Twitter Sentiment Analysis Forecasting using Arima and Pyspark

Table of Contents

Table of Figures

Github Account

https://github.com/RehamTousoun/CA2-Term2_Sentiment-Analysis

Introduction

Sentiment analysis is a popular technique used within the field of natural language processing (NLP). Its primary objective is to extract and analyze subjective information present in textual data, assigning it a numerical score ranging from -1 to 1. This technique is often combined with data mining algorithms to predict and classify the sentiment expressed in the text, usually into two or three distinct classes. The applications of sentiment analysis span across numerous domains, encompassing areas such as social media monitoring, customer feedback analysis, market research, stock markets, cryptocurrency analysis (such as bitcoins), brand reputation management, and many other fields.

Data Set Used and Licenses:

The assignment required scraping one year of Twitter data using the Twitter API, A lot of trial and error was performed to scrape Twitter, but without success. The second option was scraping from the archive website, but unfortunately, due to my computer space availability, it wasn't an option at all. I ended up using a dataset from the Kaggle website. The dataset used is Tweets22 downloaded from Kaggle at:

https://www.kaggle.com/datasets/amirhosseinnaghshzan/twitter-2022?resource=download.

Data preparation and Visualization

Exploratory Data Analysis (EDA)

Data analytics has recently gained popularity in both the academic and industrial sectors. The simple meaning of data analysis is the science of extracting proper knowledge (patterns) from analyzing raw data. It is an essential preprocessing step for any data set. EDA requires a further and detailed analysis of the data by understanding the data set, summarizing the main characteristics of the data, and visualizing the features by plotting some histograms, heatmaps, scatter plots, box plots, and many more. EDA has many stages as described below.

1- Read and Load the Tweets.csv file:

```
In [5]:
# Read and Load the Tweets.csv file
df_tweets=pd.read_csv('twitterdataset22.csv',
                      encoding='utf-8' , encoding_errors='ignore')
```

2- Display the five top records of the Tweetes.cvs:

```
df_tweets.head()
```

Out[6]:

	url	date	content	likeCount	replyCount	retweetCount	viewCount	quoteCoun
0	https://twitter.com/LaTera__/status/1477429429...	2022-01-01 23:59:59+00:00	I'm getting a sugar daddy this year!	0	10	0	210.006559	
1	https://twitter.com/allirica_rose/status/14774...	2022-01-01 23:59:59+00:00	might make em french toast after and spoon a l...	1	0	0	191.910011	
2	https://twitter.com/matrixlms/status/147742942...	2022-01-01 23:59:59+00:00	Our platform can be matched to your company's ...	0	0	0	210.006559	
3	https://twitter.com/AnarkyIsMe/status/14774294...	2022-01-01 23:59:59+00:00	Oh my God this first song. I can't with how am...	5	2	0	542.144594	
4	https://twitter.com/__amyya/status/14774294292...	2022-01-01 23:59:59+00:00	Happy new year ! 😁🎉	1	0	0	191.910011	

5 rows × 28 columns

3- Display the last five records:

Out[7]:

	url	date	content	likeCount	replyCount	retweetCount	viewCount
1048570	https://twitter.com/SMOOVmuzik/status/15651272...	2022-08-31 23:59:58+00:00	Can't wait to coach my sons sports team if he ...	3	0	0	377.618452
1048571	https://twitter.com/therealnopixxs/status/1565...	2022-08-31 23:59:58+00:00	i need a fucking snack1	1	2	0	191.910011
1048572	https://twitter.com/goodnessno/status/15651272...	2022-08-31 23:59:58+00:00	Liz is not going quietly! Bet there is much mo...	1	0	0	191.910011
		2022-08-31	Gonna get ridiculously				

4- Display the shape of data:

```
In [8]:
# Display the shape of data(rows and columns)
df_tweets.shape
```

```
Out[8]: (1048575, 28)
```

The data consists of 1048575 rows and 28 columns.

5- Display the dataframe tweets header:

```
df_tweets.columns
```

```
Out[9]: Index(['url', 'date', 'content', 'likeCount', 'replyCount', 'retweetCount',
       'viewCount', 'quoteCount', 'sourceLabel', 'links', 'media',
       'quotedTweet', 'mentionedUsers', 'coordinates', 'place', 'hashtags',
       'cashtags', 'card', 'vibe', 'username', 'UserDescription',
       'UserFavouritesCount', 'followersCount', 'friendsCount', 'location',
       'verified', 'protected', 'mediaCount'],
      dtype='object')
```

6- Filter the dataset from irrelevant columns, only 'Tweets' which store in 'content' column and 'date' is needed for the analysis:

```
In [10]:
df_tweets = df_tweets.iloc[:,1:3]
```

```
In [11]:
df_tweets.head()
```

Out[11]:

	date	content
0	2022-01-01 23:59:59+00:00	I'm getting a sugar daddy this year!
1	2022-01-01 23:59:59+00:00	might make em french toast after and spoon a l...
2	2022-01-01 23:59:59+00:00	Our platform can be matched to your company's ...
3	2022-01-01 23:59:59+00:00	Oh my God this first song. I can't with how am...
4	2022-01-01 23:59:59+00:00	Happy new year ! 😁🍾

7- Rename 'content' column with 'Tweet' and 'date' with 'Date' for clarity:

```
In [12]:
# Rename 'content' column with 'Tweet' and 'date' with 'Date' for clarity
df_tweets = df_tweets.rename({'date':'Date',
                              'content':'Tweet'}, axis=1)
```

8- Display the top 5 records of our dataframe:

```
In [13]:  1  df_tweets.head()
```

Out[13]:

	Date	Tweet
0	2022-01-01 23:59:59+00:00	I'm getting a sugar daddy this year!
1	2022-01-01 23:59:59+00:00	might make em french toast after and spoon a l...
2	2022-01-01 23:59:59+00:00	Our platform can be matched to your company's ...
3	2022-01-01 23:59:59+00:00	Oh my God this first song. I can't with how am...
4	2022-01-01 23:59:59+00:00	Happy new year ! 😁🎉

9- Display the dataframe df_tweets information:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 2 columns):
 #   Column  Non-Null Count    Dtype
---  ------  --------------    -----
 0   Date    1048575 non-null  object
 1   Tweet   1048575 non-null  object
dtypes: object(2)
memory usage: 16.0+ MB
```

The dataframe has two columns 'Date' and 'Tweet' of type object. In the next step the date will convert to datetime type to be useful for the further analysis

```
In [15]:
1 # Reformat the 'timestamp' in the 'Date'column to '%Y-%m-%d %H:%M:%S'format
2 df_tweets['Date'] = pd.to_datetime(df_tweets['Date'], format='%Y-%m-%d %H:%M:%S',errors='coerce')
3 print(df_tweets.dtypes)

Date     datetime64[ns, UTC]
Tweet                 object
dtype: object
```

```
In [16]:
1 # Convert object type of 'date' column to datetime object
2 df_tweets['Date'] =  pd.to_datetime(df_tweets['Date'])
3 # Convert UTC datetime to ns
4 df_tweets['Date'] =  pd.to_datetime(df_tweets['Date']).dt.tz_localize(None)
5 print(df_tweets.dtypes)

Date     datetime64[ns]
Tweet            object
dtype: object
```

10- Checking for null values

```
In [17]:
# Checking for null values
np.sum(df_tweets.isnull().any(axis=1))
```

```
Out[17]: 7
```

The dataset contains 7 null values

To display the records contains nun values:

```
In [18]:
# To display the records contains nun values
df_null_rows = df_tweets[df_tweets.isnull().any(axis=1)]
print(df_null_rows)
```

```
        Date    Tweet
160821   NaT        0
165204   NaT        0
170768   NaT        0
547484   NaT  content
708306   NaT        1
712689   NaT        1
718253   NaT        1
```

Now, the missing values Percentage will be calculated:

```
In [19]:
#Display the Null Percenatage
null_percentage = (len(df_null_rows) / df_tweets.shape[0]) * 100
print(f"The percentage of null values: {null_percentage:.9f}%")
```

```
The percentage of null values: 0.000667573%
```

Given that the dataset has 104,857,8 rows and 7 null values are considered trivial and have no significant impact on model performance, we can drop the null values from the dataset to ensure that the data used for training or analysis is complete and free of missing values.

```
In [20]:
df_tweets = df_tweets.dropna()
```

```
In [21]:
# Checking for null values
np.sum(df_tweets.isnull().any(axis=1))
```

```
Out[21]: 0
```

The dataset has no missing values.

Data Preprocessing and Cleaning

To prepare tweet texts for training a NLP model, it is necessary to clean the data by applying preprocessing and cleansing techniques to ensure that the text data is suitable for training the model. Let's proceed with the data cleaning process. To clean tweet data, you typically follow a series of steps. First, you remove user mentions, hyperlinks, and hashtags as they may not contribute much to the analysis. Then, you eliminate special characters, punctuation, and unnecessary whitespace. Lowercasing the text ensures consistency. Tokenization splits the text into individual words. Removing stopwords helps remove common and insignificant words. Optionally, you can apply stemming or lemmatization to normalize the words. Lastly, you can handle emoticons and abbreviations based on your analysis needs. By performing these steps, you can preprocess tweet data for tasks like sentiment analysis or topic modeling. Adapt the process as per your project requirements and the nature of the tweet data.

1. Convert the tweets to Lowercase & Replace any character which is not an uppercase letter, lowercase letter, or digit:

In [22]:
```
# Put all tweets in lowercase
df_tweets["CleanedTweet"] = df_tweets["Tweet"].str.lower()
# A regular expression [^a-zA-Z0-9] pattern that matches any character that is not an uppercase letter,
# lowercase letter, or digit.
df_tweets["CleanedTweet"]= df_tweets["CleanedTweet"].str.replace("[^a-zA-Z0-9]", " ")
```

In [23]:
```
df_tweets.head()
```

Out[23]:

	Date	Tweet	CleanedTweet
0	2022-01-01 23:59:59	I'm getting a sugar daddy this year!	i m getting a sugar daddy this year
1	2022-01-01 23:59:59	might make em french toast after and spoon a l...	might make em french toast after and spoon a l...
2	2022-01-01 23:59:59	Our platform can be matched to your company's ...	our platform can be matched to your company s ...
3	2022-01-01 23:59:59	Oh my God this first song. I can't with how am...	oh my god this first song i can t with how am...
4	2022-01-01 23:59:59	Happy new year ! 😁🎉	happy new year

2. Display the punctuations list and remove it.

In [24]:
```
# import a string module, to provide a set of common string operations and constants.
import string

# Store the string.punctuation into an object punct
punct = string.punctuation
```

In [25]:
```
# Define the list of English punctuations
punctuations_list = list(punct)
print(punctuations_list)
```

```
['!', '"', '#', '$', '%', '&', "'", '(', ')', '*', '+', ',', '-', '.', '/', ':', ';', '<', '=', '>', '?',
'@', '[', '\\', ']', '^', '_', '`', '{', '|', '}', '~']
```

In [26]:
```
# Function to clean punctuations from text
def cleaning_punctuations(text):
    translator = str.maketrans('', '', ''.join(punctuations_list))
    return text.translate(translator)

# Apply the cleaning_punctuations() function to the 'text' column in the dataset
df_tweets['CleanedTweet'] = df_tweets['CleanedTweet'].apply(lambda x: cleaning_punctuations(x))
```

Display the updated dataframe after removing punctuations:

Out[27]:

	Date	Tweet	CleanedTweet
0	2022-01-01 23:59:59	I'm getting a sugar daddy this year!	i m getting a sugar daddy this year
1	2022-01-01 23:59:59	might make em french toast after and spoon a l...	might make em french toast after and spoon a l...
2	2022-01-01 23:59:59	Our platform can be matched to your company's ...	our platform can be matched to your company s ...
3	2022-01-01 23:59:59	Oh my God this first song. I can't with how am...	oh my god this first song i can t with how am...
4	2022-01-01 23:59:59	Happy new year ! 😁🎉	happy new year

3- Cleaning Numeric characters:

In [28]:
```
# clean numeric characters from the 'Cleanedtweet' column
def cleaning_numeric(text):
    return re.sub('[0-9]+','',text)
```

In [29]:
```
# Apply the cleaning_numeric() function to the 'CleanedTweet' column in the dataset
df_tweets['CleanedTweet'] = df_tweets['CleanedTweet'].apply(lambda x: cleaning_numeric(x))
df_tweets.head()

```

Out[29]:

	Date	Tweet	CleanedTweet
0	2022-01-01 23:59:59	I'm getting a sugar daddy this year!	i m getting a sugar daddy this year
1	2022-01-01 23:59:59	might make em french toast after and spoon a l...	might make em french toast after and spoon a l...
2	2022-01-01 23:59:59	Our platform can be matched to your company's ...	our platform can be matched to your company s ...
3	2022-01-01 23:59:59	Oh my God this first song. I can't with how am...	oh my god this first song i can t with how am...
4	2022-01-01 23:59:59	Happy new year ! 😁🎉	happy new year

4- Removing the URLs:

Define the 'cleaning_URLs'that clean the tweet from any URLs:

In [30]:
```
# clean URLs from the 'Cleanedtweet' column
def cleaning_URLs (text):
    return re.sub('((www.[^s]+)|(https?://[^s]+))',' ', text)
```

Update the tweets in ''CleanedTweet' column by applying the function to it:

In [31]:
```
# Apply the cleaning_repeating_char() function to the 'CleanedTweet' column in the dataset
df_tweets['CleanedTweet'] = df_tweets['CleanedTweet'].apply(lambda x: cleaning_URLs(x))
df_tweets.head()
```

Out[31]:

	Date	Tweet	CleanedTweet
0	2022-01-01 23:59:59	I'm getting a sugar daddy this year!	i m getting a sugar daddy this year
1	2022-01-01 23:59:59	might make em french toast after and spoon a l...	might make em french toast after and spoon a l...
2	2022-01-01 23:59:59	Our platform can be matched to your company's ...	our platform can be matched to your company s ...
3	2022-01-01 23:59:59	Oh my God this first song. I can't with how am...	oh my god this first song i can t with how am...
4	2022-01-01 23:59:59	Happy new year ! 😁🎉	happy new year

5- Remove Stop words:

StopWords: Stopwords are words in English that are frequently used but typically do not contribute much to the overall meaning of a sentence. These words are often removed before classification or analysis tasks. Some examples of stopwords include "a", "an", "the", "in", "on", "is", "are", and so on. Removing stopwords helps reduce noise and improve the accuracy of text analysis and classification models.

Display the list of all words stored in stopwords:

In [33]:
```
# Store the stopwords into the object named as "stop_words"
stop_words = stopwords.words('english')

# Diplay the list of all words stored in stopwords
print(stop_words)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd",
'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'he
rself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'wh
o', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'be
ing', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'o
r', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'int
o', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on',
'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how',
'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'o
wn', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "shoul
d've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn',
"didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma',
'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'was
n', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

Removing the stop words listed above from the ‘CleanedTweet’ column:

In [34]:
```
# Removing the stop words listed above from the 'Tweet' column
def cleaning_stopwords(text):
    return " ".join([word for word in str(text).split() if word not in stop_words])
```

In [35]:
```
# Apply the function to the 'Tweet' column in the dataframe
df_tweets['CleanedTweet'] = df_tweets['CleanedTweet'].apply(lambda text: cleaning_stopwords(text))
```

Let’s have a look at the updated dataframe df_tweets after removing the stop words:

```
In [36]:  1 df_tweets.head(10)
```

Out[36]:

	Date	Tweet	CleanedTweet
0	2022-01-01 23:59:59	I'm getting a sugar daddy this year!	getting sugar daddy year
1	2022-01-01 23:59:59	might make em french toast after and spoon a l...	might make em french toast spoon little
2	2022-01-01 23:59:59	Our platform can be matched to your company's ...	platform matched company brand adding personal...
3	2022-01-01 23:59:59	Oh my God this first song. I can't with how am...	oh god first song amazing chick live hitting n...
4	2022-01-01 23:59:59	Happy new year ! 😊🎉	happy new year
5	2022-01-01 23:59:59	Seeing John Oliver at the Kennedy center! http...	seeing john oliver kennedy center https co fdw...
6	2022-01-01 23:59:59	Everyone should go watch Nuclear Family on HBO...	everyone go watch nuclear family hbo amazing a...
7	2022-01-01 23:59:59	Real pero real real! 🤣🤣🤣🤣	real pero real real
8	2022-01-01 23:59:59	Dear diary,\n\nShe tried to steal my notebook,...	dear diary tried steal notebook put name give ...
9	2022-01-01 23:59:59	Last year was wild. But damn, I understood mys...	last year wild damn understood became spiritua...

6- Stemming and lemmatization

are techniques used to reduce words to their base or root forms in order to handle different variations of a word and achieve a common representation. Here's a summary of both techniques: Stemming: Stemming involves removing prefixes, suffixes, and other word endings from a word to obtain its core form. The goal is to approximate the base form of a word, even if the resulting stem is not always a valid word itself. Stemming is a heuristic process and can sometimes lead to inaccuracies or produce stems that are not actual words. Lemmatization: Lemmatization, on the other hand, aims to determine the base or dictionary form (known as the lemma) of a word using vocabulary and morphological analysis. It considers the part of speech of the word and applies linguistic rules to derive the lemma while preserving the word's meaning. Lemmatization typically produces valid words as lemmas and is a more precise process compared to stemming. Stemming and lemmatization are optional to use. There are various stemmers we can use, such as PorterStemmer, SnowballStemmer, LancasterStemmer etc. SnowballStemmer is the most common.

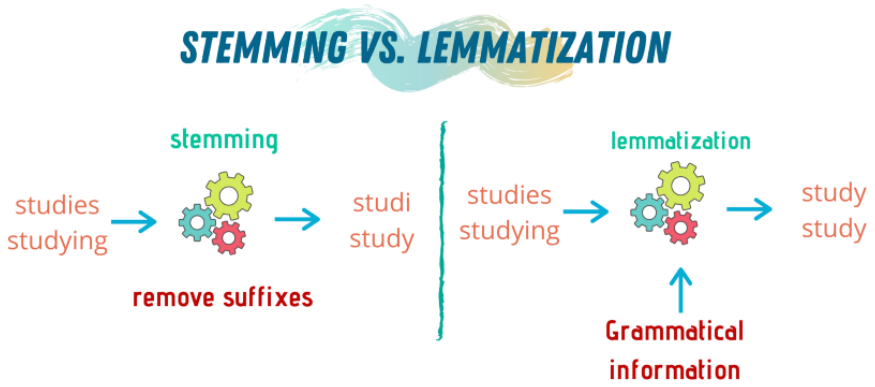


Figure 1: Stemming Vs. lemmatization.

Two functions ‘stemming_on_text’ and ‘lemmatizer_on_text’ were defined and applied to the ‘CleanedTweet’ column. Let’s display the top five records from the updated dataframe:

Out[45]:

	Date	Tweet	CleanedTweet
0	2022-01-01 23:59:59	I’m getting a sugar daddy this year!	getting sugar daddy year
1	2022-01-01 23:59:59	might make em french toast after and spoon a l...	might make em french toast spoon little
2	2022-01-01 23:59:59	Our platform can be matched to your company’s ...	platform matched company brand adding personal...
3	2022-01-01 23:59:59	Oh my God this first song. I can't with how am...	oh god first song amazing chick live hitting n...
4	2022-01-01 23:59:59	Happy new year ! 😁🎉	happy new year

7- Tokenization

Tokenization: is the process of breaking a sequence of characters or words into smaller units, known as tokens. It involves dividing the text into meaningful chunks while potentially removing certain characters like punctuation. This task is commonly referred to as tokenization and serves as a fundamental step in text processing and analysis. A tokenizer is used to create tokens for each word in a given dataset corpus and map them to corresponding indices using a dictionary. The dictionary, typically referred to as word_index, contains the index value for each unique word in the corpus. On the other hand, vocab_size represents the total count of unique words present in the corpus, indicating the size of the vocabulary. In summary, the tokenizer assigns numerical indices to words in the corpus, allowing for easier representation and processing of text data.

A function ‘tokenize’ was defined as the following:

In [46]:

```
# Define the tokenizer function
def tokenize(text):
    tokens = re.split("\W+", text)
    return tokens
```

A new column ‘TweetToken’ was created in our dataframe

Out[47]:

	Date	Tweet	CleanedTweet	TweetToken
0	2022-01-01 23:59:59	I’m getting a sugar daddy this year!	getting sugar daddy year	[getting, sugar, daddy, year]
1	2022-01-01 23:59:59	might make em french toast after and spoon a l...	might make em french toast spoon little	[might, make, em, french, toast, spoon, little]
2	2022-01-01 23:59:59	Our platform can be matched to your company’s ...	platform matched company brand adding personal...	[platform, matched, company, brand, adding, pe...
3	2022-01-01 23:59:59	Oh my God this first song. I can't with how am...	oh god first song amazing chick live hitting n...	[oh, god, first, song, amazing, chick, live, h...
4	2022-01-01 23:59:59	Happy new year ! 😁🎉	happy new year	[happy, new, year]

Now, we have cleaned all tweets. It's time to calculate the tweet sentiment by using the TextBlob library to compute the polarity and subjectivity for the Tweets. Let's first define what polarity and subjectivity means?

Calculate polarity and subjectivity for the Tweets

To obtain the polarity and subjectivity scores for all tweets, the need to import the TextBlob library is required in order to leverage its sentiment analysis capabilities. The sentiment or emotional tone of the text is indicated by the sentiment score known as polarity. This score range between -1 to 1, where -1 representing a strongly negative attitude, 0 representing neutral sentiment, and 1 representing a strongly positive mood. The subjectivity score ranges from 0 to 1, with 0 being an objective or factual statement and 1 denoting a subjective or opinionated remark. It assesses the degree of subjectivity in the text.

Two functions 'determineTextSubjectivity' and 'getTextPolarity' were created to get the tweets subjectivity and polarity.

In [50]:
```
# Get the tweets subjectivity
def determineTextSubjectivity(text):
    return TextBlob(text).sentiment.subjectivity
```

In [51]:
```
# Get the tweets polarity
def getTextPolarity(txt):
    return TextBlob(txt).sentiment.polarity
```

Also, two new columns for subjectivity and polarity were added to 'df_tweets' dataframe.

In [52]:
```
# Add two columns for subjectivity and polarity
df_tweets['Subjectivity']= df_tweets['CleanedTweet'].apply(determineTextSubjectivity)
df_tweets['Polarity']= df_tweets['CleanedTweet'].apply(getTextPolarity)
```

T have a deep look to the calculated subjectivity and polarity values, the top 15 record from the dataframe were displayed:

In [53]:
```
# Display the top 15 record from the dataframe
df_tweets.head(15)
```

Out[53]:

	Date	Tweet	CleanedTweet	TweetToken	Subjectivity	Polarity
0	2022-01-01 23:59:59	I'm getting a sugar daddy this year!	getting sugar daddy year	[getting, sugar, daddy, year]	0.000000	0.000000
1	2022-01-01 23:59:59	might make em french toast after and spoon a l...	might make em french toast spoon little	[might, make, em, french, toast, spoon, little]	0.250000	-0.093750
2	2022-01-01 23:59:59	Our platform can be matched to your company's ...	platform matched company brand adding personal...	[platform, matched, company, brand, adding, pe...	0.000000	0.000000
3	2022-01-01 23:59:59	Oh my God this first song. I can't with how am...	oh god first song amazing chick live hitting n...	[oh, god, first, song, amazing, chick, live, h...	0.577778	0.328788
4	2022-01-01 23:59:59	Happy new year ! 😁🎉	happy new year	[happy, new, year]	0.727273	0.468182
5	2022-01-01 23:59:59	Seeing John Oliver at the Kennedy center! http...	seeing john oliver kennedy center https co fdw...	[seeing, john, oliver, kennedy, center, https,...	0.100000	-0.100000
6	2022-01-01 23:59:59	Everyone should go watch Nuclear Family on HBO...	everyone go watch nuclear family hbo amazing a...	[everyone, go, watch, nuclear, family, hbo, am...	0.600000	0.400000
7	2022-01-01 23:59:59	Real pero real real! 🤣🤣🤣🤣	real pero real real	[real, pero, real, real]	0.375000	0.250000
8	2022-01-01 23:59:59	Dear diary,\n\nShe tried to steal my notebook,...	dear diary tried steal notebook put name give ...	[dear, diary, tried, steal, notebook, put, nam...	0.000000	0.000000
9	2022-01-01 23:59:59	Last year was wild. But damn, I understood mys...	last year wild damn understood became spiritua...	[last, year, wild, damn, understood, became, s...	0.273160	0.032468
10	2022-01-01 23:59:59	"Never limit yourself because of others' limit...	never limit others limited imagination never l...	[never, limit, others, limited, imagination, n...	0.142857	-0.071429
11	2022-01-01 23:59:59	quack https://t.co/M8iVui1wDf	quack https co mivuiwdf	[quack, https, co, mivuiwdf]	0.000000	0.000000
12	2022-01-01 23:59:59	cry as much as you can, its a competition, and...	cry much competition step im gonna win	[cry, much, competition, step, im, gonna, win]	0.300000	0.500000
13	2022-01-01 23:59:59	Go to the barbershop and cut my name in you ha...	go barbershop cut name hair kinda love	[go, barbershop, cut, name, hair, kinda, love]	0.600000	0.500000
14	2022-01-01 23:59:59	i won jackbox the other day https://t.co/4y4jK...	jackbox day https co yjksqej	[jackbox, day, https, co, yjksqej]	0.000000	0.000000

It's clear that the values of polarity are between -1 and 1, while for subjectivity, it ranges from 0:1.

Now, a new colum 'Sentiment' was added to the dataframe to describe the tweets polarity whether its negative, neutral or positive.

Out[56]:

	Date	Tweet	CleanedTweet	TweetToken	Subjectivity	Polarity	Sentiment
0	2022-01-01 23:59:59	I'm getting a sugar daddy this year!	getting sugar daddy year	[getting, sugar, daddy, year]	0.000000	0.000000	Neutral
1	2022-01-01 23:59:59	might make em french toast after and spoon a l...	might make em french toast spoon little	[might, make, em, french, toast, spoon, little]	0.250000	-0.093750	Negative
2	2022-01-01 23:59:59	Our platform can be matched to your company's ...	platform matched company brand adding personal...	[platform, matched, company, brand, adding, pe...	0.000000	0.000000	Neutral
3	2022-01-01 23:59:59	Oh my God this first song. I can't with how am...	oh god first song amazing chick live hitting n...	[oh, god, first, song, amazing, chick, live, h...	0.577778	0.328788	Positive
4	2022-01-01 23:59:59	Happy new year ! 😁🎉	happy new year	[happy, new, year]	0.727273	0.468182	Positive
5	2022-01-01 23:59:59	Seeing John Oliver at the Kennedy center! http...	seeing john oliver kennedy center https co fdw...	[seeing, john, oliver, kennedy, center, https,...	0.100000	-0.100000	Negative
	2022-01-01	Everyone should go watch	everyone go watch nuclear	[everyone, go, watch,			

Display the tweets sentiments pie chart and bar plot:

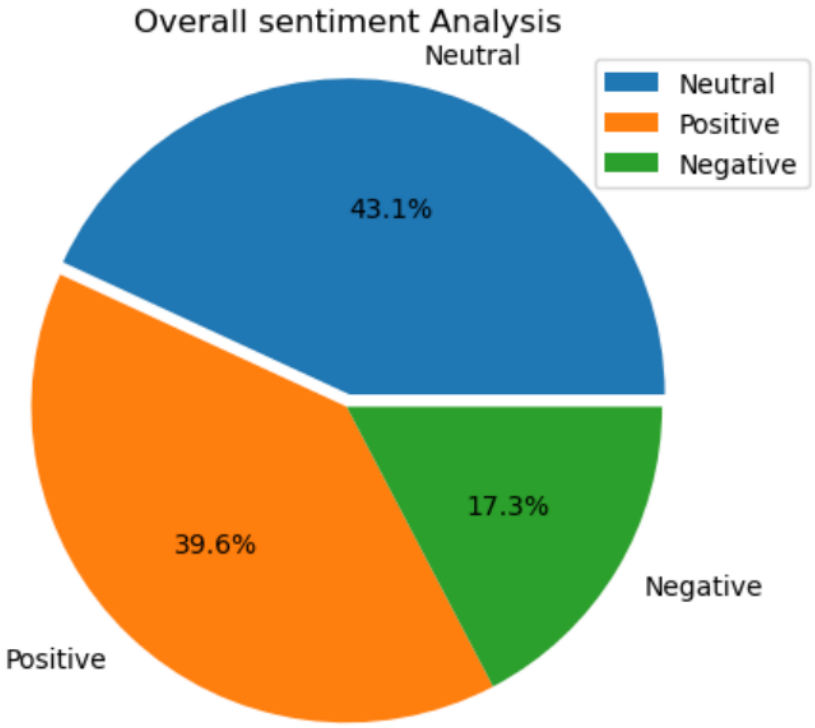


Figure 2: Overall sentiment Analysis

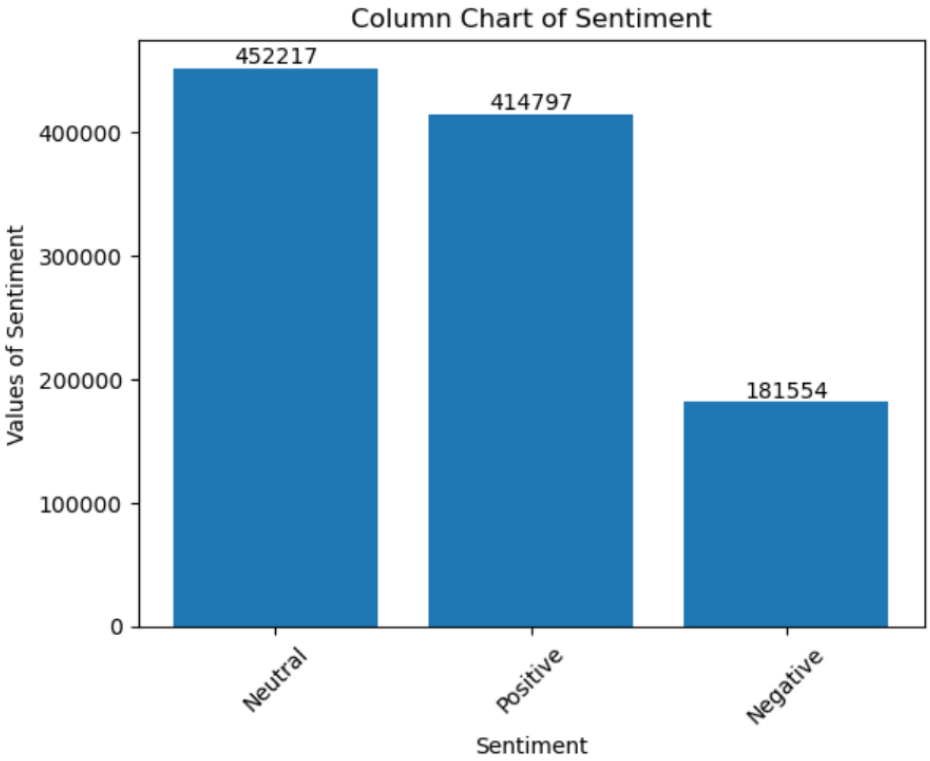


Figure 3: Bar Plot of the Sentiment.

The remaining sections on the 'Sba22483_SentimentAnalysis. .ipynb' are:

1- Selecting data sample from the original dataset, calculated the Bag of Words using CountVectorizer. In addition, the tweet length and word count were calculated, and the pair plot for the sampled data displayed to study the correlations between variables (Not mentioned here in details because never used this sample in the sentiment forecasting).

2- Multiclass Logistic Regression was applied to this balanced sample to classify the tweet sentiment into three classes: positive, negative, and neutral. The model trained with

Autoregressive Integrated Moving Average (ARIMA)

ARIMA is a commonly used time series forecasting statistical model that incorporates autoregression, differencing, and moving average components (Hyndman and Athanasopoulos, 2018). It is denoted as ARIMA (p, d, q), where p represents the order of the autoregressive component, d represents the degree of differencing, and q represents the order of the moving average component. To utilize ARIMA effectively, the following steps are typically followed (Hyndman and Athanasopoulos, 2018):

1. Visualize the time series data to understand its patterns and trends:

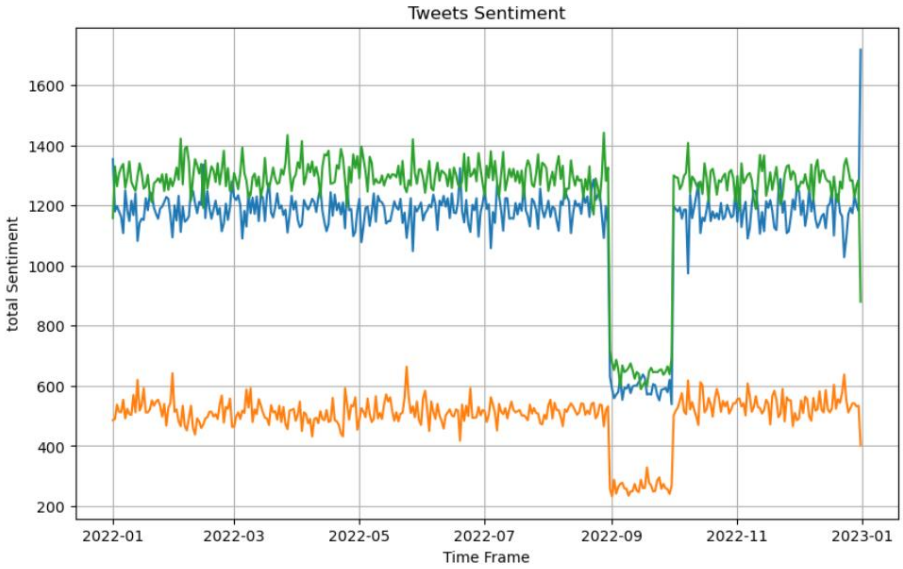


Figure 4: Time Series graph for tweets'sentiment over 2022

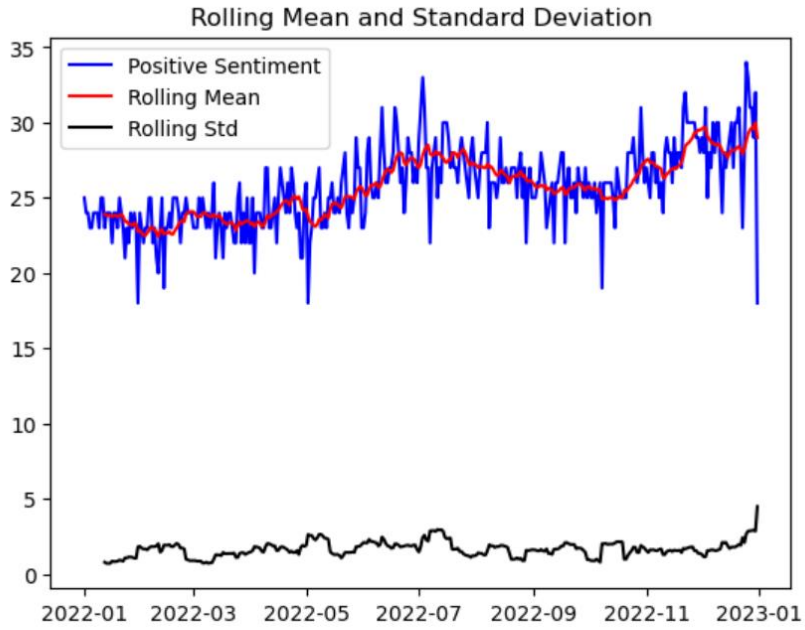
Figure 4 shows the daily tweets counts during the period from January to December 2022. The highest number of positive tweets, totaling 1718, was posted on 31st December. However, between 28th August and 4th October, the count of tweets, regardless of their sentiment (positive, neutral, or negative), was at its lowest.

2. Check for stationarity and apply differencing if necessary to achieve stationarity.:

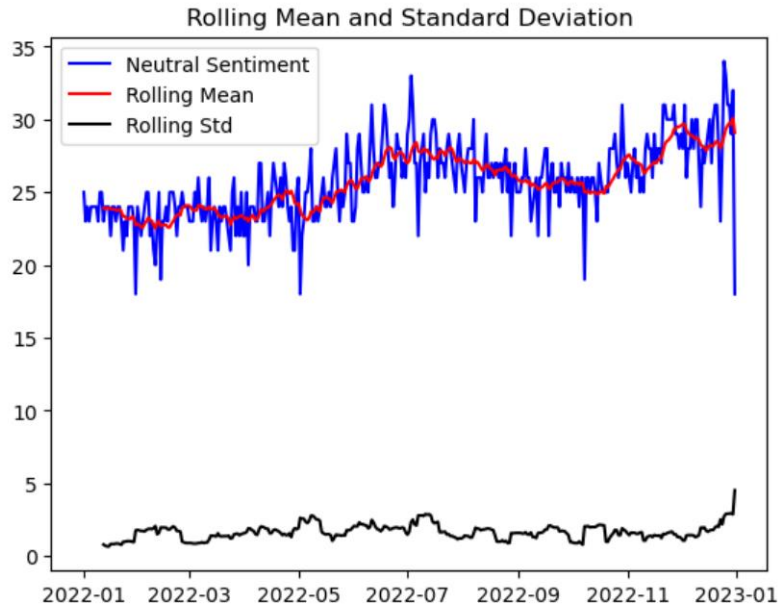
Stationarity refers to a series where statistical properties like mean, variance, and autocorrelation remain constant throughout time. Assessing the stationarity of a time series is a crucial step before conducting time series analysis. To determine stationarity, level, trend, seasonality, and noise components of the series are examined. **The Augmented Dickey-Fuller (ADF)** test was applied to examine if a time series has a unit root, which indicates non-stationarity. The series exhibits a constant mean and variance over time.

H0: The series has a unit root, meaning it is non-stationary.

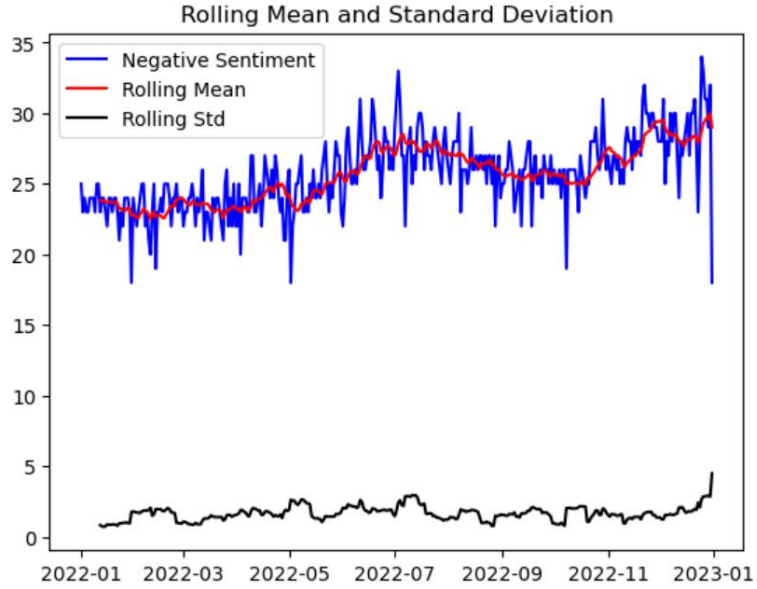
H1: The series does not have a unit root, indicating stationarity.




```
Results of dickey fuller test
Test Statistics                  -1.082929
p-value                           0.721831
No. of lags used                 13.000000
Number of observations used     351.000000
critical value (1%)              -3.449119
critical value (5%)              -2.869810
critical value (10%)             -2.571176
dtype: float64
```




```
Results of dickey fuller test
Test Statistics                  -1.040362
p-value                           0.738260
No. of lags used                 13.000000
Number of observations used     351.000000
critical value (1%)              -3.449119
critical value (5%)              -2.869810
critical value (10%)             -2.571176
dtype: float64
```




```
Results of dickey fuller test
Test Statistics                  -1.028334
p-value                           0.742790
No. of lags used                 13.000000
Number of observations used     351.000000
critical value (1%)              -3.449119
critical value (5%)              -2.869810
critical value (10%)             -2.571176
dtype: float64
```

For positive, neutral, and negative sentiment the value of p-value is 0.721, .742, 0.738 which is > 0.05 ($p > \alpha$) so we don't reject the null hypothesis. It is evidence to tell that all the sentiment is nonlinear, which means it is non-stationary.

To be more focused, the forecasting of positive sentiment will be performed in the next steps:
By removing the trend component, we eliminate the long-term systematic increase or decrease in the series. Similarly, by separating the seasonality components, we extract the recurring patterns that occur within shorter time intervals, such as daily, weekly, or yearly cycles. This enables us to study the specific seasonal effects and their impact on the data.

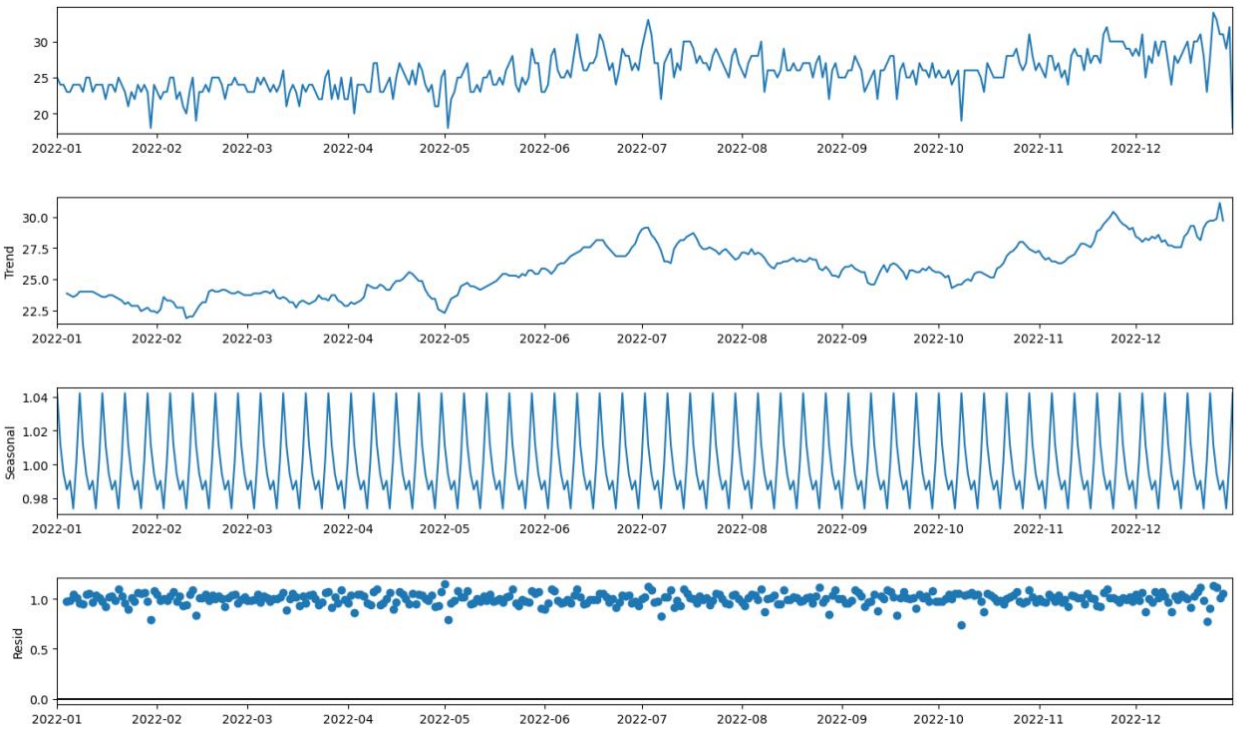


Figure 5: Trend and seasonality of the positive tweets

To reduce the scale of values and the increasing trend in the series, we employ a two-step approach:

- Apply a logarithmic transformation to the series.
- Compute the rolling average of the log-transformed series by taking the average consumption value over a window of the preceding 12 months at each subsequent data point

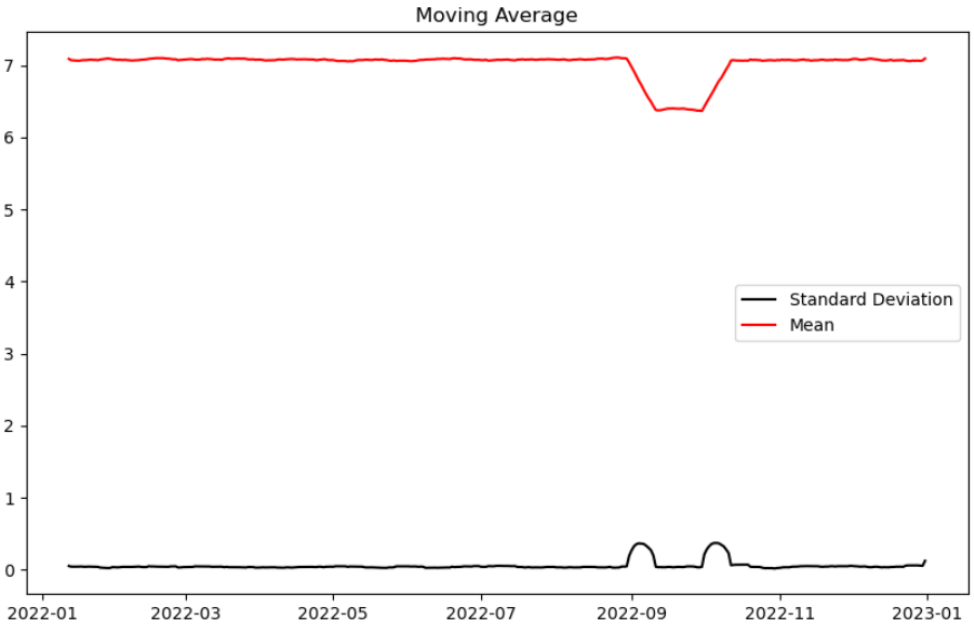


Figure 6: Positive tweets trend

Now, we are ready to build an ARIMA model and train it uses the daily count of the positive sentiment of tweets from the training data. To gain insights into the data, we will visualize it by partitioning it into separate training and testing sets.

3. Splitting the dataset into training and testing:

   The training set, X_train containing 90% of the data, while a testing set, X_test containing the remaining 10% of the data.

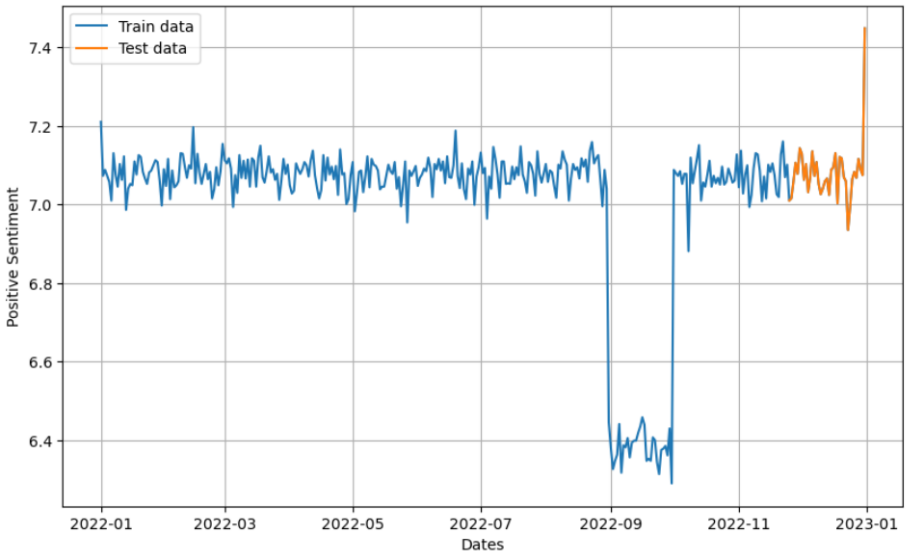


Figure 7: DataSet Splitting into training and testing.

4. Determine the p, q, and d parameters for the ARIMA model, we will use Auto ARIMA.

```
Best model:  ARIMA(1,1,2)(0,0,0)[0]
Total fit time: 4.108 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  325
Model:               SARIMAX(1, 1, 2)   Log Likelihood                 379.380
Date:                Sun, 11 Jun 2023   AIC                           -750.761
Time:                        22:53:44   BIC                           -735.638
Sample:                    01-04-2022   HQIC                          -744.724
                         - 11-24-2022
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.9585      0.077    -12.477      0.000      -1.109      -0.808
ma.L1          0.6851      0.085      8.078      0.000       0.519       0.851
ma.L2         -0.2152      0.070     -3.068      0.002      -0.353      -0.078
sigma2         0.0056      0.000     53.640      0.000       0.005       0.006
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):             25177.71
Prob(Q):                              0.91   Prob(JB):                         0.00
Heteroskedasticity (H):               6.09   Skew:                             1.44
Prob(H) (two-sided):                  0.00   Kurtosis:                        46.09
===================================================================================
```

Figure 8: Auto ARIMA model to determine p, q and d parameters.

The Auto ARIMA model determined the values of 1, 1, and 2 for the parameters p, d, and q, respectively.

5. Fit the ARIMA model to the data.

```
                               SARIMAX Results
==============================================================================
Dep. Variable:              Sentiment   No. Observations:                  325
Model:               ARIMA(1, 1, 2)   Log Likelihood                 379.380
Date:                Sun, 11 Jun 2023   AIC                           -750.761
Time:                        22:53:45   BIC                           -735.638
Sample:                    01-04-2022   HQIC                          -744.724
                         - 11-24-2022
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.9585      0.077    -12.477      0.000      -1.109      -0.808
ma.L1          0.6851      0.085      8.078      0.000       0.519       0.851
ma.L2         -0.2152      0.070     -3.068      0.002      -0.353      -0.078
sigma2         0.0056      0.000     53.640      0.000       0.005       0.006
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):             25177.71
Prob(Q):                              0.91   Prob(JB):                         0.00
Heteroskedasticity (H):               6.09   Skew:                             1.44
Prob(H) (two-sided):                  0.00   Kurtosis:                        46.09
===================================================================================
```

Figure 9: ARIMA model

6. Evaluate the model using diagnostic plots and performance metrics.

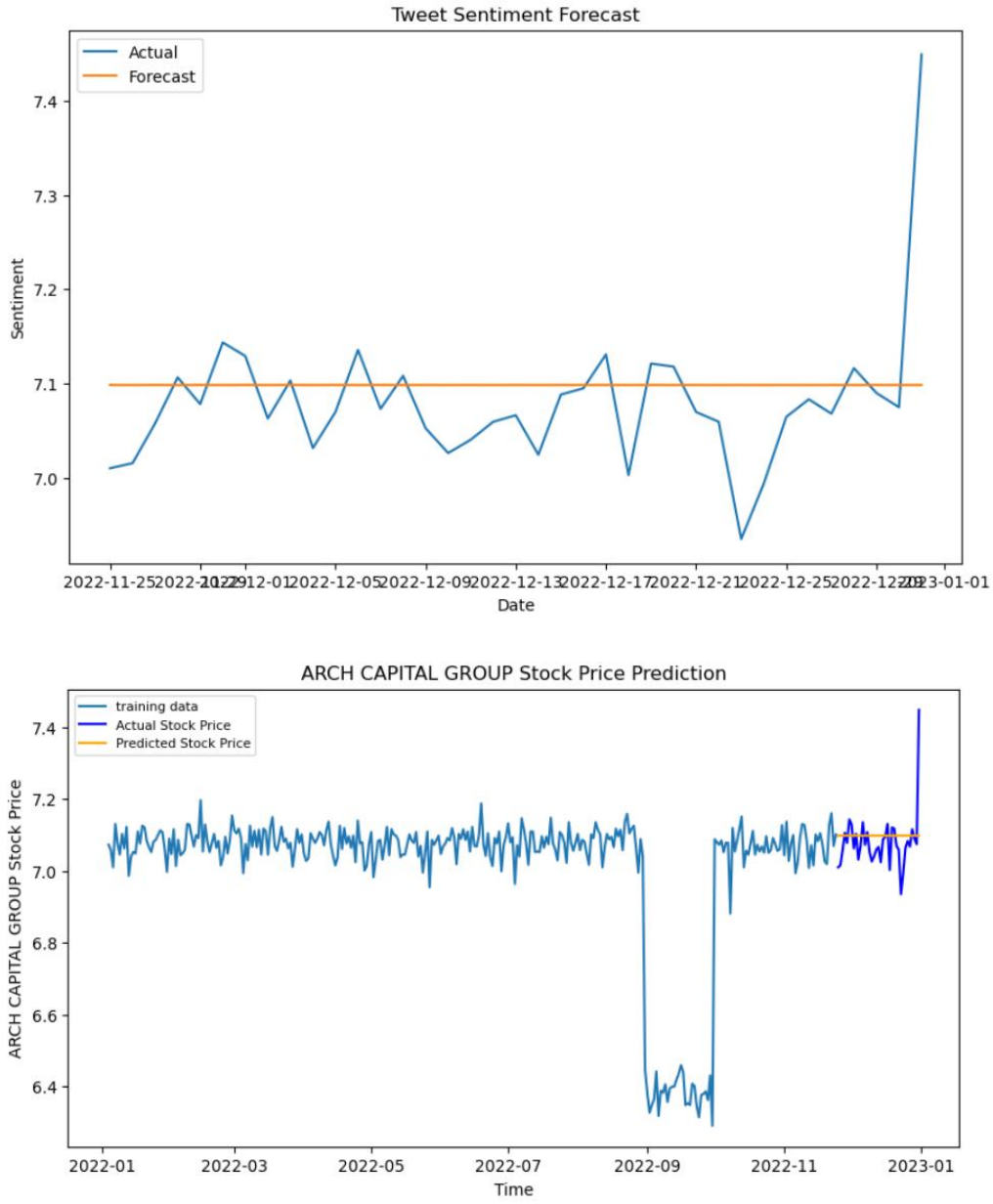


Figure 10: Positive tweets Sentiment Forecast

The ARIMA forecast appears as a straight line, it means that the model is not capturing the underlying patterns in the data effectively. This could be the unexpected changes in the data between 28th Aug – 4th OCT. Also, it may be due to the inadequate complexity of the model.

An Interactive Dashboard with Panel:

What is Panel? Panel is an open-source python library that can be used to create interactive web apps and dashboards (*Overview — Panel v1.1.0*).

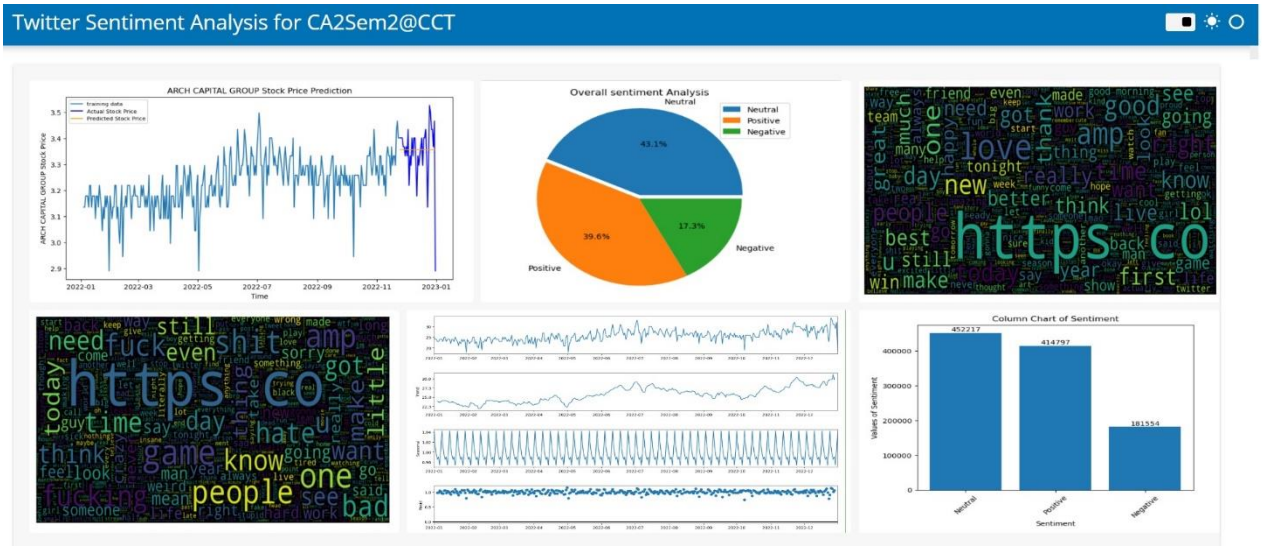


Figure 11: Twitter Sentiment Analysis Dashboard

Big Data

Abstract:

Nowadays, social media platforms like Twitter generate large datasets that are valuable for analysis. Among these platforms, Twitter's data is considered reliable due to its privacy policy. Tweets have been extensively used for sentiment analysis and extracting meaningful information. In this task the dataset sentiment 140 used. this data set consists of 1.6million tweets about the human feeling. The proposed model in this assignment utilizes Apache Spark, a powerful big data processing framework, to handle the large Twitter dataset, many tools from NLP technique is applied for data preprocessing, involving the removal of retweets, links, hashtags, English words and numbers, usernames, and emojis from the dataset. Following preprocessing, a Logistic Regression model, a popular machine learning algorithm, is trained on the preprocessed data. This model aims to predict the sentiment expressed in the tweets. The evaluation of the model on test data showcases impressive results with an accuracy score of 73%, F1 score of 73%, precision of 73%, and recall of 73%. In summary, this study presents a model for analyzing Twitter dataset. Apache Spark is employed to handle the large dataset, and a Logistic Regression model is trained for sentiment analysis of tweets.

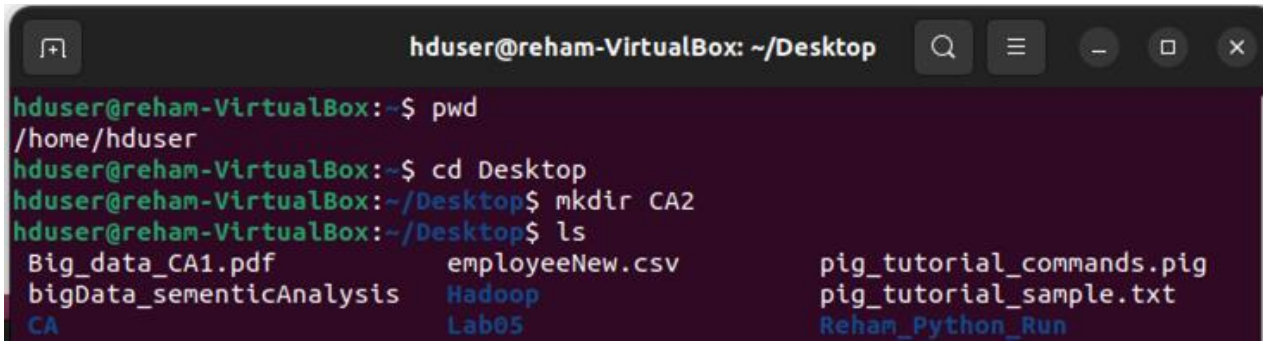
Practical Session:

The practical session includes two stages:

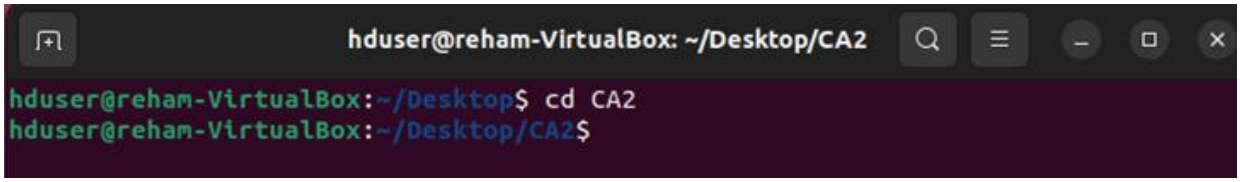
1. Using Hadoop and Pyspark to create a Jupyter Notebook in the Ubuntu operating system.
2. Use a ‘Yahoo! Cloud Serving Benchmark" (YCSB to compare the performance of SQL and NoSQL databases.

First: Using Hadoop and Pyspark to create a Jupyter Notebook

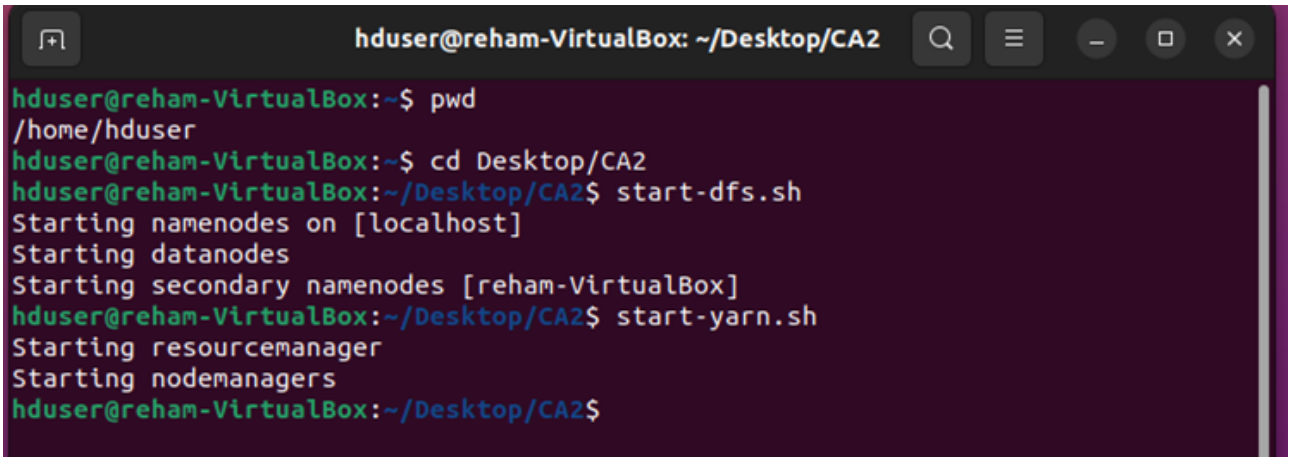
1- Pree ctrl + Alt + t to open a terminal in Ubuntu, print the current working directory using **pwd** command, a new directory was created on the Desktop with the name ‘CA2’ and it contains the downloaded dataset and others files.

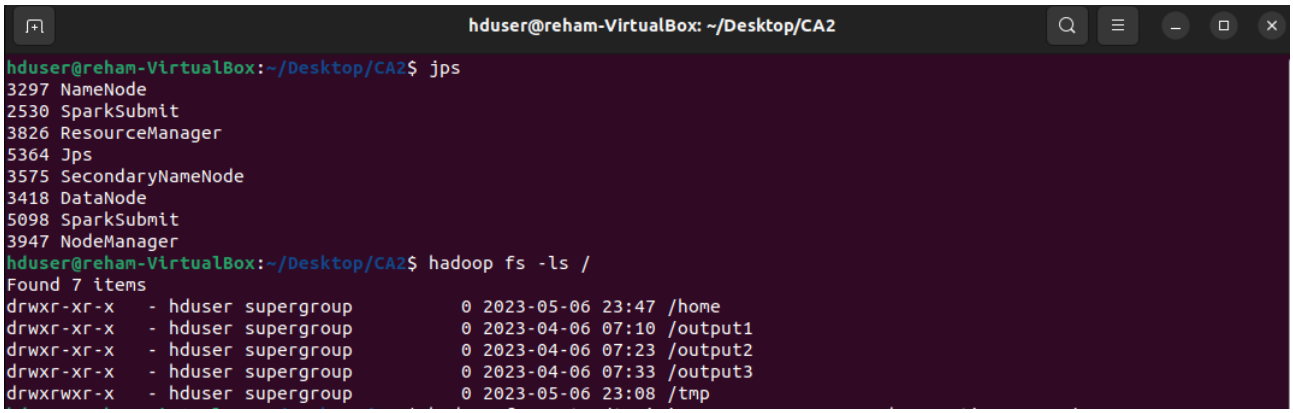


2- move to the new directory with cd command.



3- Start Hadoop and yarn.



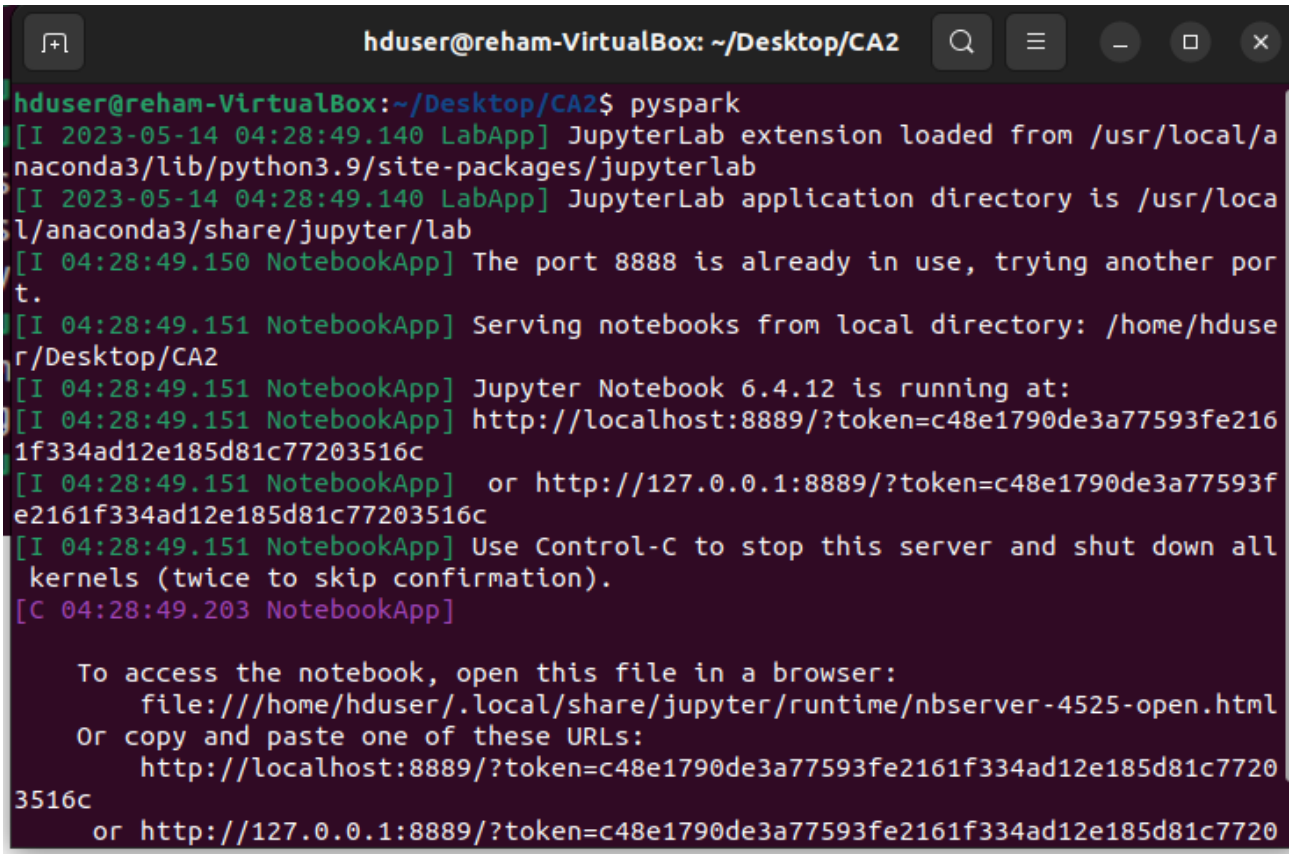


4- Load the datasets ‘training.1600000.processed.noemoticon.csv’ and ‘tweetsCA2.csv’directory into Hadoop Distributed File System (HDFS). Put in the directory ‘user1.

```
hduser@reham-VirtualBox:~/Desktop/CA2$ hadoop fs -put ./training.1600000.processed.noemoticon.csv /user1
hduser@reham-VirtualBox:~/Desktop/CA2$ hadoop fs -ls /user1
Found 16 items
-rw-r--r--   1 hduser supergroup        395 2023-04-06 13:42 /user1/HeightAndWeight_.csv
-rw-r--r--   1 hduser supergroup        948 2023-04-06 07:37 /user1/ProblemStatement1.txt
-rw-r--r--   1 hduser supergroup      11411 2023-04-13 22:20 /user1/airport-codes-na.txt
-rw-r--r--   1 hduser supergroup      19362 2023-04-06 07:07 /user1/auto-mpg-data-original.txt
-rw-r--r--   1 hduser supergroup     135287 2023-04-06 07:07 /user1/britney-spears.txt
-rw-r--r--   1 hduser supergroup   81372510 2023-04-13 22:26 /user1/ccFraud.csv.gz
drwxr-xr-x   - hduser supergroup          0 2023-05-06 23:51 /user1/data
-rw-r--r--   1 hduser supergroup   33396236 2023-04-13 22:20 /user1/departuredelays.csv
-rw-r--r--   1 hduser supergroup         56 2023-05-07 14:13 /user1/employeeNew.csv
-rw-r--r--   1 hduser supergroup         73 2023-04-13 22:19 /user1/people.json
-rw-r--r--   1 hduser supergroup      65167 2023-04-06 13:36 /user1/pg30123.txt
-rw-r--r--   1 hduser supergroup        419 2023-05-05 11:42 /user1/pig_script.pig
-rw-r--r--   1 hduser supergroup  211312924 2023-04-06 07:37 /user1/purchases.txt
-rw-r--r--   1 hduser supergroup        529 2023-04-06 07:07 /user1/sample.txt
drwxr-xr-x   - hduser supergroup          0 2023-05-05 11:43 /user1/student_output
-rw-r--r--   1 hduser supergroup  238803811 2023-05-14 04:41 /user1/training.1600000.processed.noemoticon.csv
hduser@reham-VirtualBox:~/Desktop/CA2$
```

```
-rw-r--r-- 1 hduser hadoopgroup    102965 May 14 03:49 'Tutorial 7 (Part I).ipynb'
-rw-r--r-- 1 hduser hadoopgroup     19406 May 14 03:41  TweetsSentimentAnalysis.ipynb
-rw-rw-r-- 1 hduser hadoopgroup 238803811 Mar  4  2010  tweetsCA2.csv
hduser@reham-VirtualBox:~/Desktop/CA2$ hadoop fs -put ./tweetsCA2.csv /user1
hduser@reham-VirtualBox:~/Desktop/CA2$ hadoop fs -ls /user1
Found 17 items
-rw-r--r--   1 hduser supergroup        395 2023-04-06 13:42 /user1/HeightAndWeight_.csv
-rw-r--r--   1 hduser supergroup        948 2023-04-06 07:37 /user1/ProblemStatement1.txt
-rw-r--r--   1 hduser supergroup      11411 2023-04-13 22:20 /user1/airport-codes-na.txt
-rw-r--r--   1 hduser supergroup      19362 2023-04-06 07:07 /user1/auto-mpg-data-original.txt
-rw-r--r--   1 hduser supergroup     135287 2023-04-06 07:07 /user1/britney-spears.txt
-rw-r--r--   1 hduser supergroup   81372510 2023-04-13 22:26 /user1/ccFraud.csv.gz
drwxr-xr-x   - hduser supergroup          0 2023-05-06 23:51 /user1/data
-rw-r--r--   1 hduser supergroup   33396236 2023-04-13 22:20 /user1/departuredelays.csv
-rw-r--r--   1 hduser supergroup         56 2023-05-07 14:13 /user1/employeeNew.csv
-rw-r--r--   1 hduser supergroup         73 2023-04-13 22:19 /user1/people.json
-rw-r--r--   1 hduser supergroup      65167 2023-04-06 13:36 /user1/pg30123.txt
-rw-r--r--   1 hduser supergroup        419 2023-05-05 11:42 /user1/pig_script.pig
-rw-r--r--   1 hduser supergroup  211312924 2023-04-06 07:37 /user1/purchases.txt
-rw-r--r--   1 hduser supergroup        529 2023-04-06 07:07 /user1/sample.txt
drwxr-xr-x   - hduser supergroup          0 2023-05-05 11:43 /user1/student_output
-rw-r--r--   1 hduser supergroup  238803811 2023-05-14 20:45 /user1/training.1600000.processed.noemoticon.csv
-rw-r--r--   1 hduser supergroup  238803811 2023-05-16 14:41 /user1/tweetsCA2.csv
hduser@reham-VirtualBox:~/Desktop/CA2$
```

5- Open the Pyspark platform. The Jupyter file with the name "PySparkSentimentAnalysis" is attached. Please see it. The data set was loaded, and the exploratory data analysis approaches were applied to analyze the tweets. Also, data cleaning and preprocessing were applied. Finally, logistic regression machine learning was applied to classify the tweets into "negative" and "positive."



The model achieved accuracy = 73% .

[28]:
```
#Classification Report
y_true = predictionFinal.select(['Sentiment']).collect()
y_pred = predictionFinal.select(['prediction']).collect()
print(classification_report(y_true, y_pred))
```

```
2023-05-15 15:43:44,839 WARN scheduler.DAGScheduler: Broadcasting large task bina
```

	precision	recall	f1-score	support
0	0.73	0.73	0.73	160147
4	0.73	0.73	0.73	159771
accuracy			0.73	319918
macro avg	0.73	0.73	0.73	319918
weighted avg	0.73	0.73	0.73	319918

Abstract

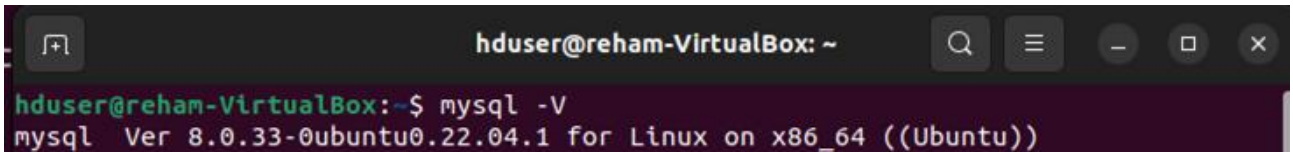
NoSQL databases were developed to meet the requirements of Big Data applications, which deal with large volumes of rapidly changing data and have a distributed architecture that supports easy scalability. However, many business applications operate in low-volume environments with structured and gradually changing information. These applications do not experience exponential data growth. NoSQL databases, being schema-less, do not adhere to the ACID properties and lack the reliability and consistency assurance provided by conventional SQL (relational) databases. This reliability and consistency are essential for numerous mission-critical small-scale applications.

Migration from a highly structured database to NoSQL for improved performance has become a popular trend among medium-scale businesses. However, the migration process is expensive and complex. This report aims to compare the performance differences between NoSQL document-oriented database MongoDB and relational database MySQL by replicating operations from both low-volume and high-volume applications using the Yahoo! Cloud Serving Benchmark (YCSB). The objective is to analyze the quantitative performance variations between these two database systems.

Practical

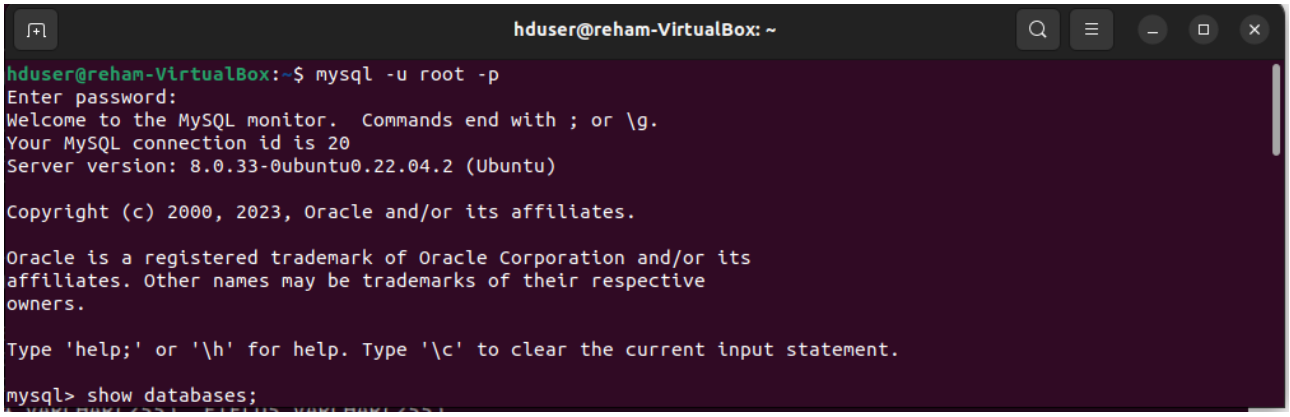
Before starting the YCSB. We need to use MySQL.

1- Check MYSQL version.

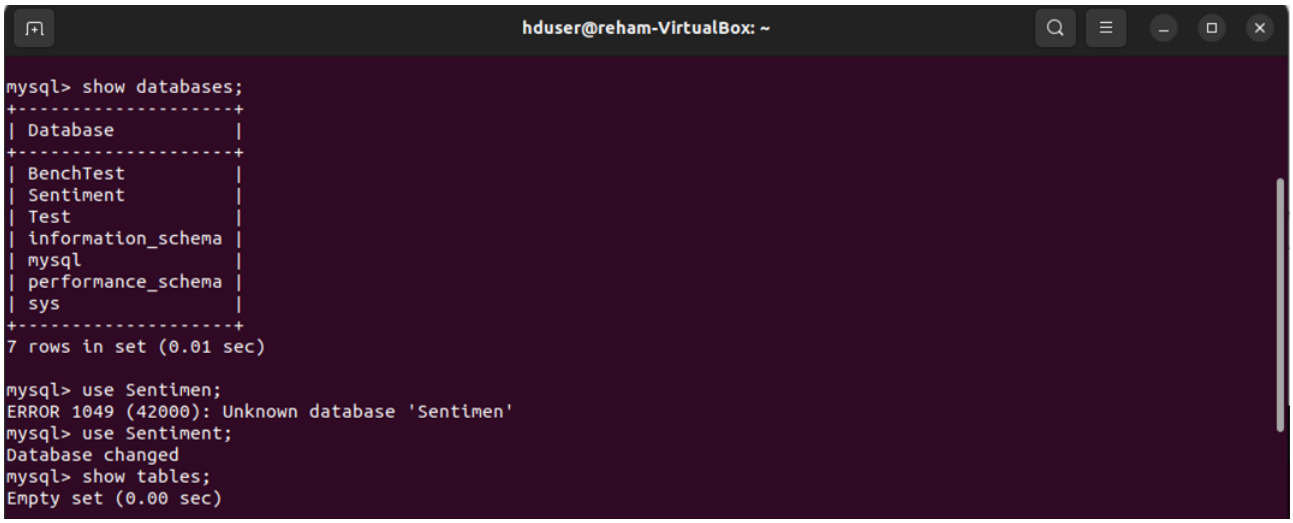


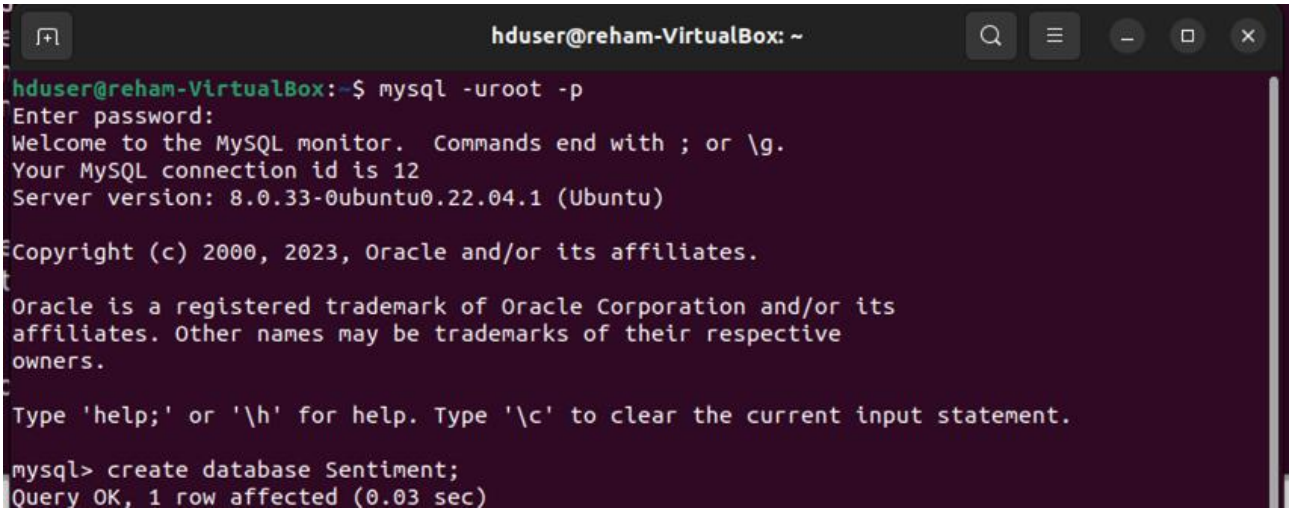
A version of MySQL was checked. It is 8.0.33

2- Start MySQl

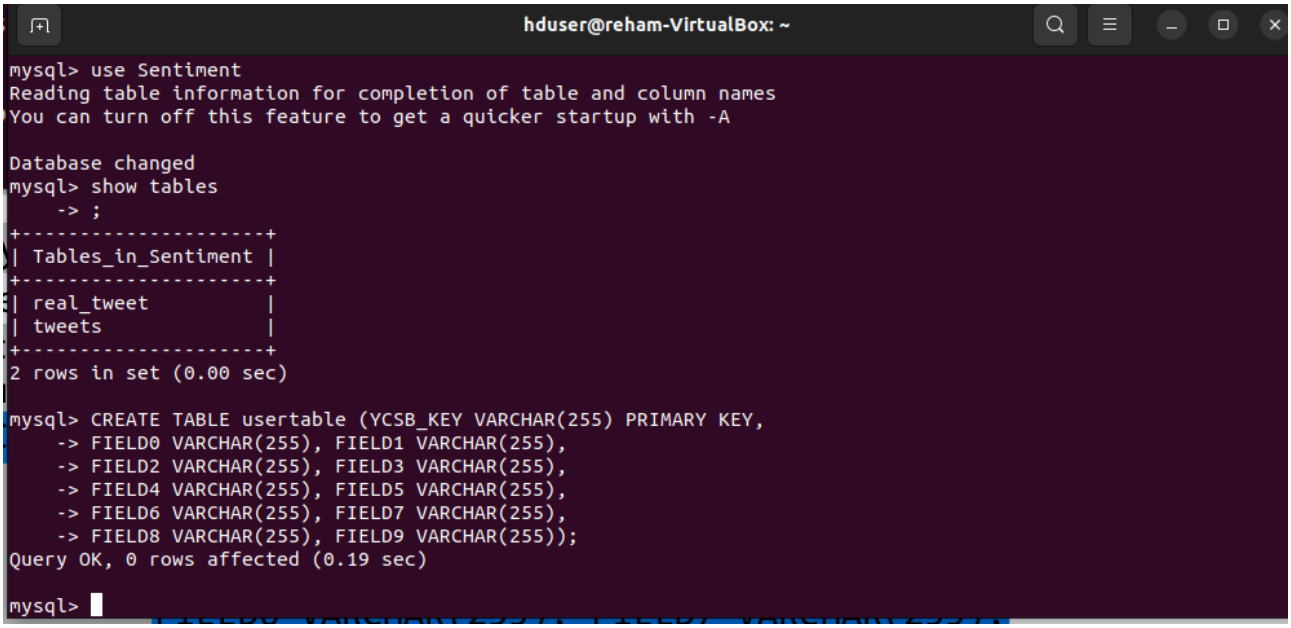


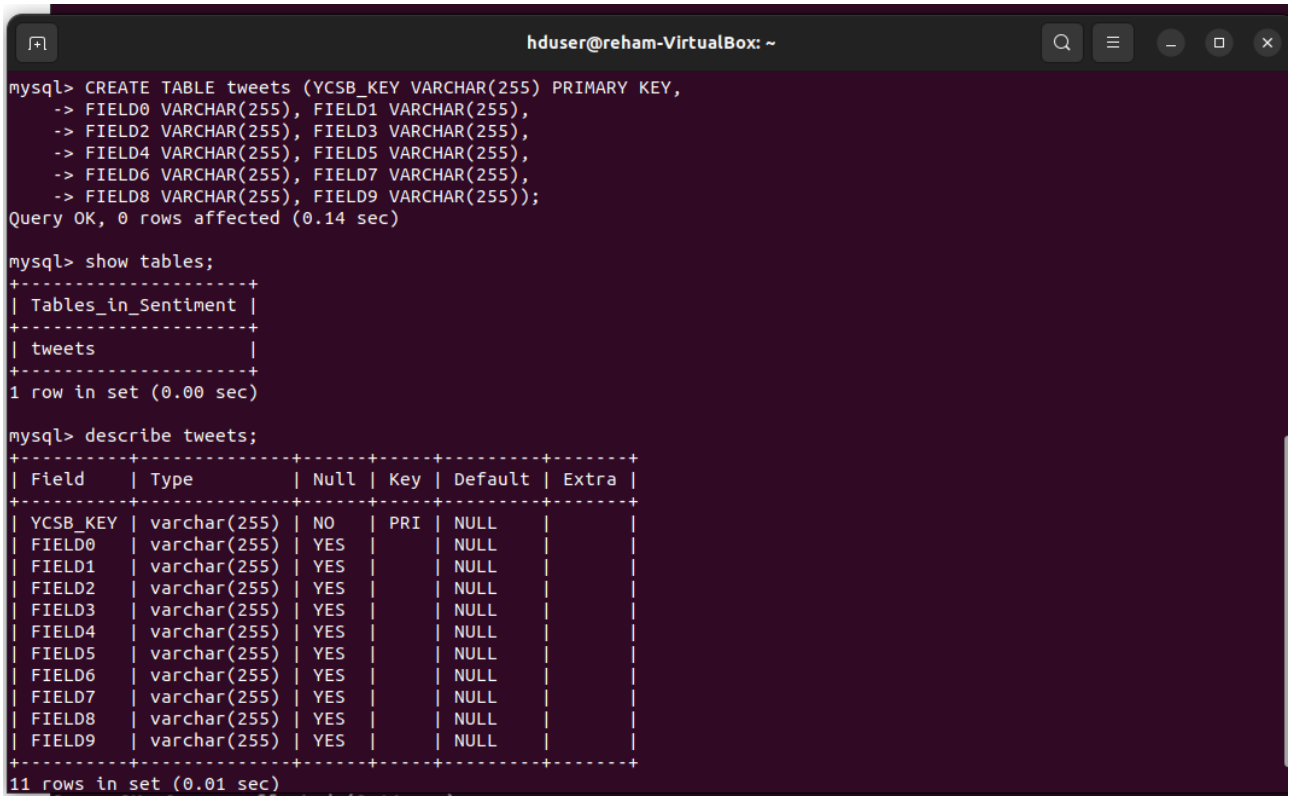
3- Display the Databases. A database named as “Sentiment” was created.





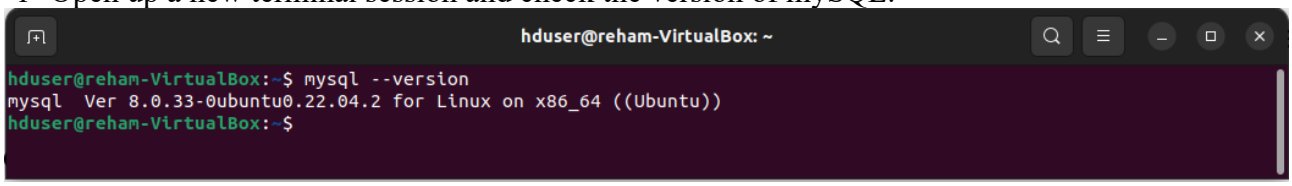
4- Anew table “usertable” created in MySQL as shown in the following figure.





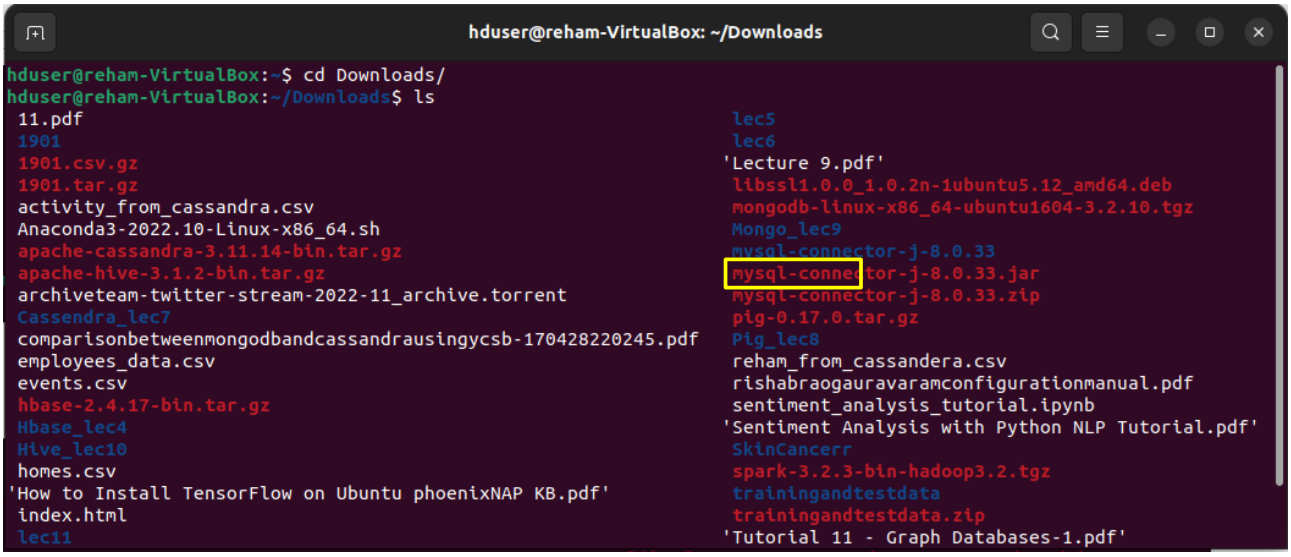
Now, we can use YCSB with mySQL and mongoDB

1- Open up a new terminal session and check the version of mySQL.

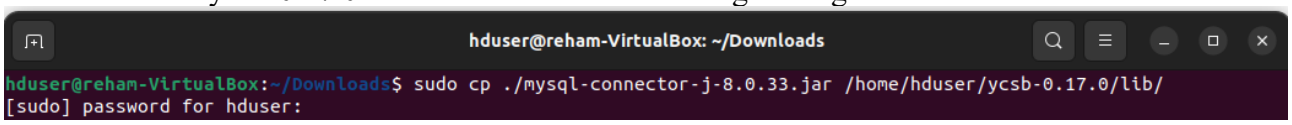


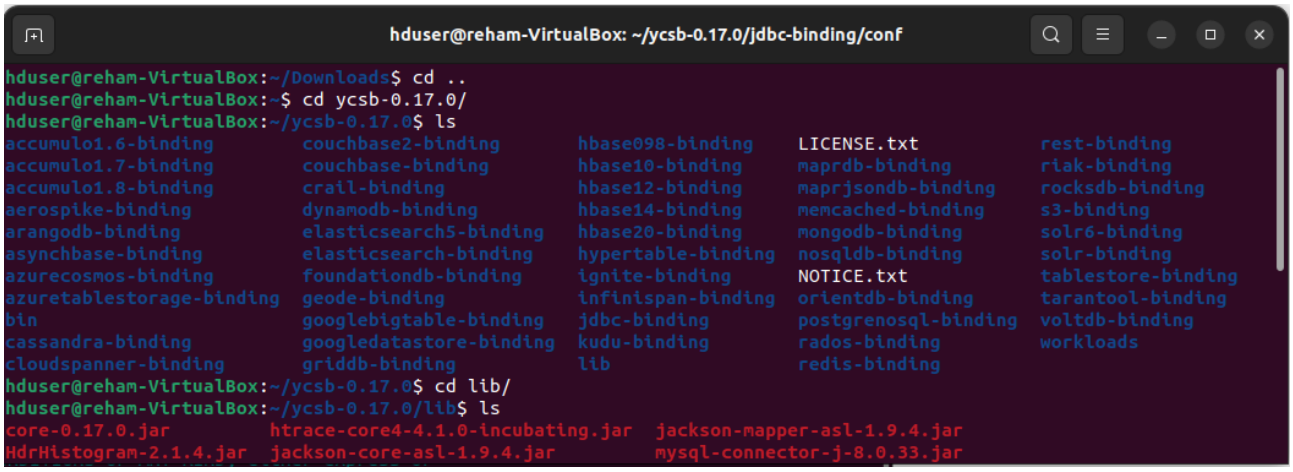
It's clear that we are using 8.0.33 mySQL

2- Download mysql connector 'mysql-connector-java-8.0.32.jar' file into Downloads folder

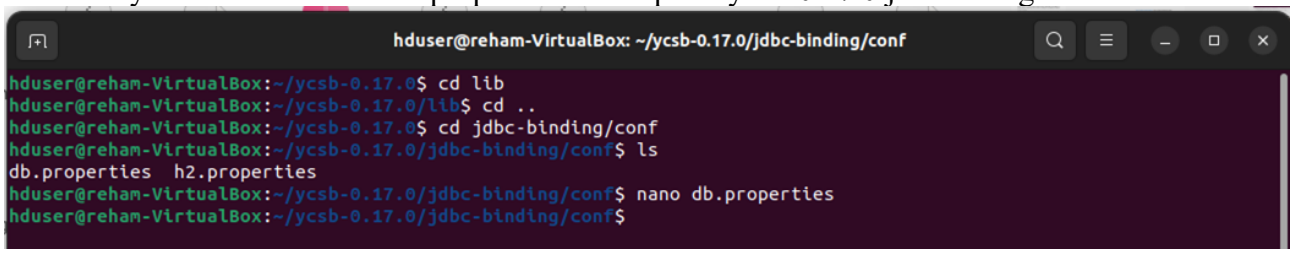


3- Copy the mysql-connector-java-8.0.33.jar file into the ycsb folder. In my cause the ycsb path is '/home/hduser/ycsb-0.17.0/lib' as shown in the following two figures:



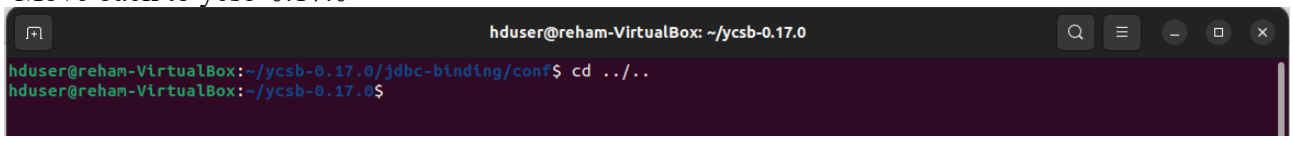


4- Modify the file named as ‘db.properties’ in the path ‘ycsb-0.17.0/jbc-binding/conf’ as shown



Save this file and exit by pressing Ctrl + X then type Y to select yes, and confirm to save the file with the same name.

Move back to ycsb-0.17.0



we will execute the workload with ycsb-0.17.0 by using the following commands

Execute a sample workload against mySQL and store the output at the file ‘output_workloada_ mysql_CA2.txt ’ as mentioned below

```
hduser@reham-VirtualBox: ~/ycsb-0.17.0
hduser@reham-VirtualBox:~/ycsb-0.17.0$ ./bin/ycsb.sh load jdbc -P ./jdbc-binding/conf/db.properties -P workloads/workloada > /ho
me/hduser/output_workloada_mysql_CA2.txt
Command line: -load -db site.ycsb.db.JdbcDBClient -P ./jdbc-binding/conf/db.properties -P workloads/workloada
YCSB Client 0.17.0

Loading workload...
Starting test.
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'. The driver is aut
omatically registered via the SPI and manual loading of the driver class is generally unnecessary.
DBWrapper: report latency for each error is false and specific error codes to track for latency are: []
hduser@reham-VirtualBox:~/ycsb-0.17.0$
```

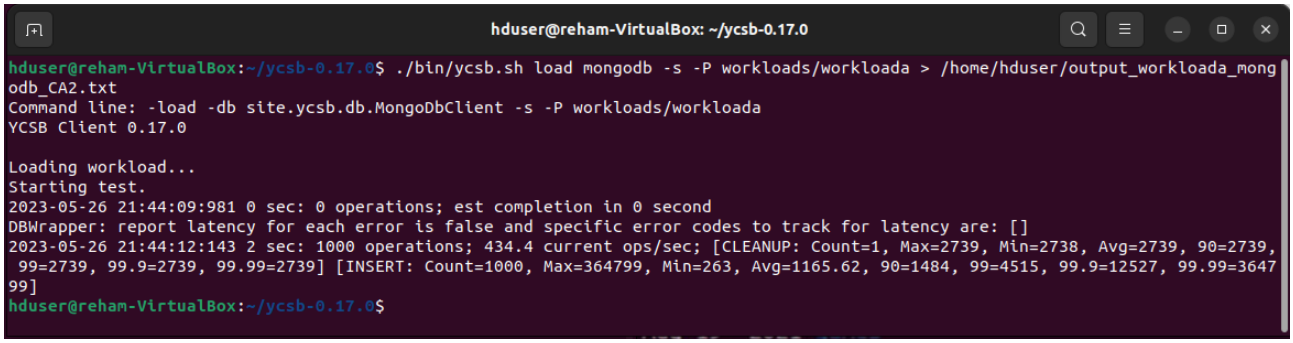
the output in the file 'output_workloada_mysql_CA2.txt':

```
Open   output_workloada_mysql_CA2.txt   Save
1 /usr/bin/java  -classpath /home/hduser/ycsb-0.17.0/conf:/home/hduser/ycsb-0.17.0/lib/HdrHistogram-2.1.4.jar:/home/hduser/ycsb-0.17.0/lib/
  core-0.17.0.jar:/home/hduser/ycsb-0.17.0/lib/htrace-core4-4.1.0-incubating.jar:/home/hduser/ycsb-0.17.0/lib/jackson-core-asl-1.9.4.jar:/home/hduser/
  ycsb-0.17.0/lib/jackson-mapper-asl-1.9.4.jar:/home/hduser/ycsb-0.17.0/lib/mysql-connector-j-8.0.33.jar:/home/hduser/ycsb-0.17.0/jdbc-binding/conf:/
  home/hduser/ycsb-0.17.0/jdbc-binding/lib/commons-collections-3.2.1.jar:/home/hduser/ycsb-0.17.0/jdbc-binding/lib/commons-lang-2.4.jar:/home/hduser/
  ycsb-0.17.0/jdbc-binding/lib/commons-pool-1.5.4.jar:/home/hduser/ycsb-0.17.0/jdbc-binding/lib/geronimo-jms_1.1_spec-1.1.1.jar:/home/hduser/
  ycsb-0.17.0/jdbc-binding/lib/geronimo-jta_1.1_spec-1.1.1.jar:/home/hduser/ycsb-0.17.0/jdbc-binding/lib/jdbc-binding-0.17.0.jar:/home/hduser/
  ycsb-0.17.0/jdbc-binding/lib/mysql-connector-j-8.0.32.jar:/home/hduser/ycsb-0.17.0/jdbc-binding/lib/openjpa-jdbc-2.1.1.jar:/home/hduser/ycsb-0.17.0/
  jdbc-binding/lib/openjpa-kernel-2.1.1.jar:/home/hduser/ycsb-0.17.0/jdbc-binding/lib/openjpa-lib-2.1.1.jar:/home/hduser/ycsb-0.17.0/jdbc-binding/lib/
  serp-1.13.1.jar site.ycsb.Client -load -db site.ycsb.db.JdbcDBClient -P ./jdbc-binding/conf/db.properties -P workloads/workloada
2 Adding shard node URL: jdbc:mysql://localhost:3306/Sentiment
3 Using shards: 1, batchSize:-1, fetchSize: -1
4 [OVERALL], RunTime(ms), 19792
5 [OVERALL], Throughput(ops/sec), 50.52546483427648
6 [TOTAL_GCS_G1_Young_Generation], Count, 4
7 [TOTAL_GC_TIME_G1_Young_Generation], Time(ms), 58
8 [TOTAL_GC_TIME_%_G1_Young_Generation], Time(%), 0.29304769603880354
9 [TOTAL_GCS_G1_Old_Generation], Count, 0
10 [TOTAL_GC_TIME_G1_Old_Generation], Time(ms), 0
11 [TOTAL_GC_TIME_%_G1_Old_Generation], Time(%), 0.0
12 [TOTAL_GCs], Count, 4
13 [TOTAL_GC_TIME], Time(ms), 58
14 [TOTAL_GC_TIME_%], Time(%), 0.29304769603880354
15 [CLEANUP], Operations, 1
16 [CLEANUP], AverageLatency(us), 4926.0
17 [CLEANUP], MinLatency(us), 4924
18 [CLEANUP], MaxLatency(us), 4927
19 [CLEANUP], 95thPercentileLatency(us), 4927
20 [CLEANUP], 99thPercentileLatency(us), 4927
21 [INSERT], Operations, 1000
22 [INSERT], AverageLatency(us), 18773.46
23 [INSERT], MinLatency(us), 9104
24 [INSERT], MaxLatency(us), 395007
25 [INSERT], 95thPercentileLatency(us), 30543
26 [INSERT], 99thPercentileLatency(us), 83199
27 [INSERT], Return=OK, 1000
```

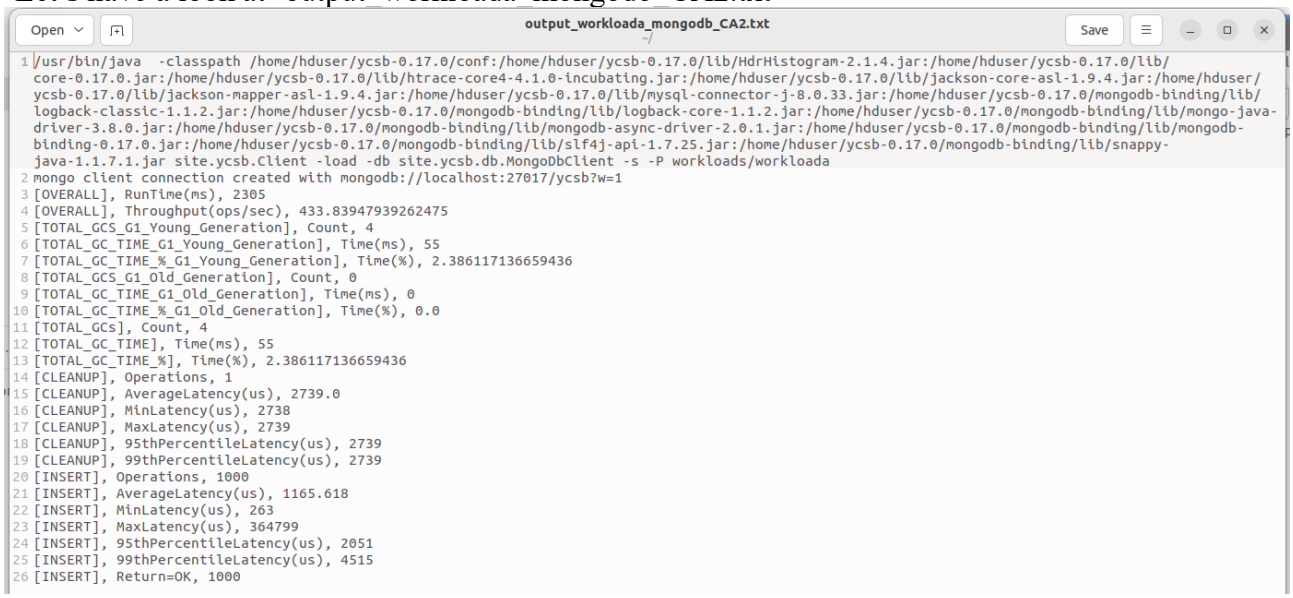
Before running another workloads, delete all the records stored in MySQL table (usertable)

```
hduser@reham-VirtualBox: ~
+--------------------------------------------------------------------------------------------------------+---------
+--------------------------------------------------------------------------------------------+---------------
+------------------------------------------------------------------------------------+-----------------------
+----------------------------------------------------------------------------+-------------------------------
+--------------------------------------------------------------------+-------------------------------------
+----------------------------------------------------------------+
1000 rows in set (0.01 sec)

mysql> delete from usertable;
Query OK, 1000 rows affected (0.12 sec)

mysql> select * from usertable;
Empty set (0.00 sec)

mysql>
```

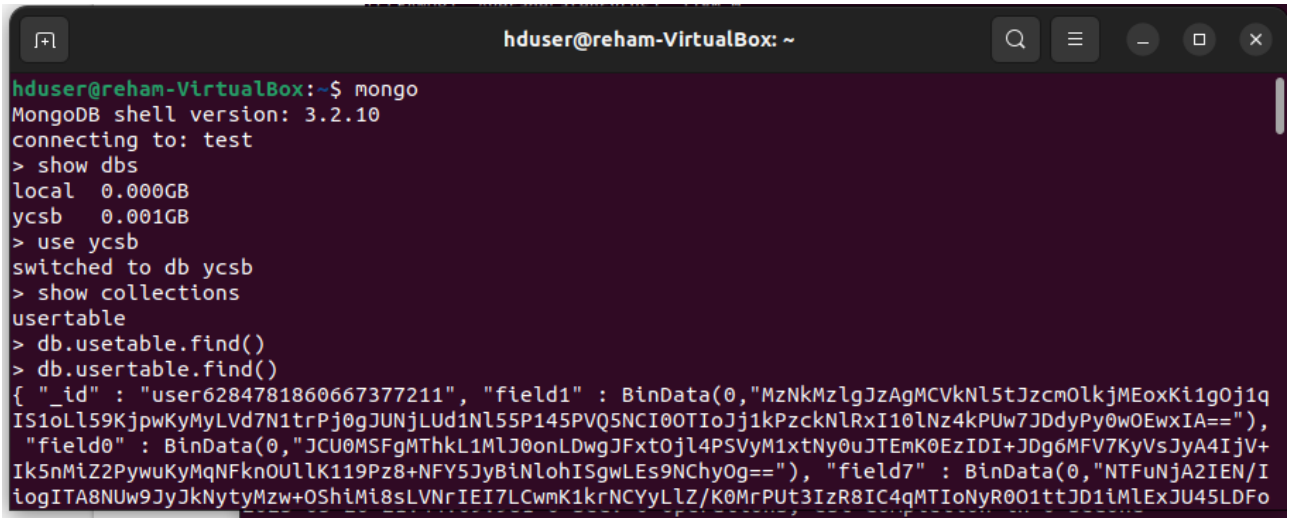
Execute a sample workload against mongoDB and save the output at 'output_workloada_mongodb_CA2.txt'.



Let’s have a look at ‘output_workloada_mongodb_CA2.txt’

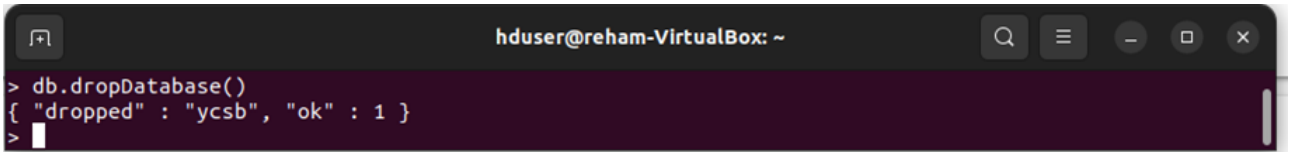


let’s check a database called "ycsb" and a collection named "usertable" in the MongoDB database to see if the MongoDB workloada was successfully executed or not. Utilizing the MongoDB client.

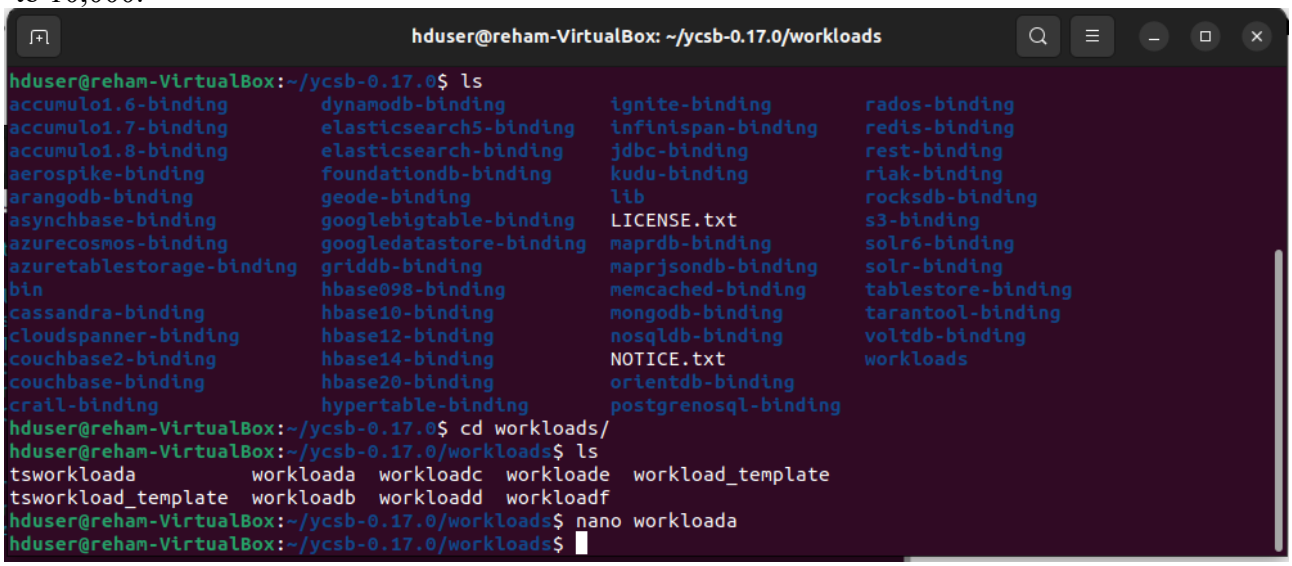


In order to execute the workload command again, remove the mondoDB database before the

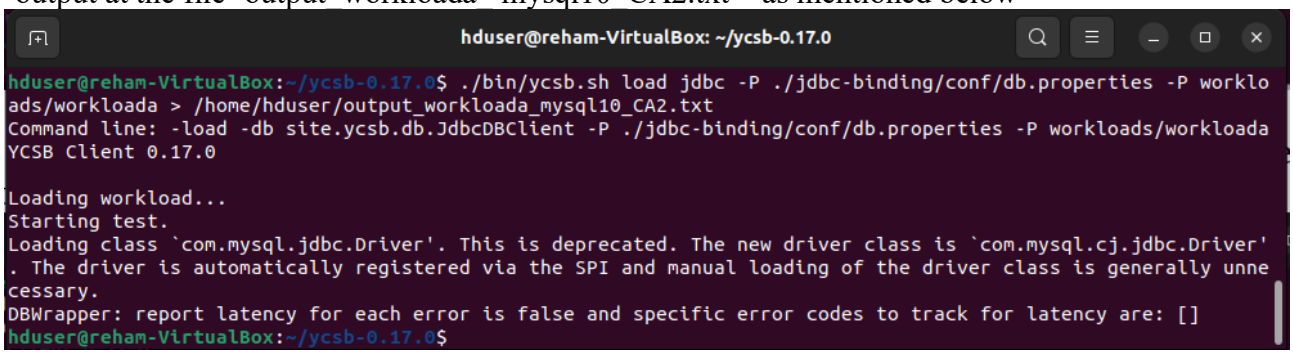
execution of the workload command for second workload



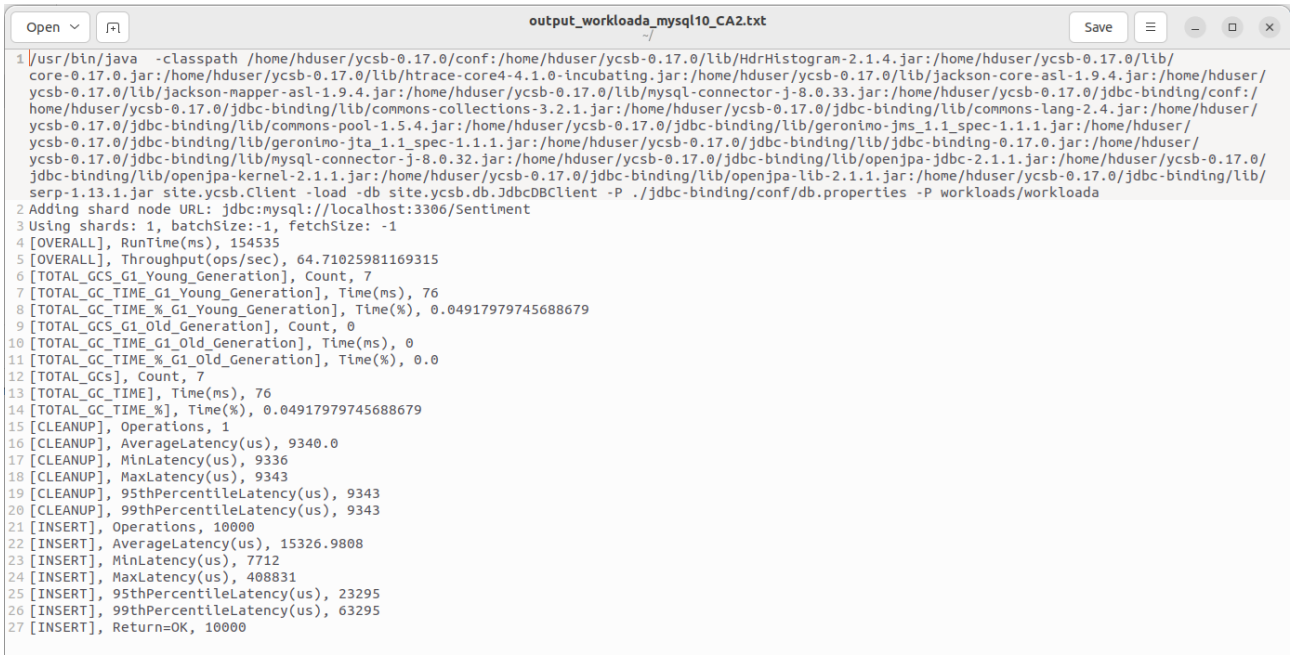
To perform a comparative analysis for read, write and update operations. the recordcount modified to 10,000.



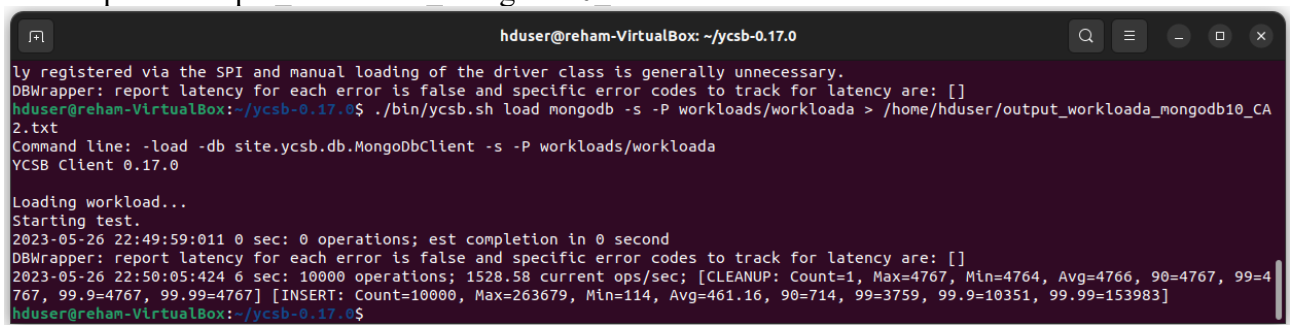
For Mysql and recordcount=10000, execute a sample workload against mySQL and store the output at the file ‘output_workloada_mysql10_CA2.txt ’ as mentioned below



the output in the file ‘output_workloada_ mysql10_CA2.txt’:



For Mongodb and recordcount=10000, Execute a sample workload against mongoDB and save the output at ‘output_workloada_mongodb10_CA2.txt’.

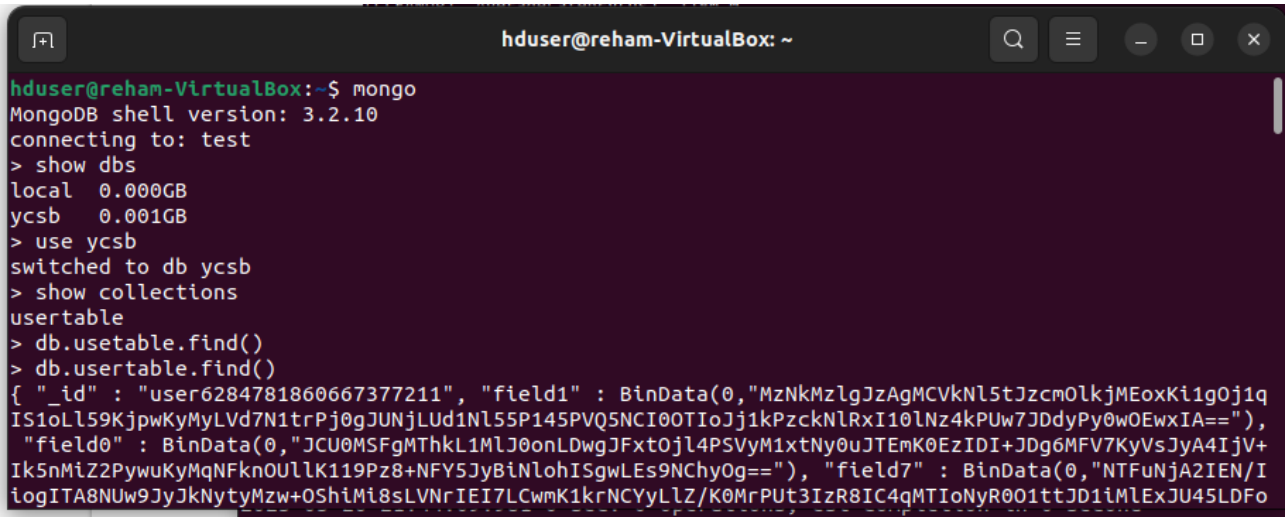


Let’s have a look at ‘output_workloada_mongodb10_CA2.txt’

Open output_workloada_mysql10_CA2.txt Save

```
/usr/bin/java  -classpath /home/hduser/ycsb-0.17.0/conf:/home/hduser/ycsb-0.17.0/lib/HdrHistogram-2.1.4.jar:/home/hduser/ycsb-0.17.0/lib/
core-0.17.0.jar:/home/hduser/ycsb-0.17.0/lib/htrace-core4-4.1.0-incubating.jar:/home/hduser/ycsb-0.17.0/lib/jackson-core-asl-1.9.4.jar:/home/hduser/
ycsb-0.17.0/lib/jackson-mapper-asl-1.9.4.jar:/home/hduser/ycsb-0.17.0/lib/mysql-connector-j-8.0.33.jar:/home/hduser/ycsb-0.17.0/jdbc-binding/conf:/
home/hduser/ycsb-0.17.0/jdbc-binding/lib/commons-collections-3.2.1.jar:/home/hduser/ycsb-0.17.0/jdbc-binding/lib/commons-lang-2.4.jar:/home/hduser/
ycsb-0.17.0/jdbc-binding/lib/commons-pool-1.5.4.jar:/home/hduser/ycsb-0.17.0/jdbc-binding/lib/geronimo-jms_1.1_spec-1.1.1.jar:/home/hduser/
ycsb-0.17.0/jdbc-binding/lib/geronimo-jta_1.1_spec-1.1.1.jar:/home/hduser/ycsb-0.17.0/jdbc-binding/lib/jdbc-binding-0.17.0.jar:/home/hduser/
ycsb-0.17.0/jdbc-binding/lib/mysql-connector-j-8.0.32.jar:/home/hduser/ycsb-0.17.0/jdbc-binding/lib/openjpa-jdbc-2.1.1.jar:/home/hduser/ycsb-0.17.0/
jdbc-binding/lib/openjpa-kernel-2.1.1.jar:/home/hduser/ycsb-0.17.0/jdbc-binding/lib/openjpa-lib-2.1.1.jar:/home/hduser/ycsb-0.17.0/jdbc-binding/lib/
serp-1.13.1.jar site.ycsb.Client -load -db site.ycsb.db.JdbcDBClient -P ./jdbc-binding/conf/db.properties -P workloads/workloada
Adding shard node URL: jdbc:mysql://localhost:3306/Sentiment
Using shards: 1, batchSize:-1, fetchSize: -1
[OVERALL], RunTime(ms), 154535
[OVERALL], Throughput(ops/sec), 64.71025981169315
[TOTAL_GCS_G1_Young_Generation], Count, 7
[TOTAL_GC_TIME_G1_Young_Generation], Time(ms), 76
[TOTAL_GC_TIME_%_G1_Young_Generation], Time(%), 0.04917979745688679
[TOTAL_GCS_G1_Old_Generation], Count, 0
[TOTAL_GC_TIME_G1_Old_Generation], Time(ms), 0
[TOTAL_GC_TIME_%_G1_Old_Generation], Time(%), 0.0
[TOTAL_GCs], Count, 7
[TOTAL_GC_TIME], Time(ms), 76
[TOTAL_GC_TIME_%], Time(%), 0.04917979745688679
[CLEANUP], Operations, 1
[CLEANUP], AverageLatency(us), 9340.0
[CLEANUP], MinLatency(us), 9336
[CLEANUP], MaxLatency(us), 9343
[CLEANUP], 95thPercentileLatency(us), 9343
[CLEANUP], 99thPercentileLatency(us), 9343
[INSERT], Operations, 10000
[INSERT], AverageLatency(us), 15326.9808
[INSERT], MinLatency(us), 7712
[INSERT], MaxLatency(us), 408831
[INSERT], 95thPercentileLatency(us), 23295
[INSERT], 99thPercentileLatency(us), 63295
[INSERT], Return=OK, 10000
```

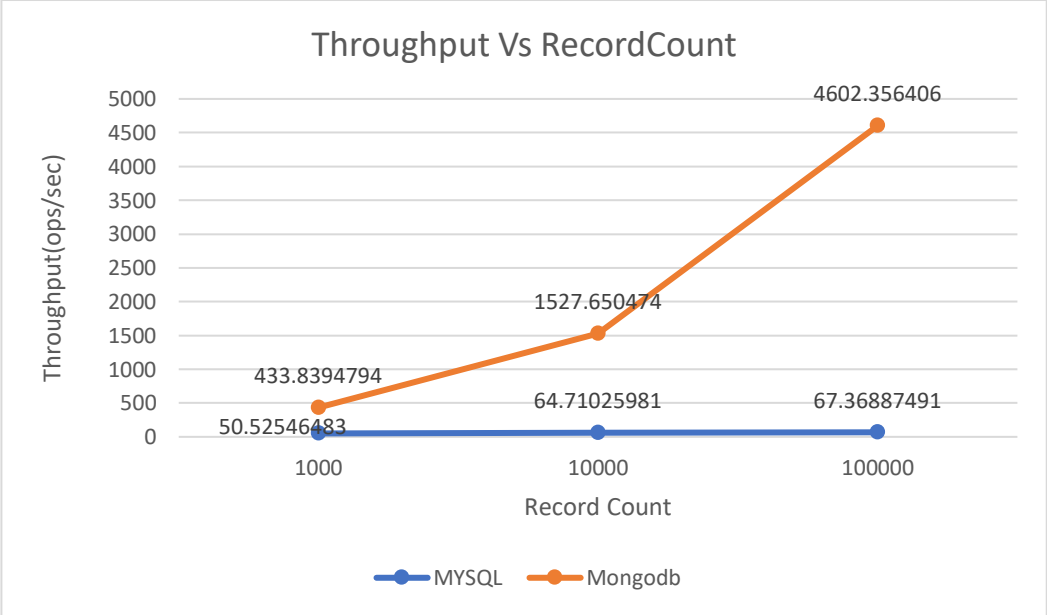
let’s check a database called "ycsb" and a collection named "usertable" in the MongoDB database to see if the MongoDB workloada was successfully executed or not. Utilizing the MongoDB client.



The Result:

This table analyzes the overall throughput (ops/sec), Average(us), MinLatency(us), and **MaxLatency(us** of MongoDB and MySQL when executed with multiple data sets. Throughput is a measure of how efficiently a database can handle and process operations within a given time frame. By comparing the throughput values, we can evaluate the performance of both databases in handling different workloads.

Record Count	Throughput(ops/sec)		Average(us)		MinLatency(us)		MaxLatency(us)	
	MYSQL	Mongodb	MYSQL	Mongodb	MYSQL	Mongodb	MYSQL	Mongodb
1000	50.52546	433.8394794	18773.5	1165.618	9104	263	395007	364799
10000	64.71026	1527.650474	15327	461.1614	7712	114	408831	263679
100000	67.36887	4602.356406	14820	199.85302	7092	78	739327	409855

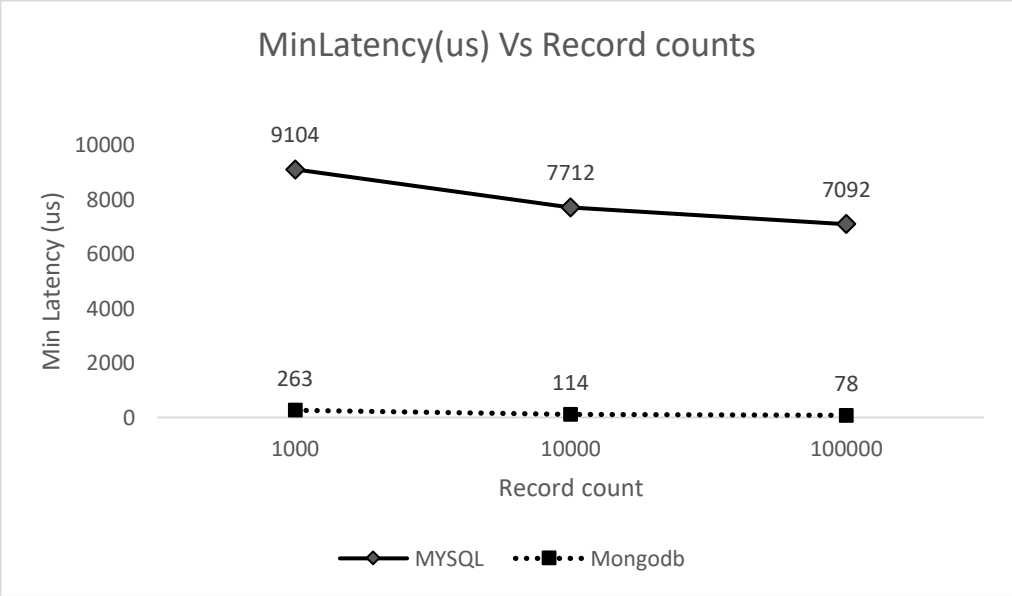


This figure shows the total run throughput of workload A (Insert, Read, Update) for both MongoDB and MySQL. MongoDB demonstrates a higher overall throughput. In contrast, MySQL has a significantly lower throughput. MongoDB performs best when executed with the largest data set (100,000),while MySQL achieves its highest throughput at 10,000 records.

Let’s describe the key difference between MinLatency and MaxLatency is:

MinLatency: This refers to the shortest response time observed during a benchmarking process or measurement. It represents the best-case scenario, indicating the quickest time taken for a request or operation to be processed and responded to by the system. while the MaxLatency represents the longest response time recorded during the benchmarking process.

To recap minimum latency refers to the shortest response time, while maximum latency refers to the longest response time observed during a benchmark or measurement, providing insights into the best and worst case performance of the system.



This figure shows MinLatencey for MYSQL and MongoDB. In the case of MySQL, the minimum latency ranged from 7092 to 9104 milliseconds across different record numbers, indicating the fastest response time observed with the record number 100000. MYSQL work better with the large dataset.

Similarly, for MongoDB, the minimum latency varied from 78 milliseconds to 263 milliseconds, reflecting the shortest response time achieved at the dataset with 100000 records.

Both MYSQL and Mongodb work better with 100000 records.

References:

1- Abramova, V., Bernardino, J. & Furtado, P. (2015), 'Sql or nosql? performance and scalability evaluation', International Journal of Business Process Integration and Management 7, 314. URL: https://doi.org/10.1504/IJBPIM.2015.073655

2- Hyndman, R.J. and Athanasopoulos, G. (2018) Forecasting: principles and practice. 2nd edition. Lexington, Ky.: OTexts.

3- JeonDong-cheol, Mun, L. & HwangHeejoung (2018), 'Performance analysis of rdbms and mongodb through ycsb in medical data processing system based hl7 fhir', Journal of Korea Multimedia Society 21(8), 934–941. URL: https://doi.org/10.9717/KMMS.2018.21.8.934

4- Overview — Panel v1.1.0 (no date). Available at: https://panel.holoviz.org/ (Accessed: 12 June 2023).