WQD7008

PARALLEL AND DISTRIBUTED COMPUTING

**1/2022/2023**

Yelp Dataset Analysis With Hadoop, Pig, Spark, Hive and Tableau

**GROUP PROJECT**

| **No.** | **MATRIC ID** | **NAME** |
| --- | --- | --- |
| **1** | **17186722/2** | **CHEN BAOGANG** |
| **2** | **S2191604** | **ONG HORNG NENG** |
| **3** | **22052733** | **RAJASEGARAN A/L M SIVAANANDAN** |
| **4** | **S2179001** | **NG CHENG WEI** |
| **5** | **22056764** | **DEVAYANI A/P BALKRISHNAN** |

[1. Introduction](#_heading=h.vvd3cgk74e0a)

[1.1. Objective](#_heading=h.oxelcqbbk9xq)

[2. Literature Review](#_heading=h.lfjhihnsrgiy)

[3. Methodology](#_heading=h.tew278qhlkx7)

[3.1. System Design](#_heading=h.hg67ng5sazbw)

[3.1.1. Overview](#_heading=h.b4w5tkufy9vx)

[3.1.2. Hadoop Components](#_heading=h.9y3ovqgxn1g1)

[3.1.3. Project Components](#_heading=h.rmukahxnm4ws)

[3.2. Dataset Collection](#_heading=h.o0lsq2mwvaah)

[3.3. Data Pre-processing](#_heading=h.j0o5mps1mauq)

[3.4. Word count task](#_heading=h.e1i7n86van8n)

[3.5. Data Analysis By Hadoop, Pig , Spark and Hive](#_heading=h.sjsl329mcjvs)

[3.6. Visualization By Tableau](#_heading=h.x76bdji7tt5y)

[4. Results and Discussion](#_heading=h.tyodfzndnvyj)

[5. Challenges](#_heading=h.e9d997dn0a25)

[6. Conclusion](#_heading=h.jrob95fxp5t)

[7. References](#_heading=h.5za7zxm1t9h1)

[8. Appendix](#_heading=h.6mngxula8288)

[8.1. Pig Script](#_heading=h.tqqg4qf5rvfi)

# Introduction

Yelp is a widely-used public database that allows individuals to access information about businesses. The Yelp Open dataset is a portion of Yelp's listings, reviews, and user information that has been made accessible to the public for use in personal, scholarly, and instructional contexts. The website promotes its utilization to teach students about databases, practise NLP, or to be used as sample production data while learning to create mobile apps. The files are available in JSON format. The database is extensive, containing over 6.9 million reviews from 1.9 million users for around 150,000 businesses from 2014 to June 2022. The sheer size of the data, which is over 4 GB when compressed and more than 9 GB when uncompressed, presents a challenge for traditional data analysis methods. However, it also presents an opportunity to utilize distributed systems and technologies in analyzing the dataset. Therefore, in this project, we explored the yelp dataset using different big data technologies such as hadoop, yarn, pig, spark, and hive to discover data insights that can help improve business performance.

# Objective

The primary objectives of this project are to evaluate the speed of Hadoop HDFS and Yarn in performing the word count task on business reviews. Secondly, we want to conduct data analysis using Hadoop HDFS, Yarn, Spark, and Hive. Additionally, we want to produce visual outputs through the use of Tableau.

* Using hadoop hdfs and yarn to perform word count task
* Using hadoop hdfs, yarn, Pig, Spark and Hive to conduct data analysis
* Using Tableau to visualize the output

# Literature Review

A similar study was conducted by Alamoudi & Azwari in 2021 where the researchers conducted EDA and data mining on Yelp restaurant reviews. By detecting client needs and highlighting areas of dissatisfaction, online review tracking aids service companies in improving their goods and services. A way to further improve this would be to find the important phrases in Yelp reviews that can be used to forecast the number of stars that the same reviewer will award the same company. This study was conducted by Zinoviev (2021). Interestingly, neither of these studies mention the use of distributed systems to deal with the large sets of data they were working with.

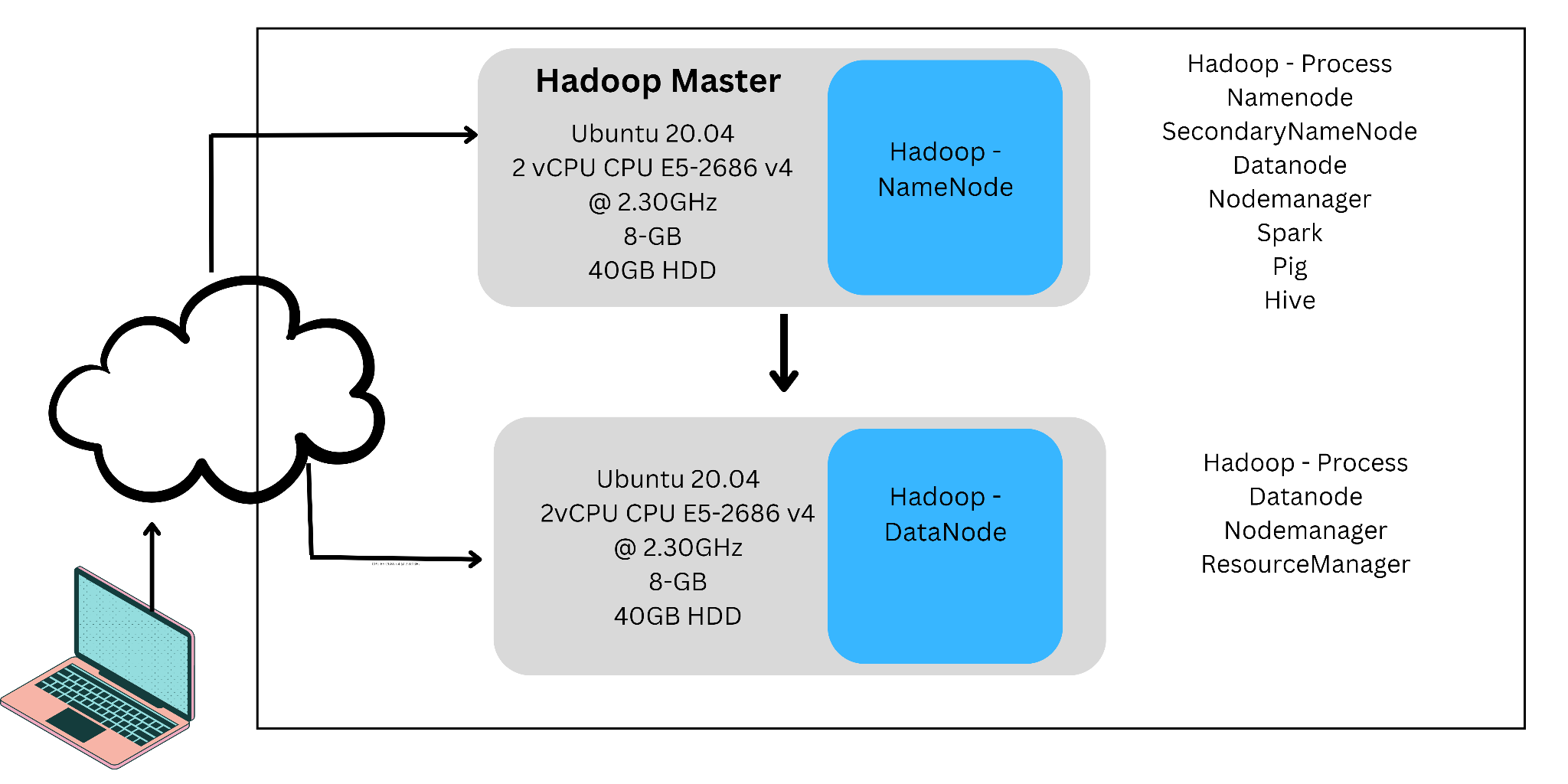
# Methodology

# System Design

# Overview

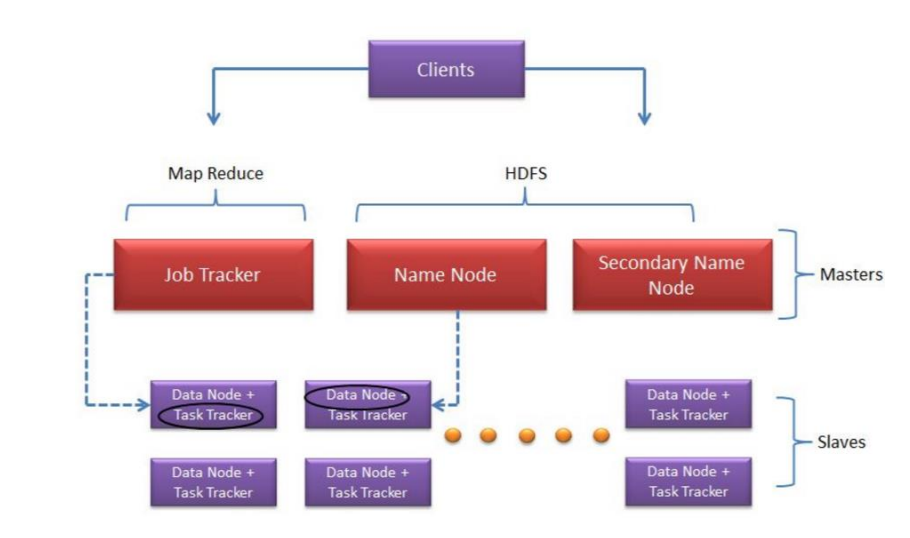
A complete system design is shown in the diagram below illustrates the workflow between the major components of Hadoop. In simple terms, data storage is done by uploading data from a local host into a Hadoop HDFS cluster which is hosted in AWS EC2 instances. The output from the cluster is stored back in HDFS.

This output is rendered and shown in a graphical manner using Tableau, which is hosted on a local machine.



# Hadoop Components

Hadoop consists of five daemons. They are divided between the master node and SlaveNodes. Master daemons consist of three Hadoop daemons such as the NameNode, SecondaryNameNode and a JobTracker. Whereas, the slave daemons are the DataNodes and the TaskTracker. Daemon is a background process. Every master service can talk to each other and all slave daemons can interact with each other. If NameNode is a Masternode its corresponding SlaveNode is DataNode. JobTracker talks to TaskTracker. If the NameNode is JobTracker its corresponding SlaveNode is TaskTracker as shown in the figure below.



*Hadoop daemons and core components*

**Components of Hadoop**

**NameNode**: It stores and maintains the metadata of HDFS and tracks where the data file is kept across the cluster. It is a single point failure for HDFS which means when the NameNode goes down, the file system goes offline.

**Secondary NameNode**: It is used to perform the housekeeping functions for the NameNode. It can be hosted on a separate machine and acts as a backup.

**JobTracker**: It manages the MapReduce jobs and distributes individual tasks to the machines running the TaskTracker.

**DataNode**: It stores actual HDFS data blocks TaskTracker. It is mainly responsible for instantiating and monitoring individual map and reduce tasks. DataNodes are commodity hardware which means if the system goes down the data doesn’t lose since HDFS has been given 3 replications by default.

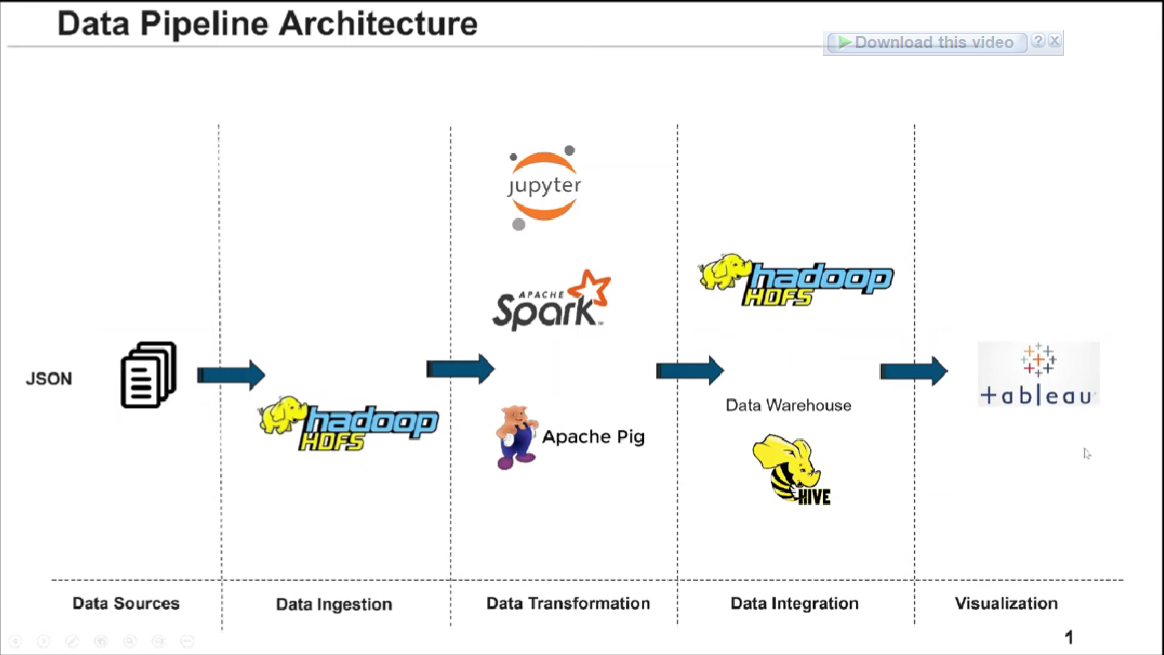
**MapReduce**: MapReduce framework is used to split data into small pieces and execute the related jobs on nodes. The results will be collected from nodes, integrated and then return to users. In this way, MapReduce transforms a single-node processing job to a parallel processing job to improve the execution efficiency.

**YARN**: Yarn framework is used for job scheduling and cluster resource management.

**HDFS**: HDFS is a specially designed File System for storing huge data sets with clusters of commodity hardware with streaming access pattern which means “Write Once Read Many Times”. The block size of each file is 64MB or 128MB.

A heartbeat is sent from the TaskTracker to the JobTracker every few minutes to check its status. One important thing to keep in mind is that Hadoop master nodes don’t talk to the SlaveNodes. However, all the DataNodes can talk amongst themselves. The metadata is stored in the namespace of the NameNode which keeps track of all the tasks that’s being done.

# Project Components



**AWS cloud service**: Amazon Elastic Compute Cloud is a part of Amazon.com's cloud-computing platform, we use Amazon Web Services to create two instances which run big data softwares.

**VMware**: We also tried to create three virtual machines with VMware in a local laptop for local testing and local debugging.

**Jupyter**: For data pre-processing, we have used python in Jupyter notebook to perform basic data cleaning and data processing.

**HDFS**: HDFS is a distributed file system that handles large datasets running on commodity hardware. It is used to scale a single Apache Hadoop cluster to hundreds (and even thousands) of nodes. In this project , we have set up the first instance as namenode, second node as well as datanode, and second instance as datanode, resource manager.

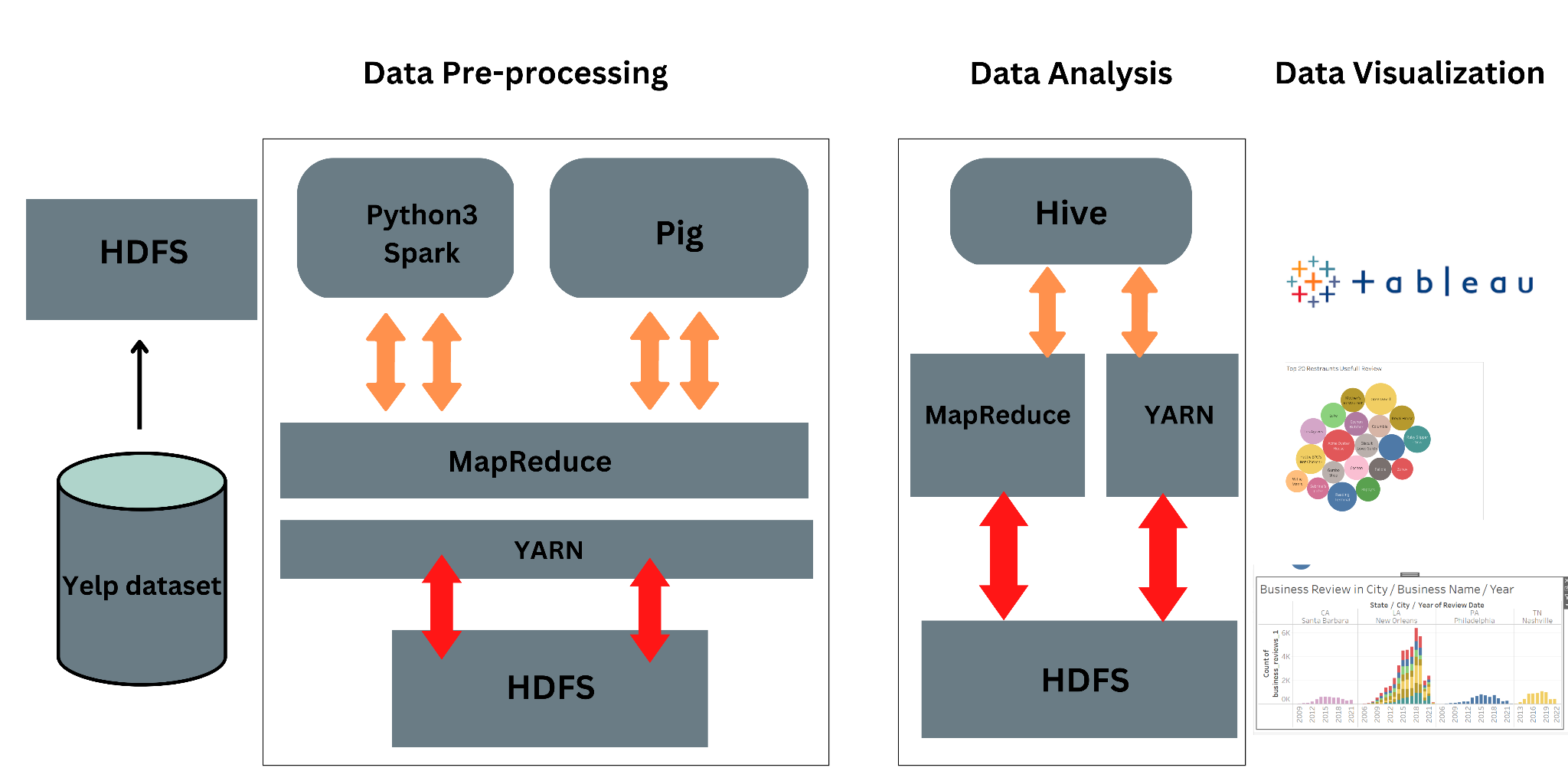
**Yarn**: YARN stands for “Yet Another Resource Negotiator”, YARN was described as a “Redesigned Resource Manager” at the time of its launching, but it has now evolved to be known as large-scale distributed operating system used for Big Data processing, in this project , we set up yarn in the second instance as resource manager to handle big data tasks.

**Pig**:Apache Pig is a platform for large-scale data analysis that combines infrastructure for programme evaluation with a high-level language for describing data analysis applications. Pig programmes stand out because its structure lends itself to significant parallelization.

**Spark**: Apache Spark is a multi-language engine for doing data engineering, data science, and machine learning on single-node machines or clusters.

**Hive**: It is a data warehouse built on top of Hadoop and is used for analyzing, summarizing and querying data using HIVE Query Language (HQL). These queries are compiled into map-reduce jobs which are executed by Hadoop. The data is organized in the form of tables using HIVE”. It is the module that allows the extraction logic of the data to be formulated using an SQL-like language.

**Tableau**: Tableau is an excellent data visualization and business intelligence tool used for reporting and analyzing vast volumes of data. After running all scripts , the final output is conducted by Hive, then we import the excel file to Tableau Desktop and create different sheets to draw different graphs.



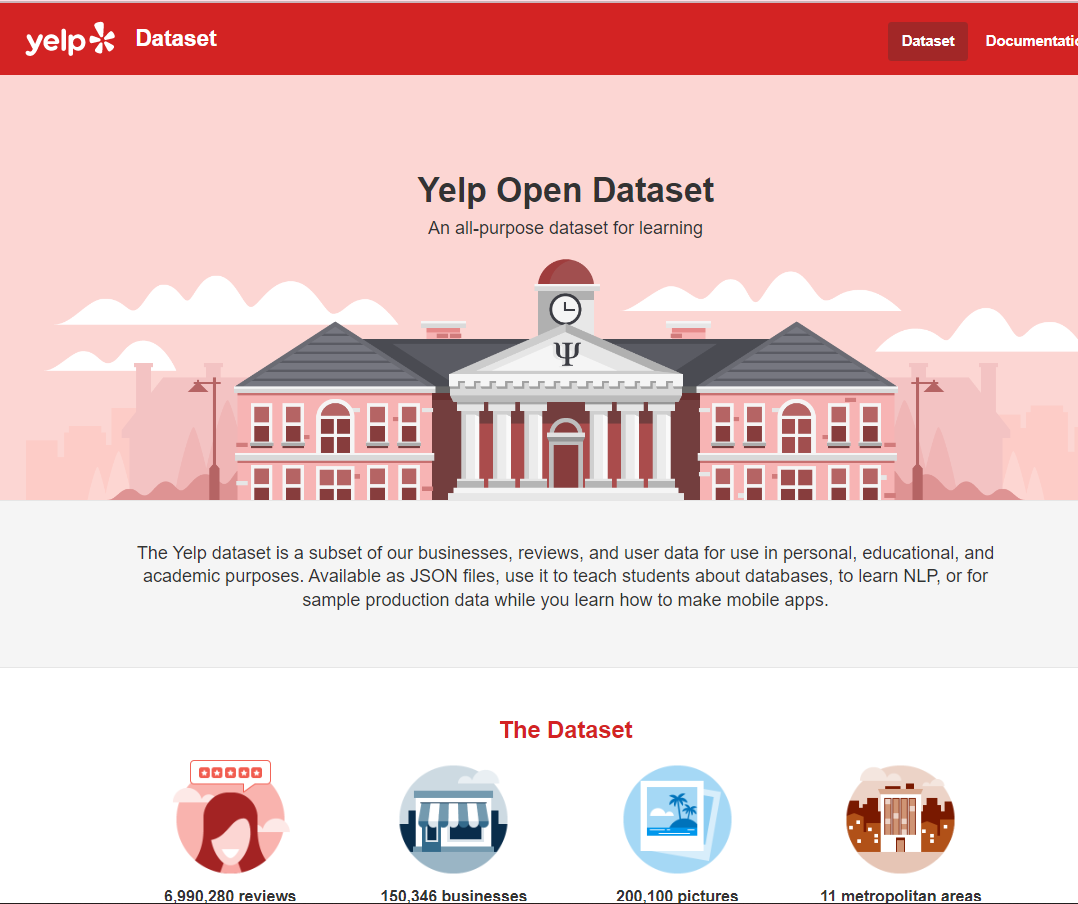
The main idea and concept of this project are discussed along with the architecture required for it. Data used in this project is discussed and the tools required to start Hadoop and its services are discussed in detail. Some of the core components of Hadoop are NameNode, DataNodes, Secondary NameNode, JobTracker, and TaskTracker. The next section gives a brief description of how the data is provided and analyzed and extracted for visualization in tableau.

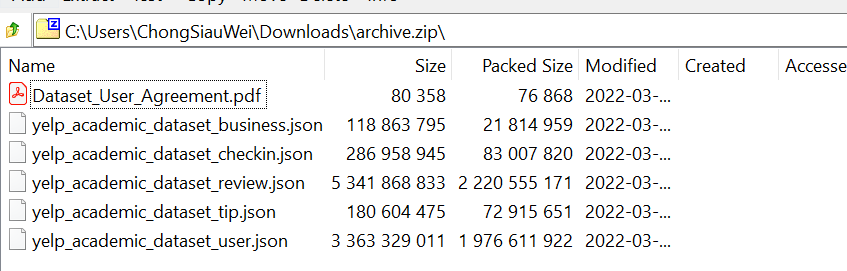
# Dataset Collection

This dataset was obtained from Yelp: <https://www.yelp.com/dataset>. Over 1.2 million business attributes like hours, parking, availability, and ambience in the Yelp dataset.

The Yelp dataset consists of five data feeds, which are business, review, pictures, etc. However, due to the limited resources on our devices, we primarily work on the reviews of all the academic-related businesses.

The dataset is 4.04 GB in compressed form, and 9.29 GB after unpacking. We want to read only the “yelp\_academic\_dataset\_review” JSON file from the dataset, of size 5.34 GB and the sample raw structure is as follows:





# Data Pre-processing

**Import JSON format file.**

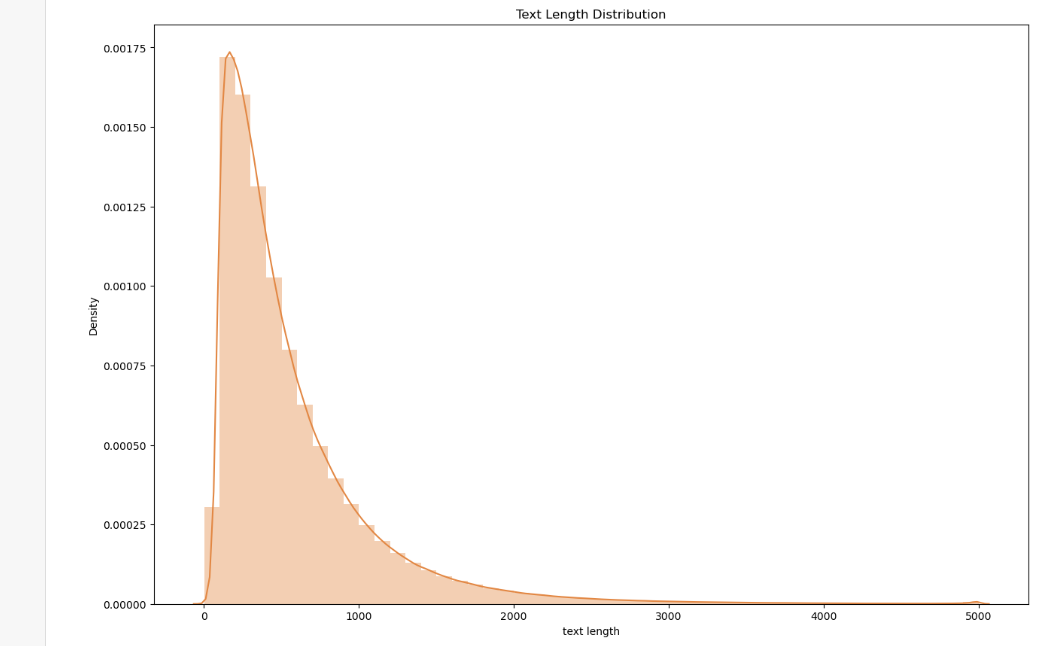
* Import the unpacking JSON file by using python.



**Convert data to Pandas data frame**

* We have converted the data to pandas Data Frame with two columns and applied a new column text length:
* Stars : Number of stars given by the user to the business.
* Text : User review text.
* Text length: Length of review text.



* We create a distplot plot to observe the distribution of text lengths.

**Tokenizing the whole dataset**

* Performing this step is due to there being some issues with the extracted string. Sample as below:

1. Punctuation is not handled properly: "buy??" will be considered as a different word from "buy".
2. Contractions are not understood: "didn't" is not going to be related to "did", nor to "not".

**Data Cleaning**

* After performing the tokenization, we have removed all the null values, unnecessary attributes, brackets, numbers, and punctuation marks. Besides that, we have also converted the whole dataset to lowercase.

Sample code as below:



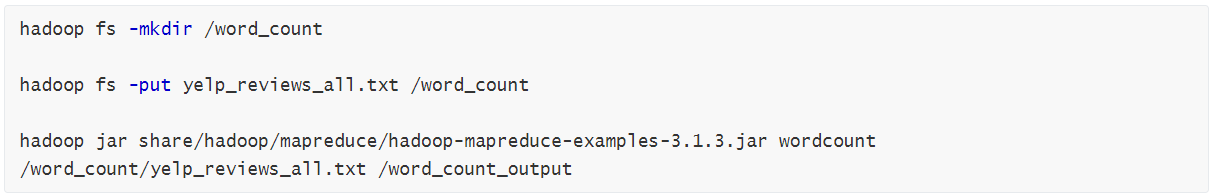
**Export to CSV.**

* Export the clean data to csv for further action.

# Word count task

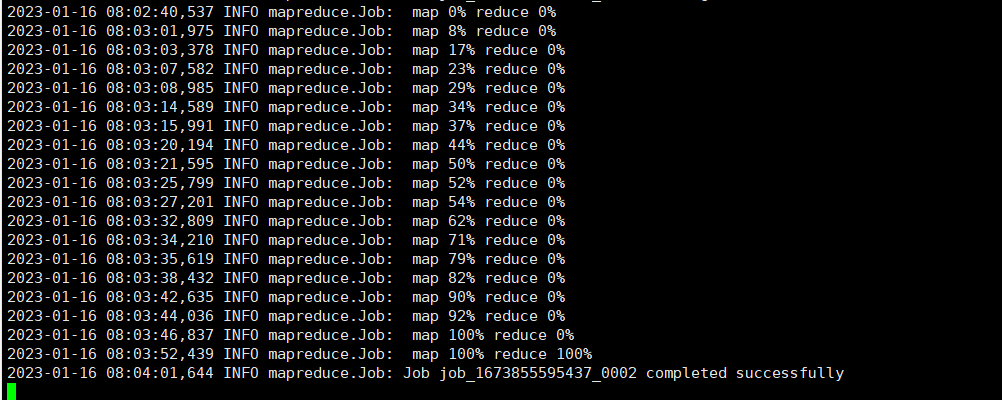
In the word counting task, we use Hadoop/Yarn MapReduce technology, take the Yelp review dataset as input, and count the number of times different words appear in each review. Then, the program outputs the count of each word. The final consideration in evaluating the effectiveness of a distributed system, particularly Hadoop HDFS and YARN in managing large datasets, is the amount of time it takes to complete word count tasks.

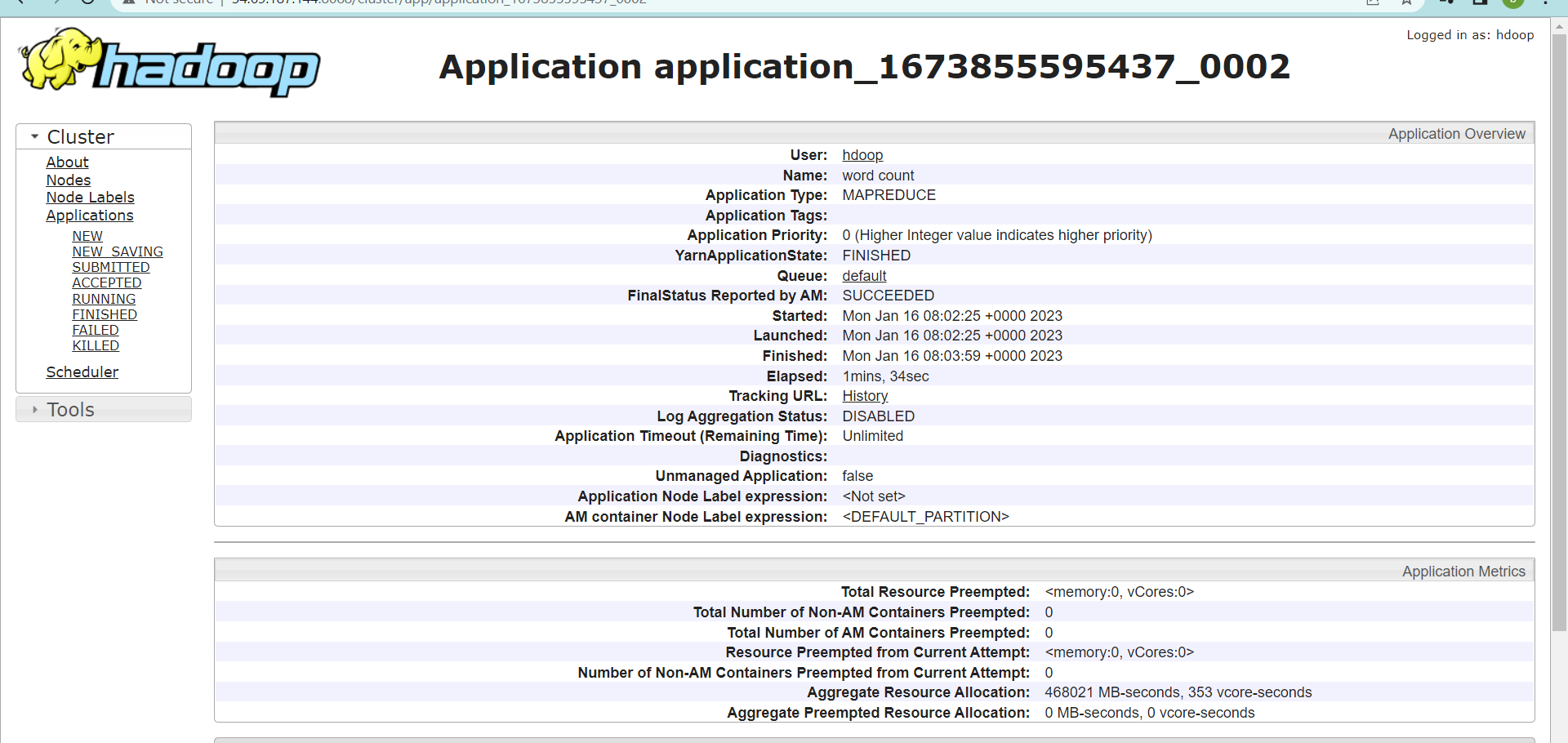
**Upload data into hadoop hdfs**

****

In the beginning, we create a folder named word\_cout in hdfs, and then put review data from the linux system into the hdfs platform, lastly, we run WordCount application in hadoop that counts the number of occurrences of each word in a given input set.

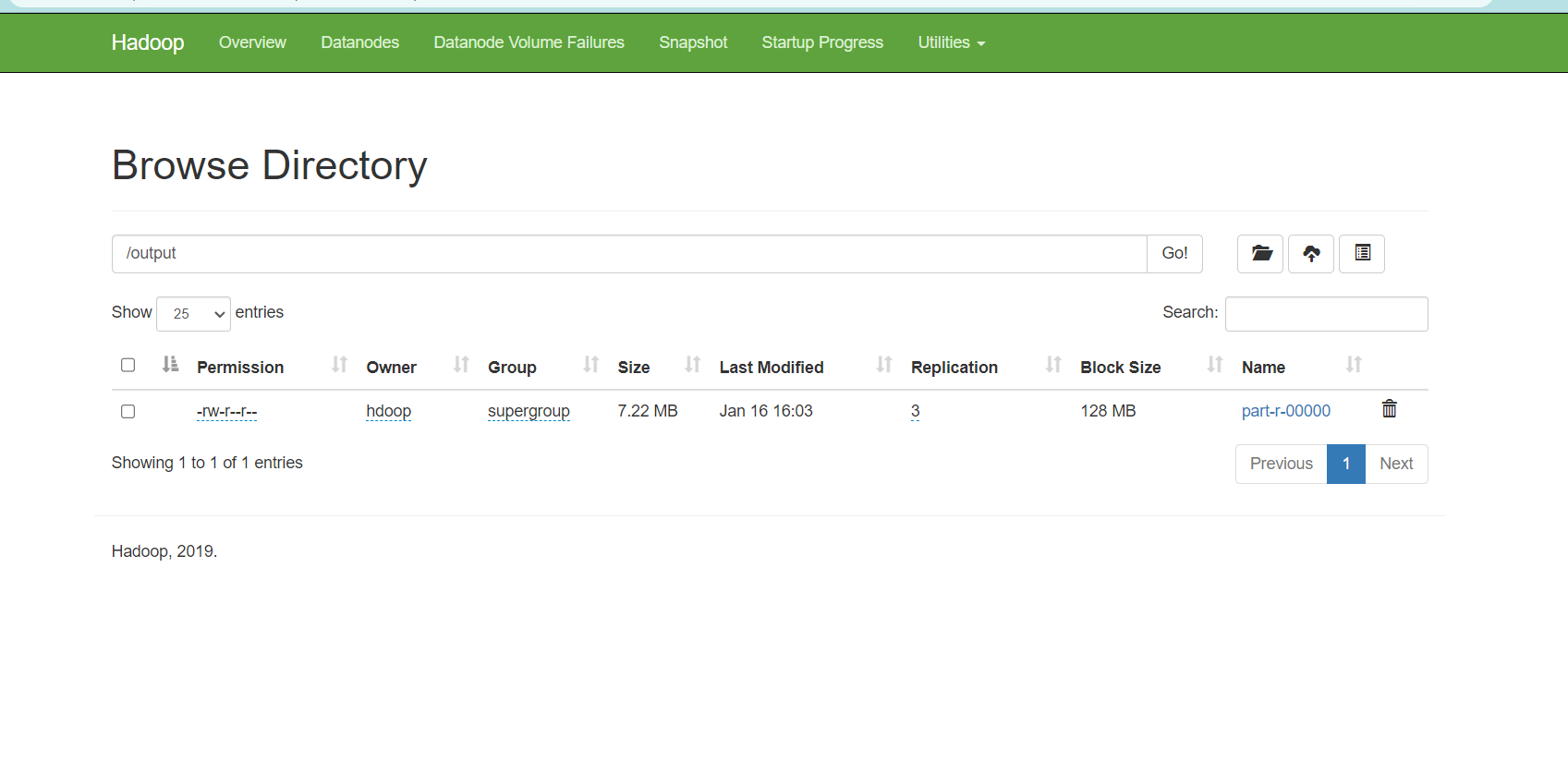
**Run WordCount task with Mapreduce:**

****

****

We can get a detailed look at the running WordCount task by checking the hadoop cluster logs, it clearly shows that we completed the map and the reduce process in just 1 minute and 34 seconds. its status is successful which means WordCount application already completed word count task and we already got output now.

**Output in HDFS web UI**

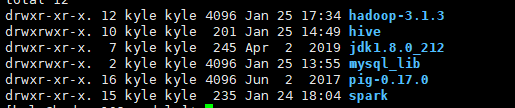
****

After successfully running WordCount task, the output will automatically stored in HDFS, and we can find the output in the HDFS Web UI, It provides various operations such as download, check file header (first 32K), check file trailer (last 32K) and delete, we downloaded this output for further visualization by tableau.

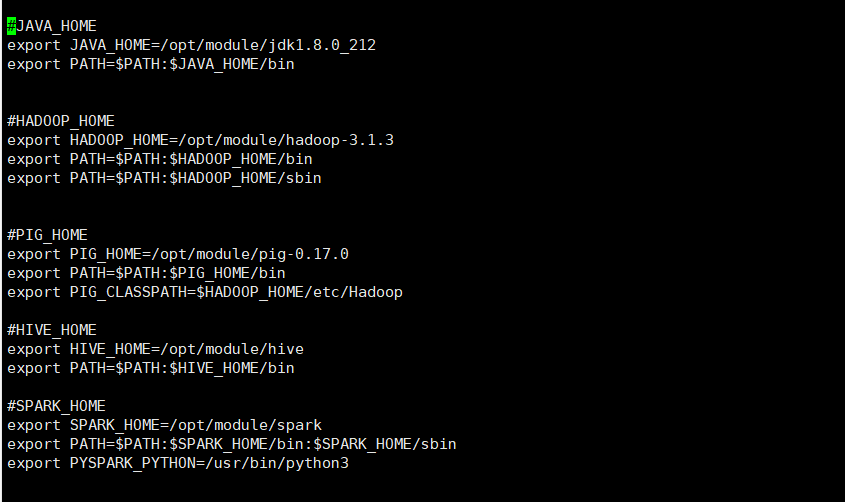
# Data Analysis By Hadoop, Pig , Spark and Hive

**Install required software**:

* Hadoop.3.1.3
* Hive
* Pig
* Spark
* Python3



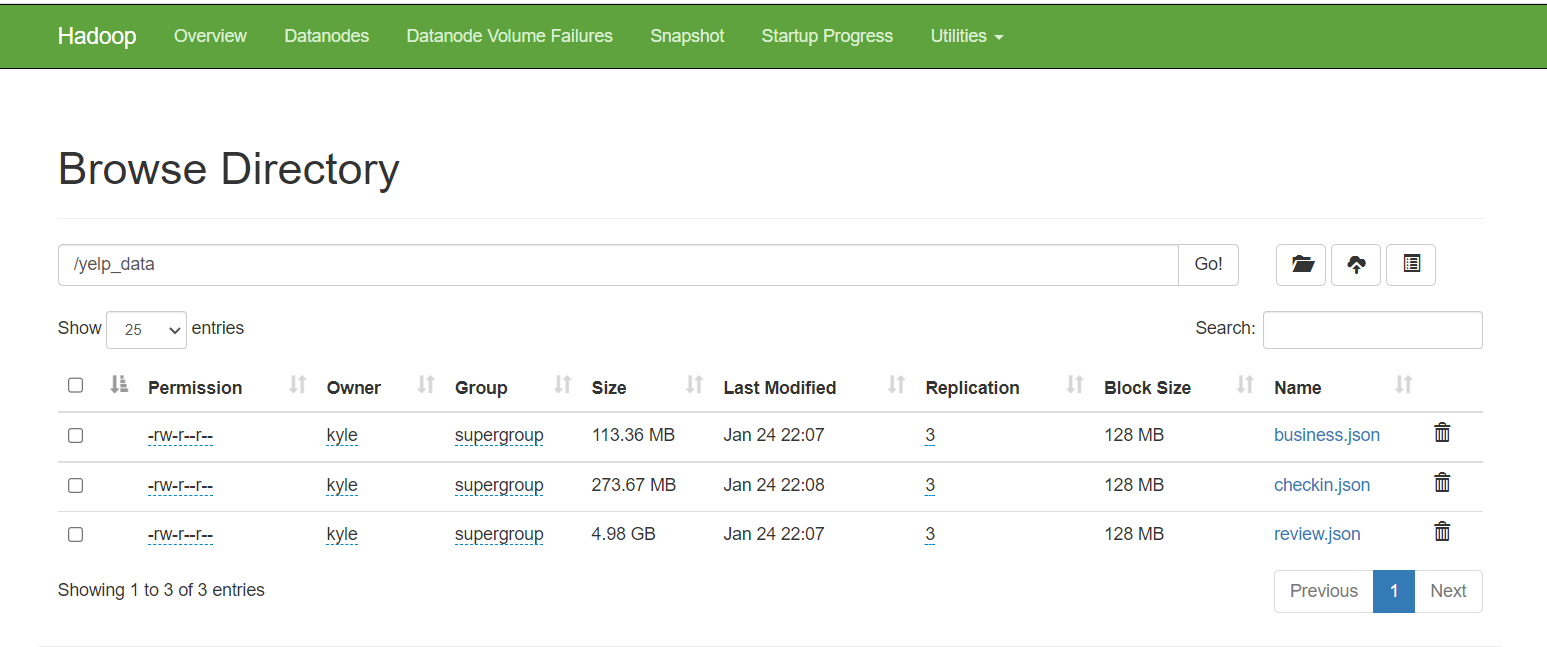
**Set environment variables**



Add java, hadoop, pig (Shrivastava,2022), hive (Kaplarevic,2021), spark (Tucakov,2022) environment variables in the linux system, so that we can run the program smoothly

**Ingest dataset into HDFS**

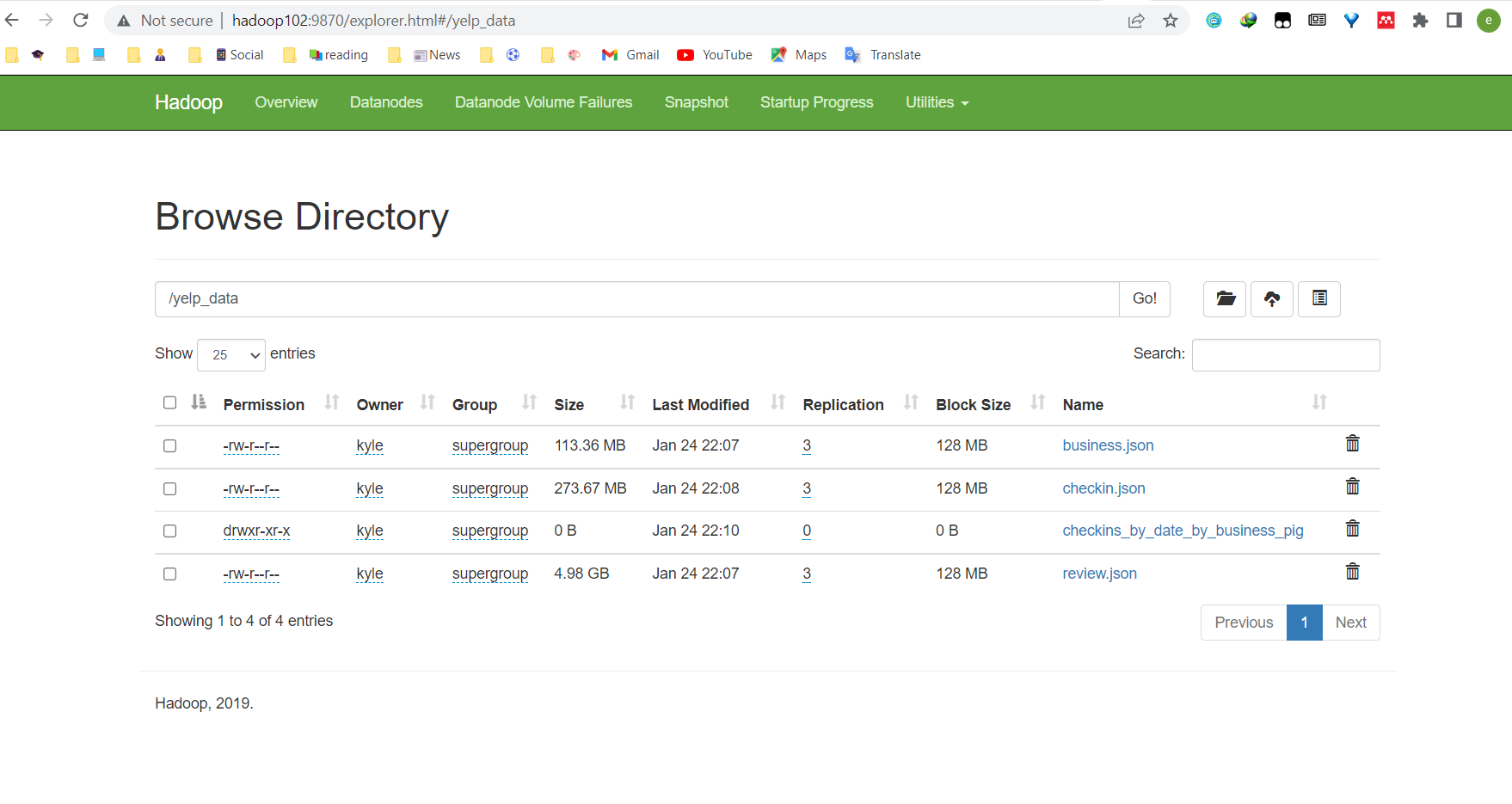
| **# create directory to store the input files**  hadoop fs -mkdir /data  **# Import input files in HDFS**  hadoop fs -put review.json /data/  hadoop fs -put business.json /data/  hadoop fs -put checkin.json /data/ |
| --- |

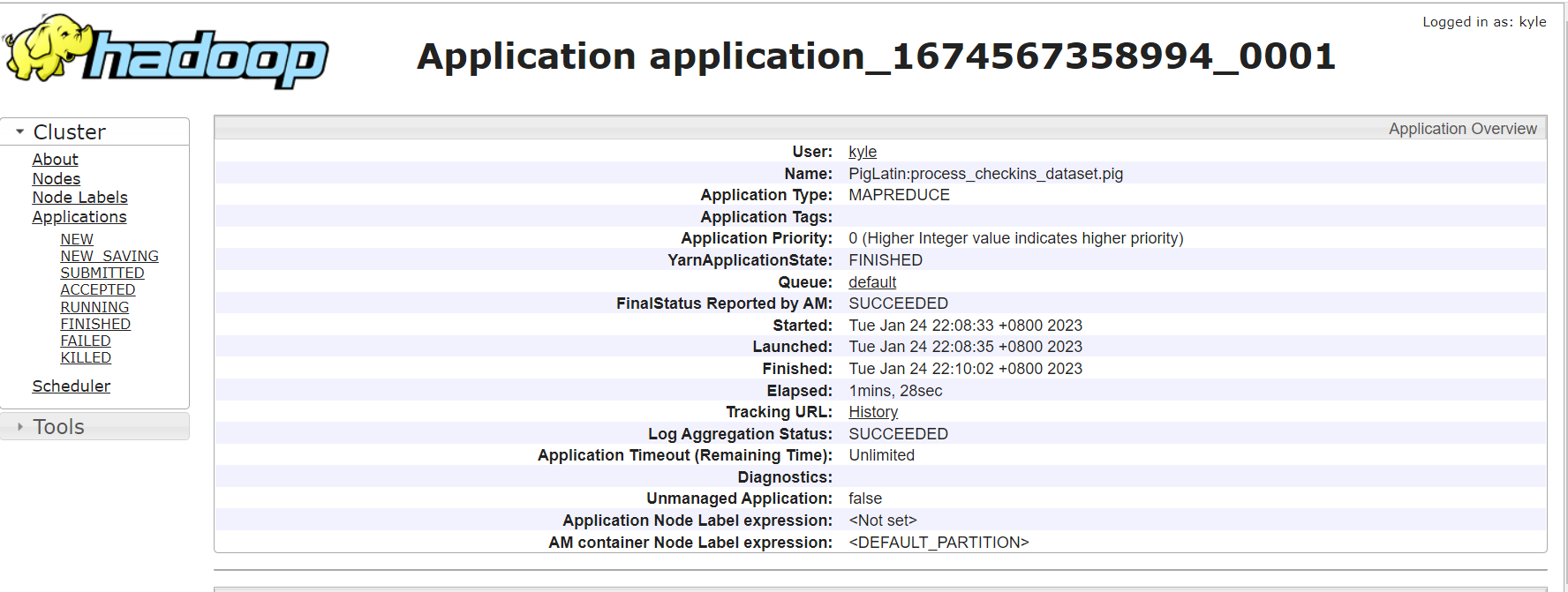
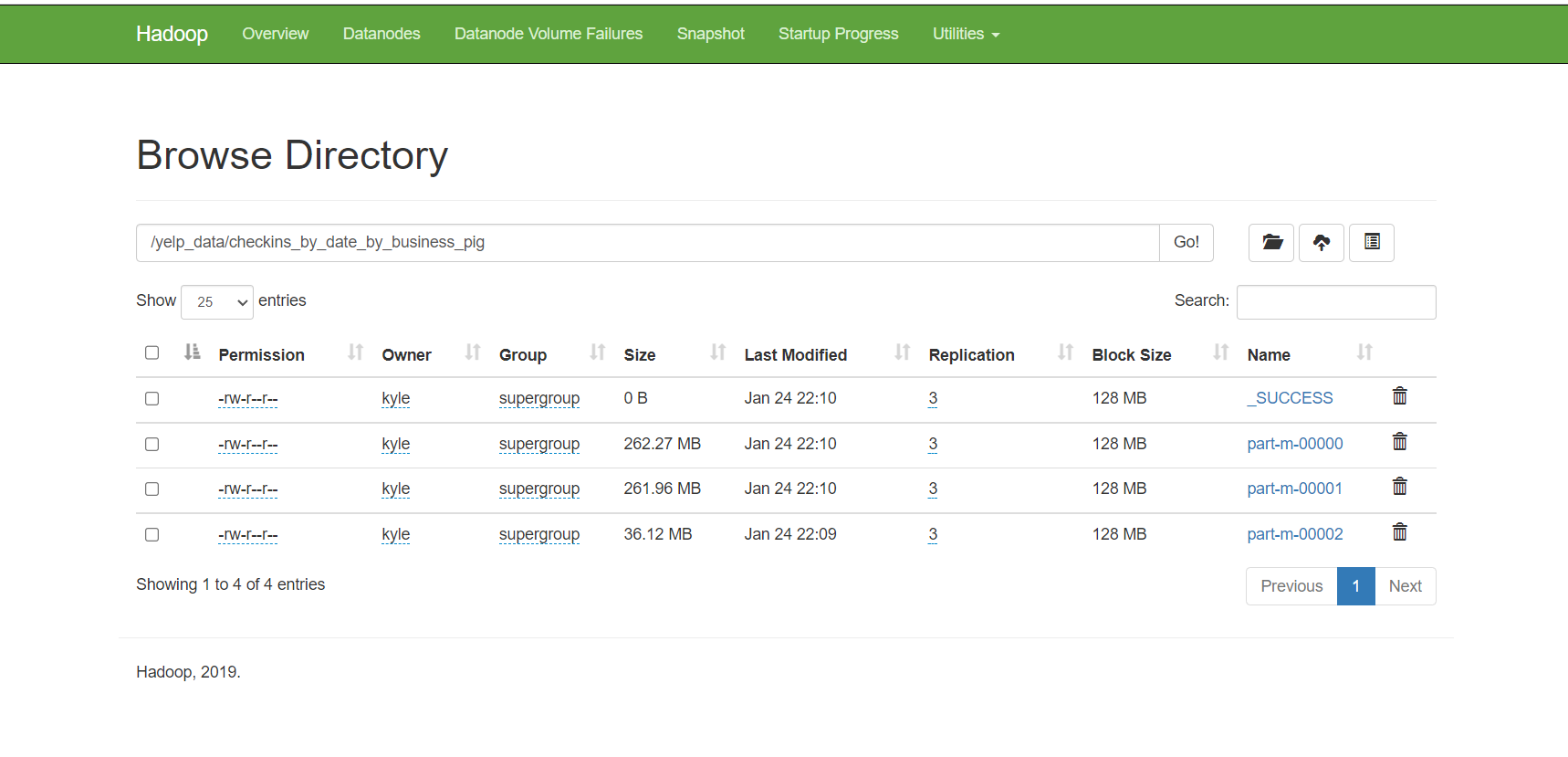


After the data is successfully imported into hadoop, the HDFS Web UI displays business.jon, checkin.json, and review.json files

**Pig Script**

| **#Executing pig script for exploding checkins data into individual rows**  pig -x mapreduce process\_checkins\_dataset.pig > output.msg 2> output.err |
| --- |

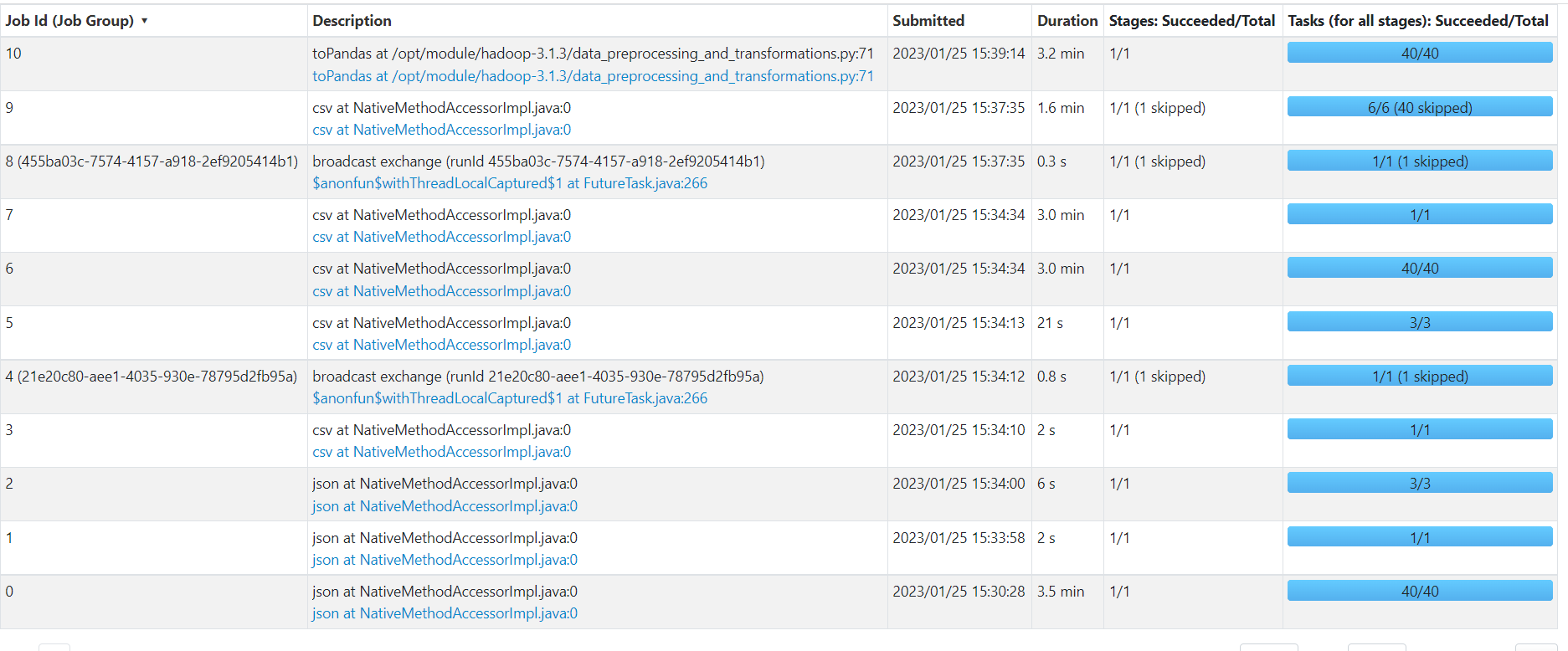
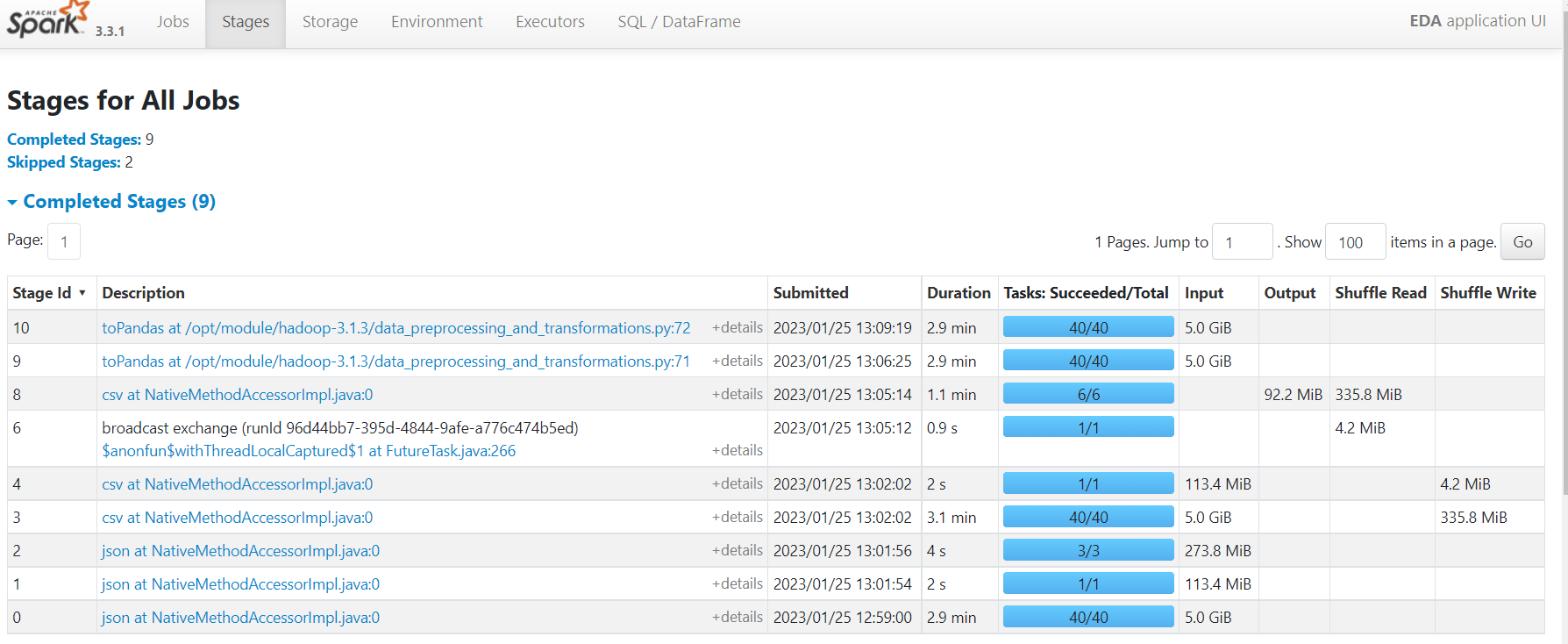
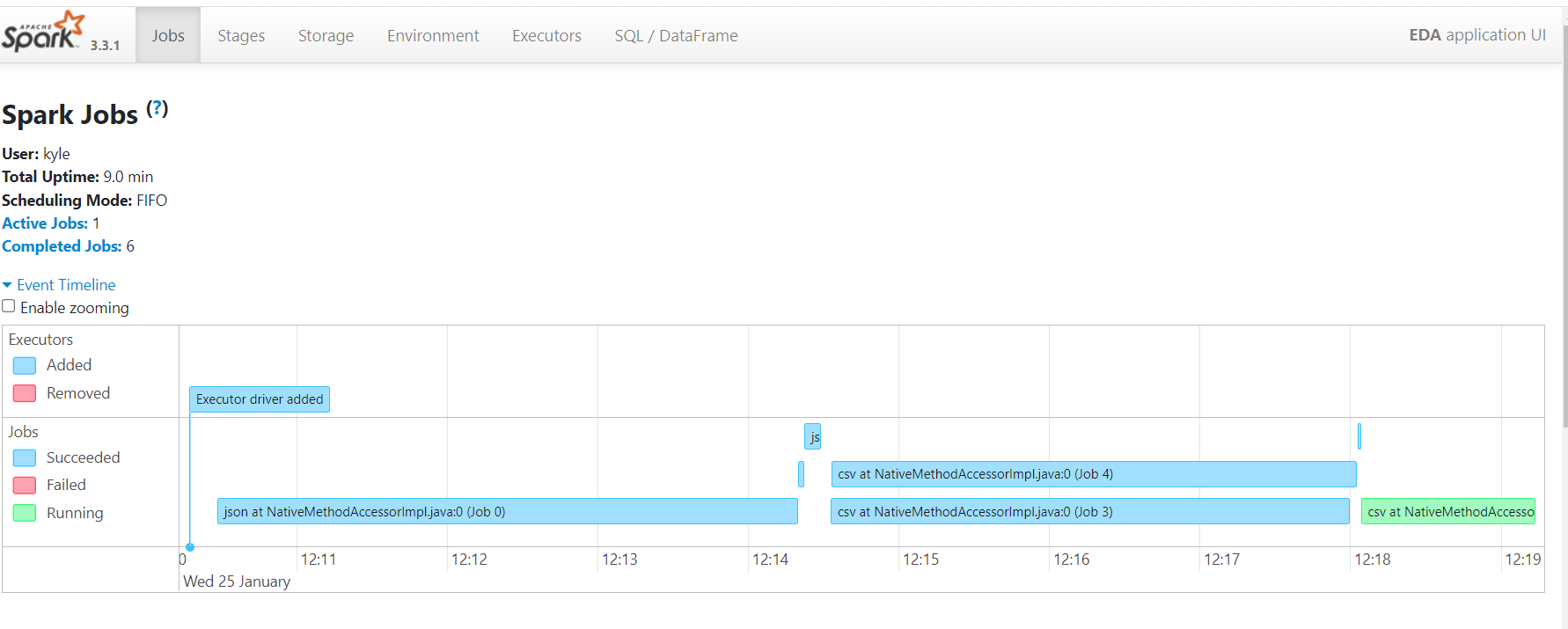
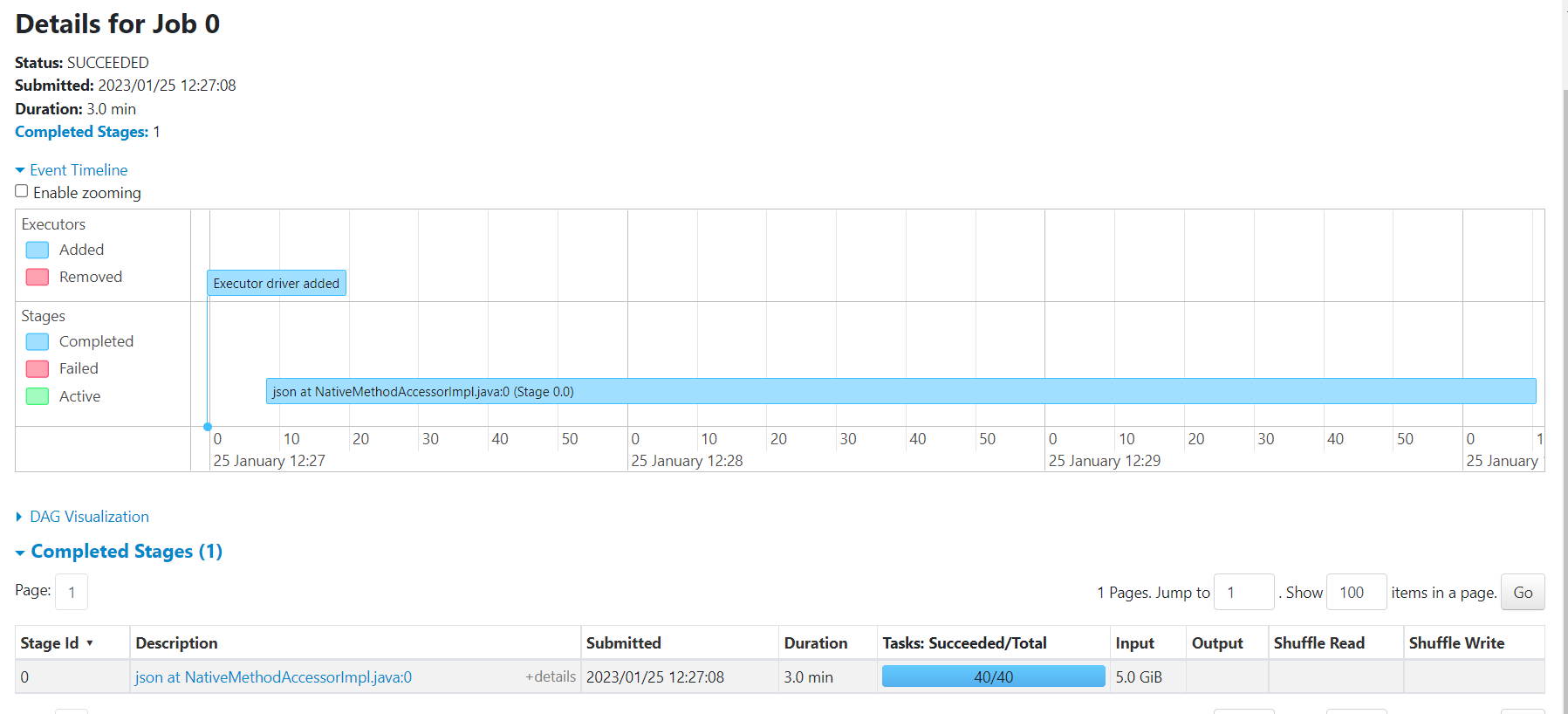




**Spark Script**

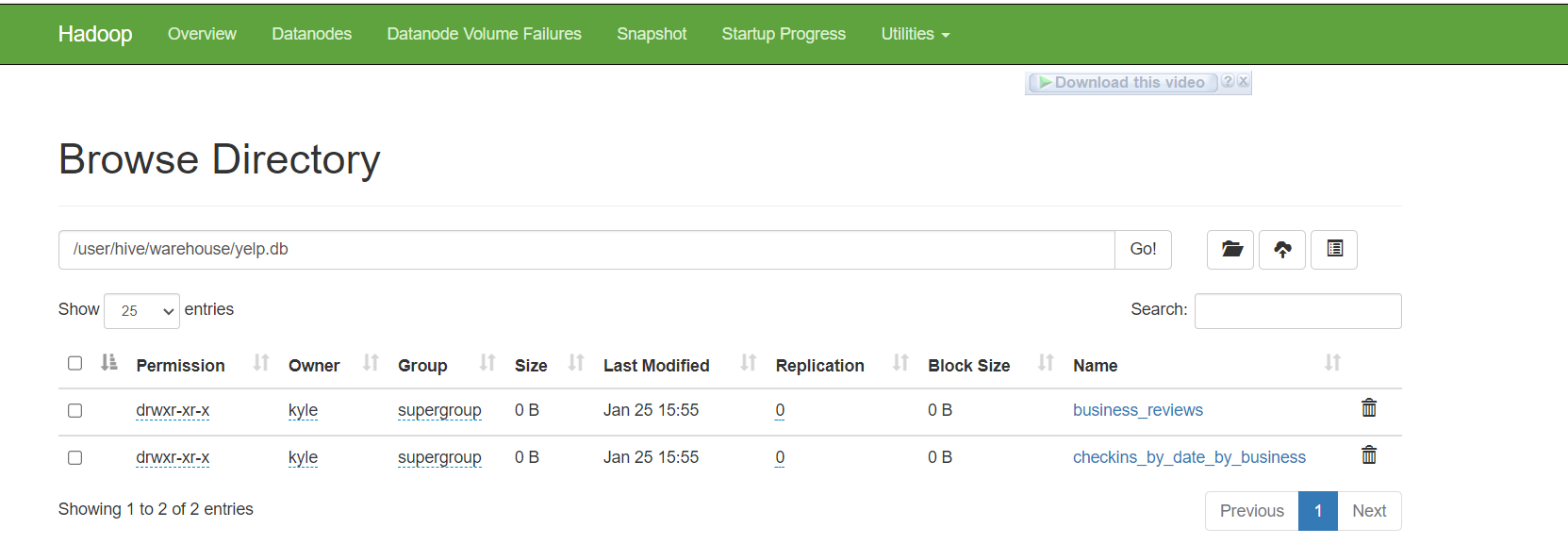
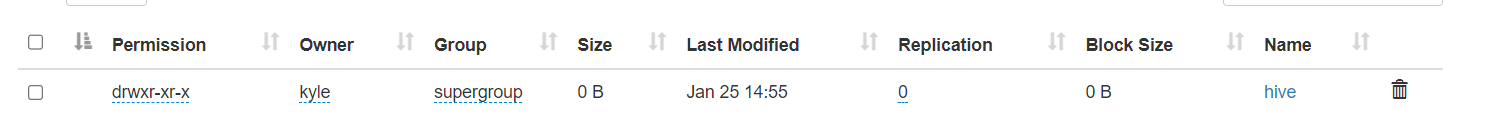
| **#Execute the pyspark script to transform the reviews and business data**  spark-submit data\_preprocessing\_and\_transformations.py |
| --- |

**Spark Task Web UI**



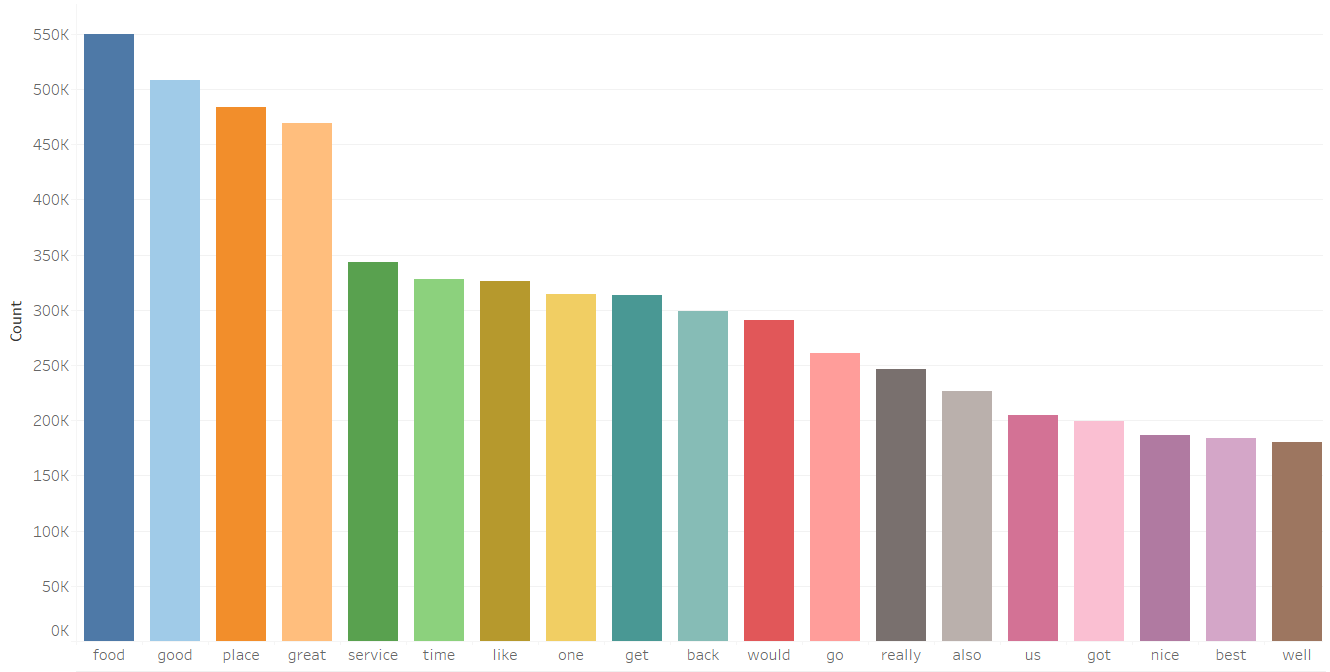
**Hive Tasks with final Output**

| **#Execute hive scripts to load the transformed data from hdfs to a hive table**  hive -f hive\_ddl.hql |
| --- |



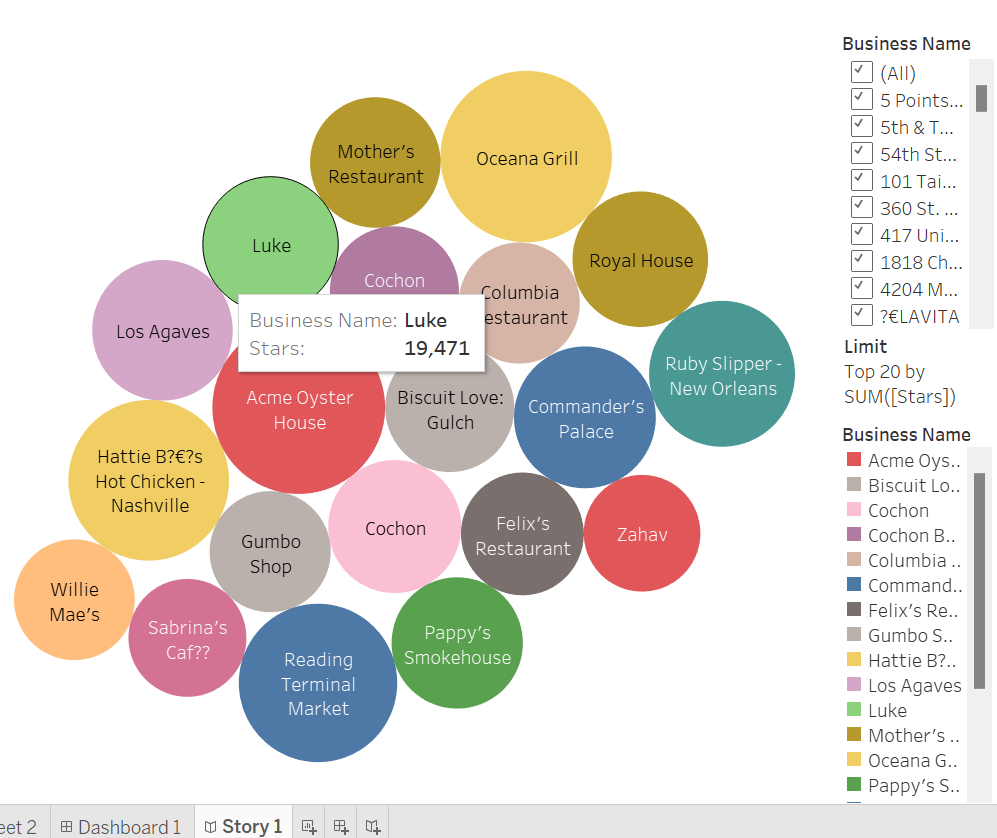
# Visualization By Tableau

* 1. **Word count task visualization**

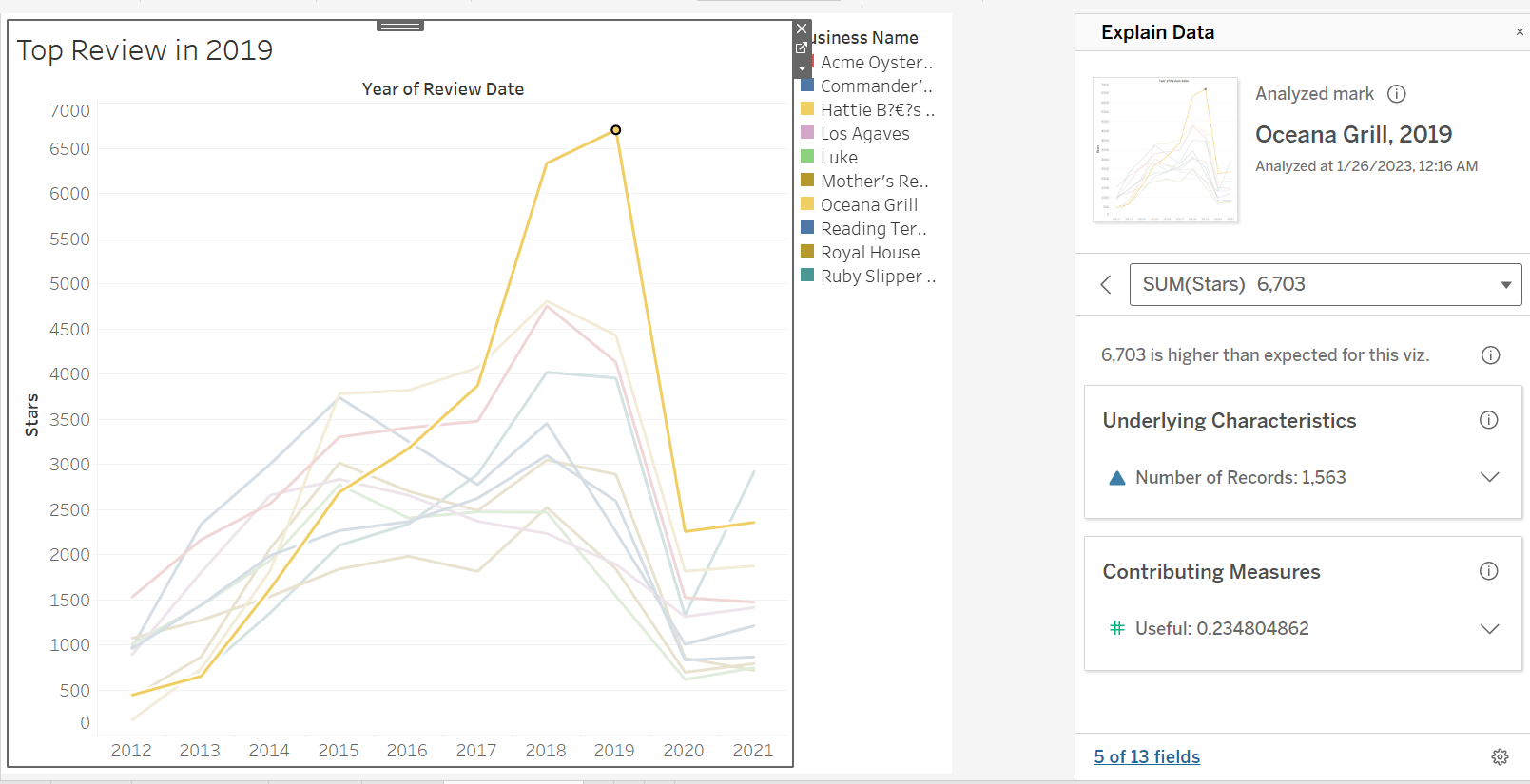
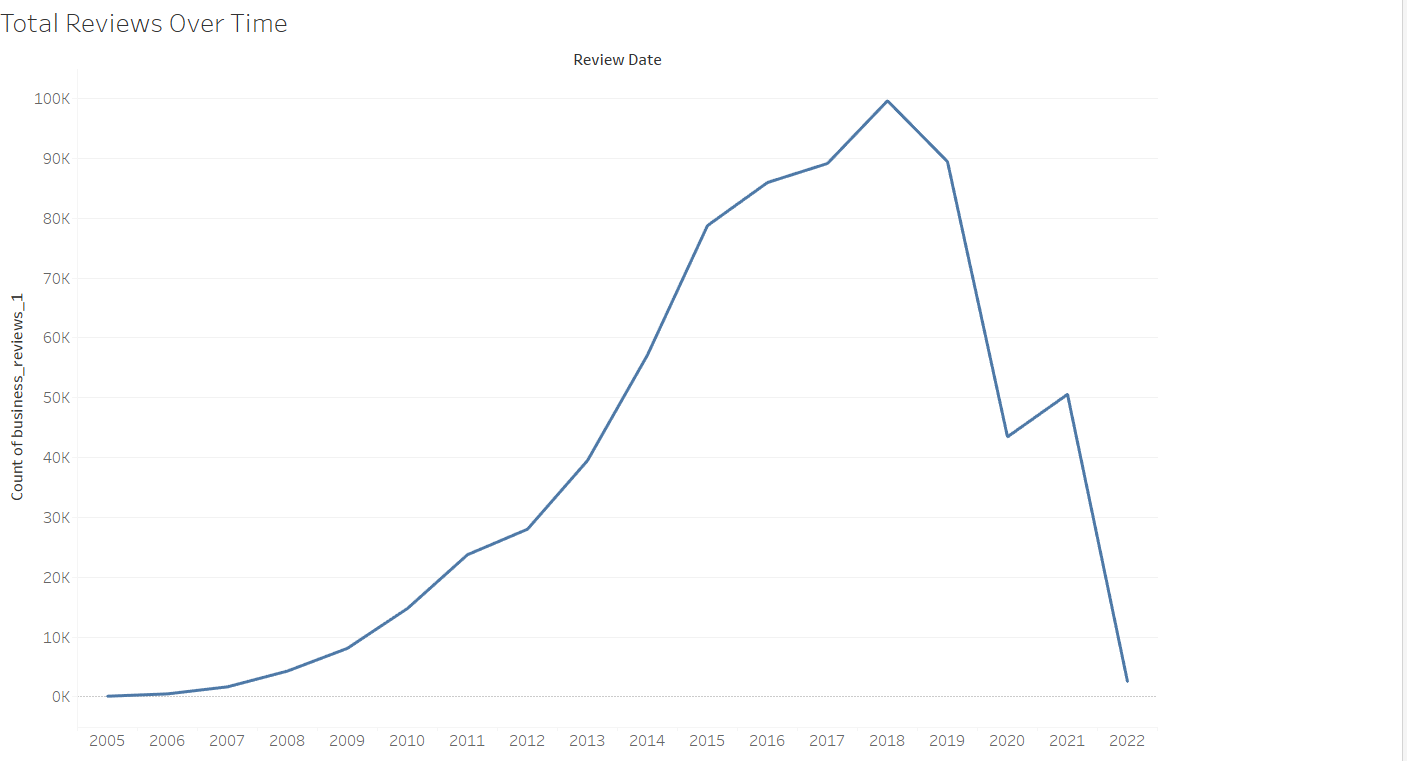
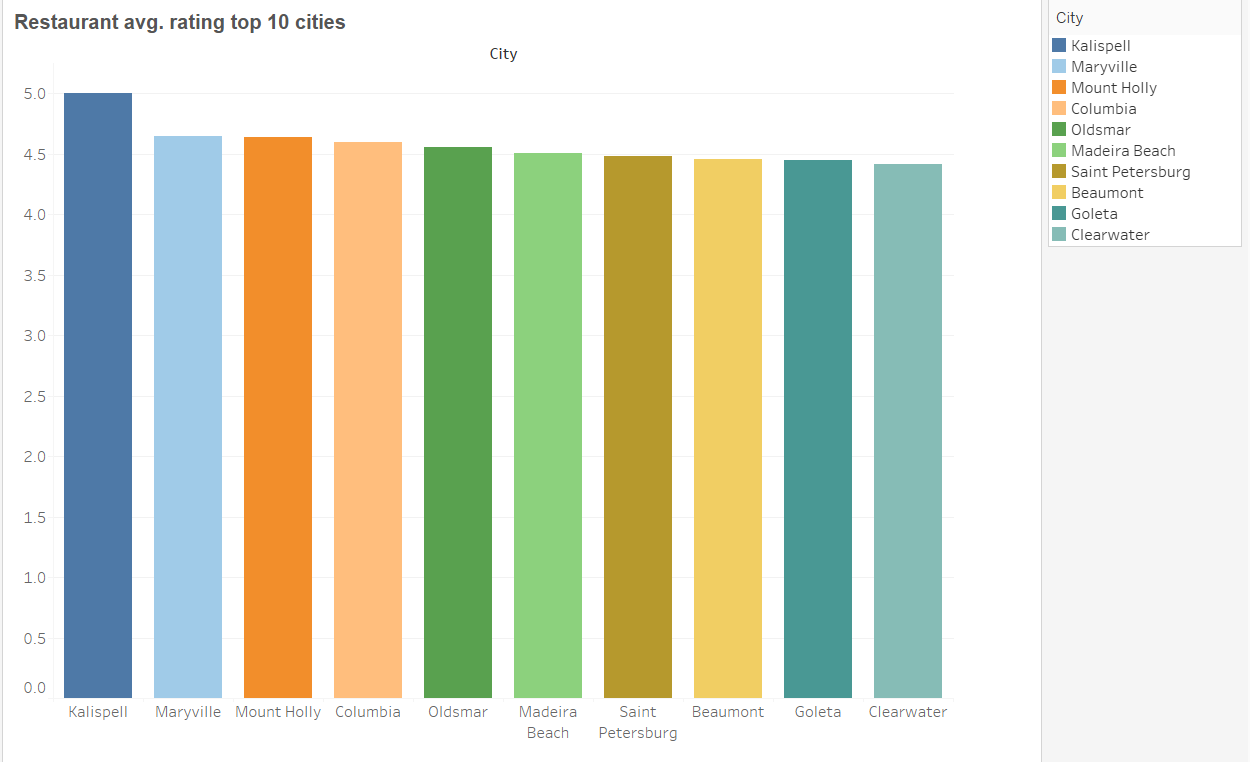
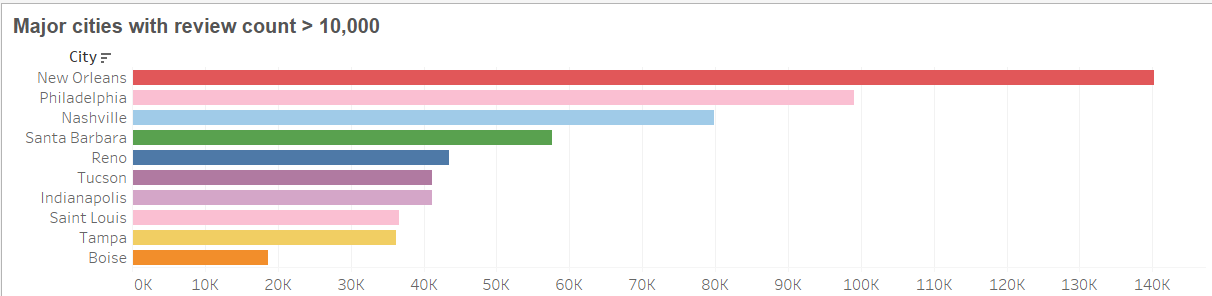
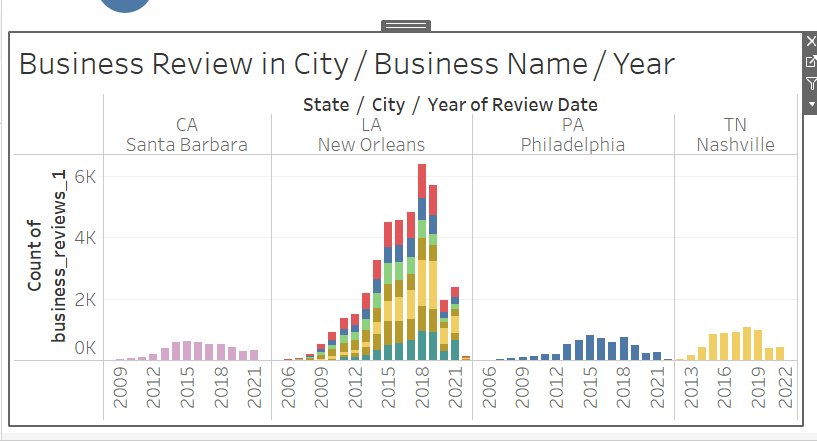
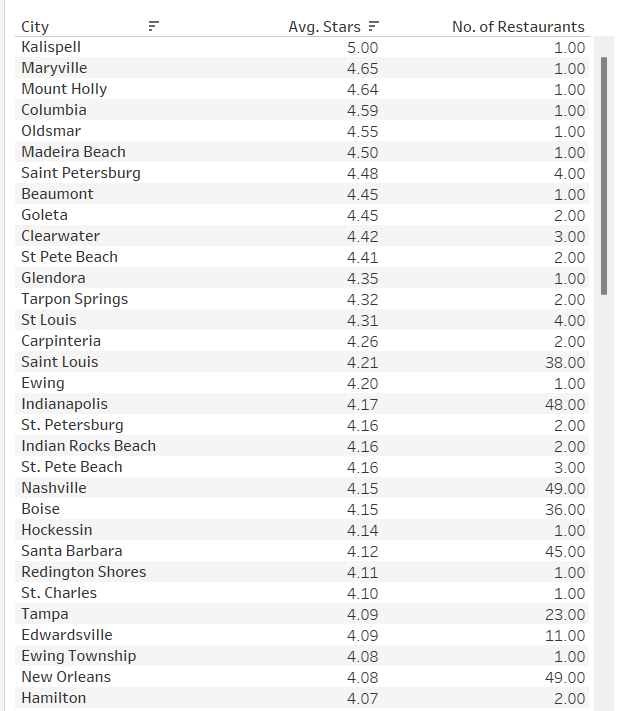
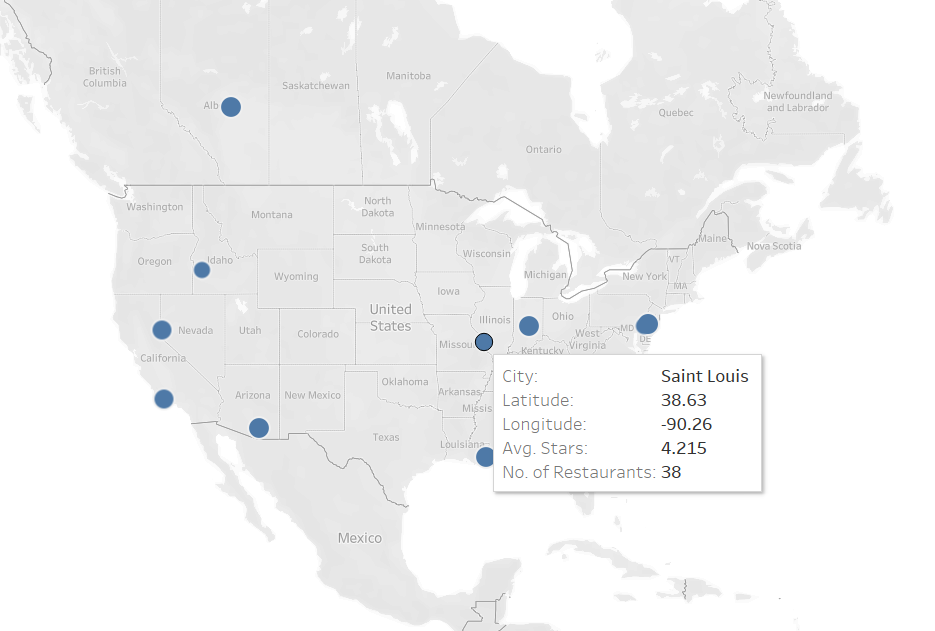
****

We have found which are the Top 20 words with the most number of occurrences in yelp review dataset

**3.8. Data Analysis task visualization**



From the bubble chart, we have found which cities have good reviews. Most good reviews have been found on the basis of the number of stars they’ve written.



# Results and Discussion

This section will discuss the findings from the visualization made via Tableau.

In the word count task, the top 10 words with the most number of occurrences in yelp review dataset are food, good, place, great, service, time, like, one, get and back, and food appears more than 549,000 times at most.

Next is the data analysis task , the bubble charts were used to represent and highlight correlations between numerical variables. The first graph shows the top restaurants with useful reviews and a business named Luke had the highest value with over 19000 cumulative stars. Next, New Orlean - LA and Philadelphia - PA are the top 2 cities/states with the highest reviews and ratings.

In 2020 sharp drops in rating and reviews due to Covid-19 lockdown in the United States. Restaurants closed and people were not going to shop often. According to (Sedov, 2022) during 2020, the first year of the COVID-19 pandemic, restaurants suffered from reduced consumer traffic due to multiple factors: lockdowns, operations restrictions and social distancing.

* New Orlean - LA and Philadelphia - PA are top 2 city/states with highest reviews ( Common criteria : Big metropolitan cities in United States)
* Ocena Grill (Philadelphia-PA) contributes highest spike in customer review
* In accumulated customer reviews from 2005-2022 Acme Oyster House (LA) - contain huge reviews among all Restaurants.

# Challenges

The challenges faced can be broken down into several categories.

* Field of research:
  + Deciding on a suitable use case/domain that fit the scope of the project
  + Finding a suitable sized dataset
* System setup:
  + AWS account termination, which led to the need to use a different account
  + AWS disconnection/timeout on Lab
  + Hadoop, Pig, Spark, Hive configuration
  + Hive SQL query, Schema setup
  + Deciding on appropriate visualizations
* Soft skills:
  + Communication
  + Time management

The challenges were primarily overcome with extensive research, working on the system design and building from scratch with improved understanding of the subject matter, and working together with designated team members.

# Conclusion

With the given analysis in this project, we are able to observe how often words appear in reviews when customers write reviews for restaurants, how good reviews are for a city, which city has the most reviews, which city has the highest restaurant star rating, in our analysis, we learned that Yelp got the largest number of reviews in 2020, after that, due to the pandemic, the total number of reviews dropped sharply and reached the lowest number in 2022, from the bubble chart, we found which cities had most positive reviews, in the histogram we list the top 10 cities with the highest average star rating in restaurant reviews.

In conventional data analysis, manipulating huge datasets over 9 GB files (Yelp, I, 2022)seems like an impossible task, even importing excel files with over 1 million rows is impossible, however, through various big data technologies, we have really learned and understood the charm and magic of distributed systems. Performance in extracting out analysed large dataset in traditional Relational Database Management Systems (RDMS) like Mysql, will hit with bottleneck, according to (Sultana, 2015) author conduct and experimental study to find performance comparison analysing Big Data dataset between Distributed File System (Hadoop) vs RDBMS (Mysql) had concluded that it easier to get data in a traditional database than out. As the field of Big Data matures and grows the need to process larger and larger amounts of data in a timely manner will continue to be a concern. The Exascale computing architecture offers a promising and cost-effective platform to address that concern. Distributed file systems such as Hadoop offer a relatively simple means of taking advantage of the parallel processing required within distributed file systems.

In just a few minutes, the hdfs mapreduce system can not only help us process files exceeding 4GB, but can even do data cleaning, sql query, table building and data analysis.With tableau, we visualize our data output with engaging and intuitive charts to show different information for different business views.

# References

Alamoudi, E. S., & Azwari, S. A. (2021). Exploratory Data Analysis and Data Mining on Yelp Restaurant Review. *2021 National Computing Colleges Conference (NCCC)*. <https://doi.org/10.1109/nccc49330.2021.9428850>

Sedov, D. (2022). Restaurant closures during the COVID-19 pandemic: A descriptive analysis. Economics Letters, 110380. <https://doi.org/10.1016/j.econlet.2022.110380>

Sultana, A. (2015). Using Hadoop to Support Big Data Analysis: Design and Performance Characteristics Recommended Citation. https://repository.stcloudstate.edu/cgi/viewcontent.cgi?article=1042&context=msia\_etds

Shrivastava, D. V. K. (2022, July 1). *How to install Apache pig on a Hadoop cluster on ubuntu*. Medium. Retrieved January 26, 2023, from https://aws.plainenglish.io/how-to-install-apache-pig-on-a-hadoop-cluster-on-ubuntu-948412465948

Kaplarevic, V. (2021, August 23). *How to install Apache Hive on ubuntu {step-by-step guide}*. Knowledge Base by phoenixNAP. Retrieved January 26, 2023, from https://phoenixnap.com/kb/install-hive-on-ubuntu

Tucakov, D. (2022, February 17). *How to install spark on ubuntu {instructional guide}*. Knowledge Base by phoenixNAP. Retrieved January 26, 2023, from <https://phoenixnap.com/kb/install-spark-on-ubuntu>

Yelp Dataset. (n.d.). Www.yelp.com. https://www.yelp.com/dataset/download

Yelp, I. (2022, March 17). *Yelp dataset*. Kaggle. Retrieved January 26, 2023, from https://www.kaggle.com/datasets/yelp-dataset/yelp-dataset?resource=download

Zinoviev, D. (2021). Yelp Restaurant Reviews as Star Rating Predictors.

*Academia Letters*. <https://doi.org/10.20935/al998>

‌

# Appendix

# **Pig Script**

| checkin = LOAD '/data/checkin.json' USING JsonLoader('business\_id:chararray, date:chararray');  -- split by ',' and create a row so we can dereference it later  splt = foreach checkin generate business\_id, FLATTEN(STRSPLIT(date, '\\,')) ;  -- first column is business\_id, rest is converted into a bag and flatten it to make rows  id\_vals = foreach splt generate $0 as business\_id, FLATTEN(TOBAG(\*)) as value;  -- there will be records with (business\_id, business\_id), but id should not have ':'  id\_vals = foreach id\_vals generate business\_id, INDEXOF(value, ':') as p, STRSPLIT(value, ':', 2) as vals;  final = foreach (filter id\_vals by p != -1) generate business\_id, FLATTEN(vals) as (col, val);  -- Write final output to csv  store final into '/data/checkins\_by\_date\_by\_business\_pig' using PigStorage(','); |
| --- |

* 1. **Spark Script**

| #!/usr/bin/env python  # coding: utf-8  # Import libraries required to find pyspark installations on jupyter notebook  import findspark  findspark.init()  findspark.find()  import pyspark  # Import the required pyspark libraries that will help us explode the list of checkin dates for each business  # into individual rows  from pyspark.sql.functions import col, split  from pyspark.sql.functions import explode  # Import SparkSession from Pyspark  from pyspark.sql import SparkSession  # Create spark object with the necessary configuration  spark = SparkSession.builder .appName('EDA') .master('local') .enableHiveSupport() .getOrCreate()  # Read all the input files (stored on HDFS in JSON format) and create a spark dataframe on top of it  review = spark.read.json('hdfs://NameNode:9000/data/review.json')  business = spark.read.json('hdfs://NameNode:9000/data/business.json')  checkin = spark.read.json('hdfs://NameNode:9000/data/checkin.json')  # create a temporary view on top of each dataframe for querying using Spark SQL  review.createOrReplaceTempView("review")  business.createOrReplaceTempView("business")  checkin.createOrReplaceTempView("checkin")  # Since there are thousands of categories, we will focus our analysis only on categories such as restaurant, pizza & sandwich.  business\_subset = spark.sql("select b.\*, row\_number() over(partition by b.state order by b.review\_count desc) as rnk from business b where (lower(categories) like '%restaurant%' or lower(categories) like '%pizza%' or lower(categories) like '%sandwich%')")  business\_subset.createOrReplaceTempView("business\_subset")  # Join reviews with subset of businesses to get it's required details.  business\_reviews=spark.sql("select r.business\_id, r.review\_id, r.date, r.useful, r.stars, b.city, b.state, b.latitude, b.longitude, b.name, b.postal\_code from review r inner join (select \* from business\_subset b where rnk<=50) b on r.business\_id=b.business\_id")  # Split the string column on the basis of comma to create a list of dates  checkin\_date\_list=checkin.select(col("business\_id"), split(col("date"), ",\s\*").alias("date"))  # Converting the array of dates into individual rows  checkin\_explode=checkin\_date\_list.withColumn("date", explode(checkin\_date\_list.date))  # Create a temporary view for querying data in Spark SQL  checkin\_explode.createOrReplaceTempView("checkin\_explode")  # Filter data for subset of categories and locations as we did in above steps  checkin\_explode\_subset=spark.sql("select c.business\_id, c.date from checkin\_explode c inner join (select \* from business\_subset b where rnk<=50) b on c.business\_id=b.business\_id")  # Write spark datframe to HDFS  #Chekins transformation was done in Pig  #checkin\_explode\_subset.write.csv('/data/checkins\_by\_date\_by\_business')  business\_reviews.write.csv('/data/business\_reviews')  reviews\_bad\_review=spark.sql("select r.text from review r where business\_id='DkYS3arLOhA8si5uUEmHOw' and stars=1 order by useful desc limit 100")  reviews\_good\_review=spark.sql("select r.text from review r where business\_id='DkYS3arLOhA8si5uUEmHOw' and stars=5 order by useful desc limit 100")  reviews\_bad\_review\_pd=reviews\_bad\_review.toPandas()  reviews\_good\_review\_pd=reviews\_good\_review.toPandas()  # Let's make some word clouds!  # Terminal / Anaconda Prompt: conda install -c conda-forge wordcloud  from wordcloud import WordCloud  import matplotlib.pyplot as plt  from sklearn.feature\_extraction import text  stop\_words = text.ENGLISH\_STOP\_WORDS.union(['good','went','did','didn''t','said','it','pizza','slice','told','order','ordered','food','just','I''m','going','like','really','asked','place','time','got','called','came'])  wc = WordCloud(stopwords=stop\_words, background\_color="white", colormap="Dark2",  max\_font\_size=150, random\_state=42)  import json  import csv  import re  import string  import pandas as pd  def clean\_text(text):  '''Make text lowercase, remove text in square brackets, remove punctuation and remove words containing numbers.'''  text = text.lower()  text = re.sub('\[.\*?\]', '', text)  text = re.sub('[%s]' % re.escape(string.punctuation), '', text)  text = re.sub('\w\*\d\w\*', '', text)  '''Get rid of some additional punctuation and non-sensical text that was missed the first time around.'''  text = re.sub('[‘’“”…]', '', text)  text = re.sub('\n', '', text)  return text  reviews\_bad\_review\_pd['text'] = reviews\_bad\_review\_pd['text'].apply(lambda x: clean\_text(x))  reviews\_good\_review\_pd['text'] = reviews\_good\_review\_pd['text'].apply(lambda x: clean\_text(x))  wc.generate(' '.join(reviews\_bad\_review\_pd['text']))  plt.imshow(wc, interpolation='bilinear')  plt.axis("off")  plt.show()  plt.savefig("/home/hdoop/yelp/bad\_reviews\_word\_cloud2.png")  wc.generate(' '.join(reviews\_good\_review\_pd['text']))  plt.imshow(wc, interpolation='bilinear')  plt.axis("off")  plt.show()  plt.savefig("/home/hdoop/yelp/good\_reviews\_word\_cloud2.png") |
| --- |

* 1. **Hive Script**

| --Create Database called as yelp  create database yelp;  --Set current database as yelp  use yelp;  drop table yelp.business\_reviews;  --Create table for storing business reviews  CREATE TABLE yelp.business\_reviews  (  business\_id varchar(100),  review\_id varchar(100),  review\_date timestamp,  useful int,  stars float,  city varchar(100),  state varchar(100),  latitude float,  longitude float,  business\_name varchar(50),  postal\_code varchar(5)  )  ROW FORMAT DELIMITED  FIELDS TERMINATED BY ','  STORED AS TEXTFILE;  --Load the data from HDFS into the created hive table  load data inpath '/home/hdoop/scripts/shell/data/business\_reviews/\*.csv' into table yelp.business\_reviews;  --select \* from yelp.business\_reviews limit 10;  --Create table for storing checkins by date for each business  CREATE TABLE yelp.checkins\_by\_date\_by\_business  (  business\_id varchar(100),  checkin\_date timestamp  )  ROW FORMAT DELIMITED  FIELDS TERMINATED BY ','  STORED AS TEXTFILE;  --Load the data from HDFS into the created hive table  load data inpath '/home/hdoop/scripts/shell/data/checkins\_by\_date\_by\_business/part\*' into table yelp.checkins\_by\_date\_by\_business;  --select \* from yelp.checkins\_by\_date\_by\_business limit 10; |
| --- |

* 1. **AWS LAB Details:**

**voclabs/user2255056=RAJASEGARAN\_A\_L\_M\_SIVAANANDAN\_\_**

**Account: 776079374440**

