```

3. Over Yelp, average polarity comes out to be:

```
[compute-1-1:21000] >  select count(polarity) as Count,avg((polarity*5)+5)/2 as average_polarity from data_yelp where business_name like '%McDonalds%';
Query: select count(polarity) as Count,avg((polarity*5)+5)/2 as average_polarity from data_yelp where business_name like '%McDonalds%'
Query submitted at: 2019-11-30 20:28:00 (Coordinator: http://compute-1-1.local:25000)
Query progress can be monitored at: http://compute-1-1.local:25000/query_plan?query_id=7043a5ed9f0031ad:32918a4e00000000
+-------+------------------+
| count | average_polarity  |
+-------+------------------+
| 232   | 2.510612693991349 |
+-------+------------------+
Fetched 1 row(s) in 0.41s
[compute-1-1:21000] >
```

So, we have been able to compute the average polarity of different datasets.

1. If we talk about ***'Starbucks'*** dataset over 3 platforms, we can see that seeing the average polarity, Yelp has the highest average polarity. It also gives us an indication that people are more likely to post a review regarding ***'Starbucks'*** on ***Yelp***.
2. If we talk about ***'McDonalds'*** dataset over 3 platforms, we can see that seeing the average polarity, Reddit has the highest average polarity. It also gives us an indication that people are more likely to post a review regarding ***'McDonalds'*** on ***Reddit***.

NOTE: These results are based on the data which are given, the subject may likely change as data size increases.

Analyses 5: Compare the user ratings and average polarity ratings for different states and compare the results.

Solution: We have compared the average user ratings and average polarity ratings across different states for all the datasets. Let's see one by one

1. **STARBUCKS**

```
[compute-1-1:21000] > Select state, avg(user_rating) as average_user_rating, avg((polarity*5)+5)/2 as average_polarity from data_for_analyses where business_name like "%Starbucks%" and user_rating is not
null group by state;
Query: Select state, avg(user_rating) as average_user_rating, avg((polarity*5)+5)/2 as average_polarity from data_for_analyses where business_name like "%Starbucks%" and user_rating is not null group by
tate
Query submitted at: 2019-12-01 16:34:29 (Coordinator: http://compute-1-1.local:25000)
Query progress can be monitored at: http://compute-1-1.local:25000/query_plan?query_id=2e48f9b992eaa006:a419a67200000000

+-------+---------------------+--------------------+
| state | average_user_rating | average_polarity   |
+-------+---------------------+--------------------+
|  WI   | 3.129139072847682   | 2.845141008987388  |
|  ON   | 3.396859083191851   | 2.933296582941169  |
|  NV   | 2.900223048327137   | 2.768533948253902  |
|  SC   | 3.22680412371134    | 2.807625758284031  |
|  QC   | 3.640350877192982   | 2.943418216886481  |
|  NC   | 3.071868583162217   | 2.817071324967752  |
|  OH   | 3.42914979757085    | 2.877979269768498  |
|  IL   | 3                   | 2.902597257988436  |
|  AZ   | 3.039394384830222   | 2.822624431696321  |
|  PA   | 3.449333333333333   | 2.938863927620308  |
|  AB   | 3.231770833333333   | 2.90656281360062   |
+-------+---------------------+--------------------+
Fetched 11 row(s) in 1.40s
```

The query used is:
Select state, avg(user_rating) as average_user_rating, avg((polarity*5)+5)/2 as average_polarity from data_for_analyses where business_name like "%Starbucks%" and user_rating is not null group by state;

From the above screenshot, we can see that in the state 'QC' average user rating is 3.64 and the average polarity rating is 2.94, which is the highest among all. We can conclude that the STARBUCKS being the most popular brand in state 'QC' among all other states.

```
[compute-1-1:21000] > Select state, avg(user_rating) as average_user_rating, avg((polarity*5)+5)/2 as average_polarity from data_for_analyses where business_name like "%McDonalds%" and user_rating is not
null group by state;
Query: Select state, avg(user_rating) as average_user_rating, avg((polarity*5)+5)/2 as average_polarity from data_for_analyses where business_name like "%McDonalds%" and user_rating is not null group by
tate
Query submitted at: 2019-12-01 16:34:44 (Coordinator: http://compute-1-1.local:25000)
Query progress can be monitored at: http://compute-1-1.local:25000/query_plan?query_id=c949096e4f116ae3:b7b0a40000000000
+-------+---------------------+--------------------+
| state | average_user_rating | average_polarity   |
+-------+---------------------+--------------------+
|  ON   | 2.357142857142857   | 2.545783975995922  |
|  NV   | 2.382352941176471   | 2.748391650384292  |
|  NC   | 1.777777777777778   | 2.595914246824881  |
|  OH   | 1                   | 1.883822185918689  |
|  IL   | 3.181818181818182   | 2.743339343896051  |
|  AZ   | 1.522388059701492   | 2.319336349015131  |
|  AB   | 1.833333333333333   | 2.376954749537011  |
+-------+---------------------+--------------------+
Fetched 7 row(s) in 1.55s
```

The query used is:
Select state, avg(user_rating) as average_user_rating, avg((polarity*5)+5)/2 as average_polarity from data_for_analyses where business_name like "%McDonalds%" and user_rating is not null group by state;

From the above screenshot, we can see that in state ‘IL’ average user rating is 3.18 and the average polarity rating is 2.74, which is the highest among all for Mcdonalds. We can conclude that McDonalds is the most popular brand in state ‘IL’ among all states.

Analyses 6: Compare the average user ratings and average polarity ratings yearly, to see how the restaurant’s performance.

Solution:

Starbucks :
Below is the query and results for average user ratings and average polarity ratings for ‘Starbucks’:
Select year(to_timestamp(trim(date_stamp), 'yyyy-MM-dd')) as period, avg(user_rating) as average_user_rating, avg((polarity*5)+5)/2 as average_polarity from data_for_analyses where business_name like "%Starbucks%" and user_rating is not null group by period order by period;

```
[compute-1-1:21000] > Select year(to_timestamp(trim(date_stamp), 'yyyy-MM-dd')) as period, avg(user_rating) as average_user_rating, avg((polarity*5)+5)/2 as average_polarity from data_for_analyses where
usiness_name like "%Starbucks%" and user_rating is not null group by period order by period;
Query: Select year(to_timestamp(trim(date_stamp), 'yyyy-MM-dd')) as period, avg(user_rating) as average_user_rating, avg((polarity*5)+5)/2 as average_polarity from data_for_analyses where business_name l
ke "%Starbucks%" and user_rating is not null group by period order by period
Query submitted at: 2019-12-01 18:12:22 (Coordinator: http://compute-1-1.local:25000)
Query progress can be monitored at: http://compute-1-1.local:25000/query_plan?query_id=e04d6494caec02c1:64169dc00000000
+--------+--------------------+-------------------+
| period | average_user_rating | average_polarity  |
+--------+--------------------+-------------------+
| 2006   | 2.25               | 2.036258010193706 |
| 2007   | 3.692307692307693  | 2.655653922746961 |
| 2008   | 3.615384615384615  | 2.931280138884456 |
| 2009   | 3.353658536585366  | 2.875694916931424 |
| 2010   | 3.572115384615385  | 2.950665247304781 |
| 2011   | 3.433613445378151  | 2.951887309108078 |
| 2012   | 3.473333333333333  | 2.964648868355774 |
| 2013   | 3.440397350993377  | 2.956344081909349 |
| 2014   | 3.265328874024526  | 2.89068241704015  |
| 2015   | 3.13099556630391   | 2.843670263465883 |
| 2016   | 3.002957607624055  | 2.814642328810233 |
| 2017   | 2.892805126711331  | 2.766470980075643 |
| 2018   | 2.97304806890803   | 2.779246716367921 |
| 2019   | 2.960170697012802  | 2.777272305092497 |
| 2020   | 3.072307692307692  | 2.822158169633342 |
+--------+--------------------+-------------------+
Fetched 15 row(s) in 0.62s
```

Seeing the above results, it’s clear that Starbucks does the fine business between year 2010-2014 and after that there ratings declined between the year 2016-2019.

McDonalds :
Below is the query and results for average user ratings and average polarity ratings for ‘McDonalds’:
Select year(to_timestamp(trim(date_stamp), 'yyyy-MM-dd')) as period, avg(user_rating) as average_user_rating, avg((polarity*5)+5)/2 as average_polarity from data_for_analyses where business_name like "%McDonalds%" and user_rating is not null group by period order by period;

```
[compute-1-1:21000] > Select year(to_timestamp(trim(date_stamp), 'yyyy-MM-dd')) as period, avg(user_rating) as average_user_rating, avg((polarity*5)+5)/2 as average_polarity from data_for_analyses where
usiness_name like "%McDonalds%" and user_rating is not null group by period order by period;
Query: Select year(to_timestamp(trim(date_stamp), 'yyyy-MM-dd')) as period, avg(user_rating) as average_user_rating, avg((polarity*5)+5)/2 as average_polarity from data_for_analyses where business_name
ke "%McDonalds%" and user_rating is not null group by period order by period
Query submitted at: 2019-12-01 18:12:26 (Coordinator: http://compute-1-1.local:25000)
Query progress can be monitored at: http://compute-1-1.local:25000/query_plan?query_id=554c4eabbbdb7c34:34968db00000000
+--------+--------------------+-------------------+
| period | average_user_rating | average_polarity  |
+--------+--------------------+-------------------+
| 2010   | 3.5                | 2.836111118085682 |
| 2011   | 2                  | 2.829422950744629 |
| 2012   | 3                  | 2.622717749327421 |
| 2013   | 2.777777777777778  | 2.465335810557008 |
| 2014   | 1.65               | 2.349281987233553 |
| 2015   | 2.388888888888889  | 2.551084604864526 |
| 2016   | 2.326086956521739  | 2.545754286339101 |
| 2017   | 1.918367346938775  | 2.50890644701977  |
| 2018   | 2.022727272727273  | 2.531607875399376 |
| 2019   | 1.95               | 2.479516484410851 |
| 2020   | 1                  | 2.028724737465382 |
+--------+--------------------+-------------------+
Fetched 11 row(s) in 0.62s
```

Seeing the above results, it can be seen that Mcdonald's average ratings declined between the years 2010-2014, but it shows progress in 2015 and after.

NOTE: All the analytics that we have made is based upon the data which is given. So, the results are subject to change as the data changes.