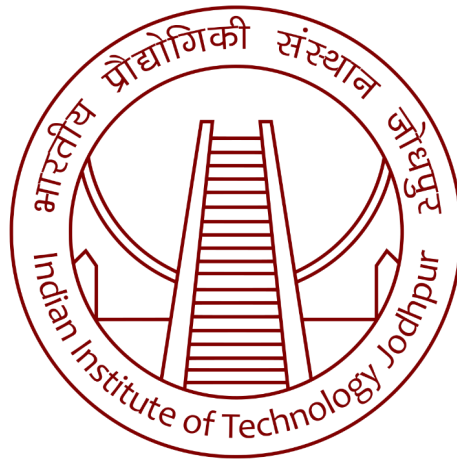


## **Project Report**

**Course Name: Software and Data Engineering (SDE)**

**Course Code: CSL7090**

**Indian Institute of Technology, Jodhpur**



॥ त्वं ज्ञानमयो विज्ञानमयोऽसि ॥

**Project Title:** Exploring BigData: CDH Platform Architecture, EDA on Local Machine

**Submitted To: -**

Dr. Sumit Kalra,  
Assistant Professor,  
Dept. of CSE,  
IIT Jodhpur.

**Submitted By: -**

Bhawna Bhoria (M22MA003)  
Jash Patel (M22CS061)

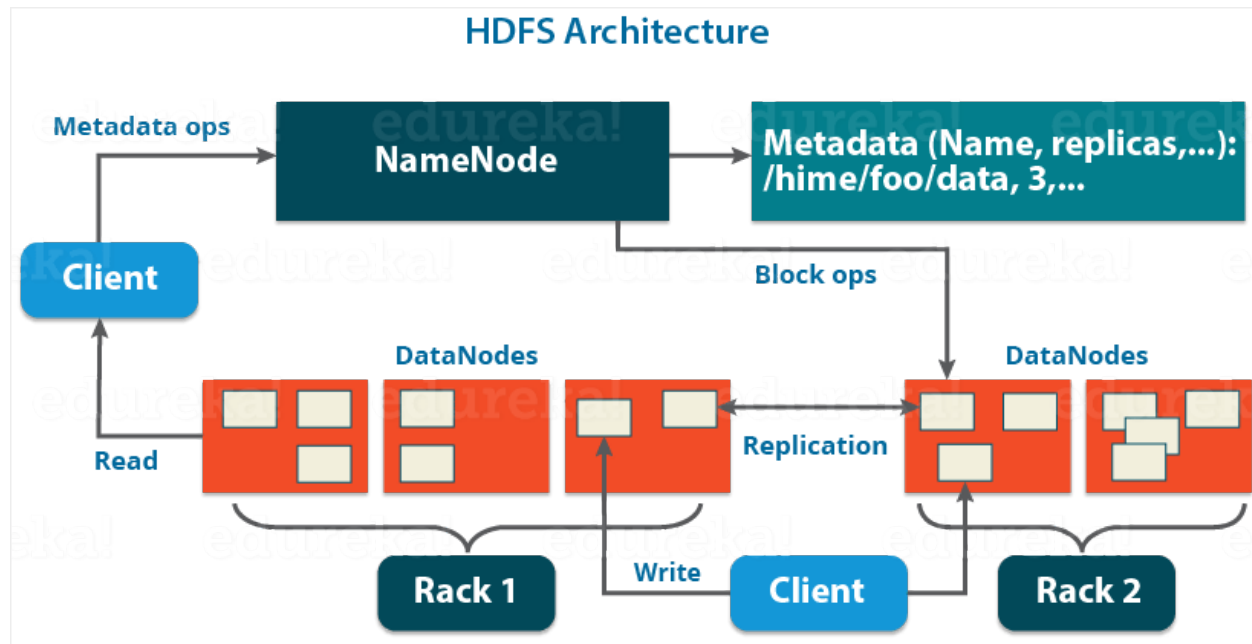
## **Abstract**

As big data and associated technologies continue to advance and gain widespread acceptance, the demand for a comprehensive big data ecosystem encompassing data resources, APIs, open-source platforms, data infrastructure, and components for big data analysis and applications has become increasingly significant. This project delves into the utilization of the big data ecosystem specifically centered around the Cloudera Distribution for Hadoop (CDH), the recent trends in data scale development and the current state-of-the-art advancements in big data analysis systems. It then shifts focus to the technical architecture of the big data ecosystem built on CDH, providing an analysis of the system's hierarchical structure. Users can leverage its resources for data exploration, analysis, and visualization, all within an interactive and user-friendly interface. The platform supports various data science and machine learning libraries, allowing for the seamless integration of advanced analytical techniques into big data workflows.

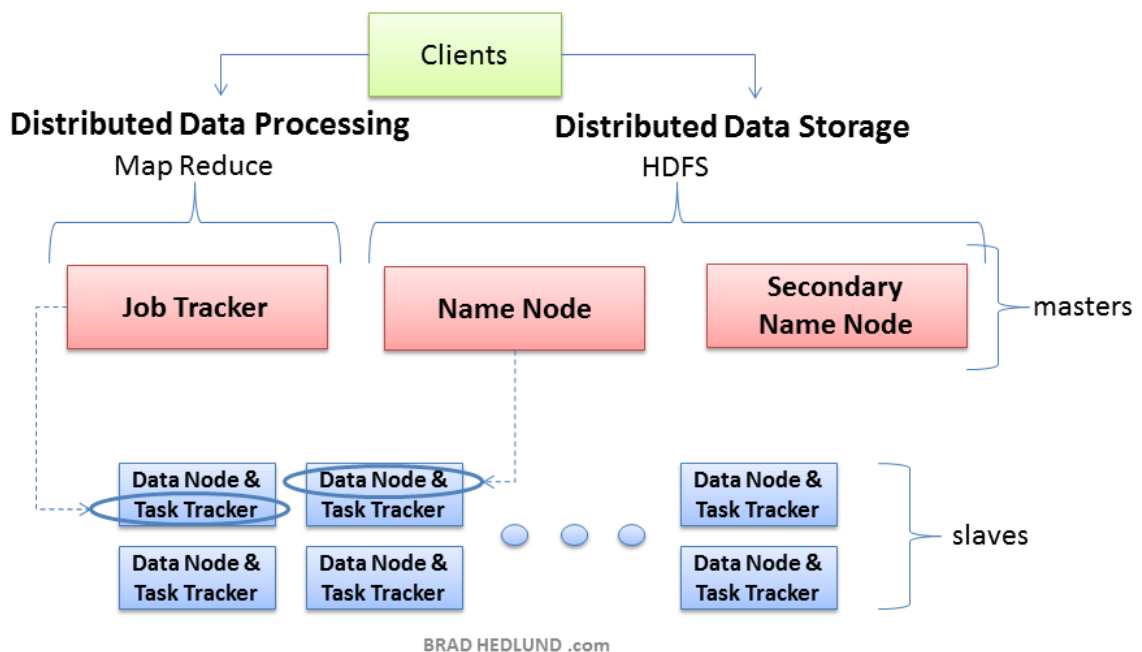
## **Introduction**

Over the past few years, there has been a swift expansion in the volume of data. As reported by the International Data Corporation (IDC), a global information technology consulting firm, the worldwide data storage capacity escalated to 44ZB in 2020 and is projected to potentially reach 2500ZB by 2030. Notably, the quantity of data requiring processing has far surpassed the upper limits of available processing capacity, leading to a significant backlog of data that cannot be collected or processed promptly. The big data ecosystem such as CDH, on-premise resources or local platform comprehend intricate systems and we will look at all these and observe the processing techniques. This project includes two chapters. Chapter-1 contains practical insights into CDH platform architecture and tools, chapter 2 contains exploratory data analysis on local machine platform.

## Platform 1 : CDH Architecture



In the customized distribution, we are using Namenode and YARN Resource Manager, which is “Yet Another Resource Negotiator” to manage and allocate resources within a Hadoop cluster. In more recent versions, the Job Tracker and Task Tracker functionalities have been replaced by the ResourceManager and NodeManagers, respectively, offering a more flexible and scalable resource management framework and that is the version we will be using.



## 1.1 Setting the CDH Cluster

### 1.1.1 Customized Cloudera Distribution for Hadoop Image:-

```
[bhawnabhorla@Bhawnas-MacBook-Pro Scripts % docker images
REPOSITORY          TAG          IMAGE ID        CREATED         SIZE
sde_project          latest       b7be6f417387   13 days ago    7.37GB
cloudera-5-13        latest       35b346d3eb86   4 weeks ago    7GB
cloudera/quickstart  latest       4239cd2958c6   7 years ago    6.34GB
```

Mapping the required ports in order to run the CDH Ecosystem.

```
bhawnabhorla@Bhawnas-MacBook-Pro Scripts % docker run --hostname=quickstart.cloudera --privileged=true -t -
i -p 8888:8888 -p 10000:10000 -p 10020:10020 -p 11000:11000 -p 18080:18080 -p 18081:18081 -p 18088:18088 -p
19888:19888 -p 21000:21000 -p 21050:21050 -p 2181:2181 -p 25000:25000 -p 25010:25010 -p 25020:25020 -p 500
10:50010 -p 50030:50030 -p 50060:50060 -p 50070:50070 -p 50075:50075 -p 50090:50090 -p 60000:60000 -p 60010
:60010 -p 60020:60020 -p 60030:60030 -p 7180:7180 -p 7183:7183 -p 7187:7187 -p 80:80 -p 8020:8020 -p 8032:8
032 -p 802:8042 -p 8088:8081 -p 8983:8983 -p 9083:9083 35b346d3eb86 /usr/bin/docker-quickstart
Starting mysqld: [ OK ]
```

Further installation of the services available on CDH.

```
Started Hadoop httpfs (hadoop-httpfs): [ OK ]
starting historyserver, logging to /var/log/hadoop-mapreduce/mapred-historyserver-quickstart.cloudera.out
Started Hadoop historyserver: [ OK ]
starting nodemanager, logging to /var/log/hadoop-yarn/yarn-yarn-nodemanager-quickstart.cloudera.out
Started Hadoop nodemanager: [ OK ]
starting resourcemanager, logging to /var/log/hadoop-yarn/yarn-yarn-resourcemanager-quickstart.cloudera.out
Started Hadoop resourcemanager: [ OK ]
starting master, logging to /var/log/hbase/hbase-hbase-master-quickstart.cloudera.out
Started HBase master daemon (hbase-master): [ OK ]
starting rest, logging to /var/log/hbase/hbase-hbase-rest-quickstart.cloudera.out
Started HBase rest daemon (hbase-rest): [ OK ]
starting thrift, logging to /var/log/hbase/hbase-hbase-thrift-quickstart.cloudera.out
Started HBase thrift daemon (hbase-thrift): [ OK ]
Starting Hive Metastore (hive-metastore): [ OK ]
Starting Hive Server2 (hive-server2): [ OK ]
Starting Sqoop Server: [ OK ]
Sqoop home directory: /usr/lib/sqoop2
Setting SQOOP_HTTP_PORT: 12000
Setting SQOOP_JMX_PORT: 12001
```

jps output :-

```
[[root@quickstart /]# jps
880 JournalNode
2000 NodeManager
7292
6163 Bootstrap
2713 HMaster
5165 HistoryServer
5073 Bootstrap
5494 HRegionServer
1673 Bootstrap
674 DataNode
1321 SecondaryNameNode
1081 NameNode
563 QuorumPeerMain
1802 JobHistoryServer
7183 Bootstrap
2281 ResourceManager
9166 Jps
3953 RunJar
3346 ThriftServer
7443
3030 RESTServer
3616 RunJar
```

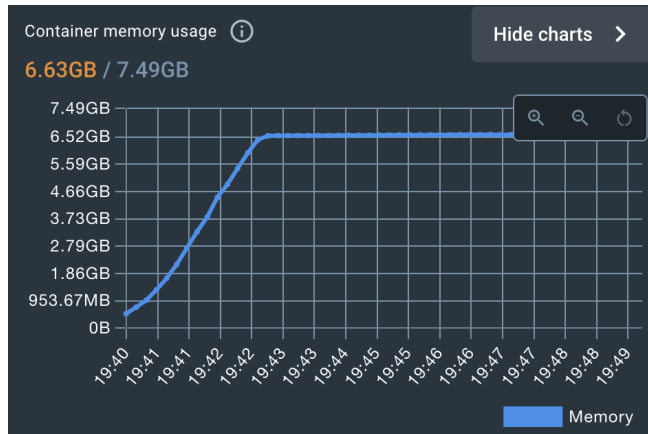
Checking Namenode :-

```
[[root@quickstart /]# hdfs getconf -confKey fs.defaultFS
hdfs://quickstart.cloudera:8020
```

## 1.1.2 Laptop configurations:

Physical Memory on device: 16GB

CDH Distribution Memory Occupance : 8 GB



## 1.2 Running the CDH Ecosystem and checking the Node of the cluster :-

The user interface of CDH is available at port 8888 with user credentials as **cloudera**.

127.0.0.1:8888/filebrowser/

UE Query Editors Data Browsers Workflows Search Security File Browser Job Browser

File Browser

Search for file name Actions Move to trash

Home / user / cloudera

Name	Size	User	Group	Permissions
hdfs		hdfs	supergroup	drwxr-xr-x
.		cloudera	cloudera	drwxr-xr-x

Logged in as: dr.who

### Nodes of the cluster

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCoers Used	VCoers Total	VCoers Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
0	0	0	0	0	0 B	8 GB	0 B	0	8	0	1	0	0	0	0

User Metrics for dr.who

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved	VCoers Used	VCoers Pending	VCoers Reserved
0	0	0	0	0	0	0	0 B	0 B	0 B	0	0	0

Show: 20 entries

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail	VCoers Used	VCoers Avail	Version
/default-rack	RUNNING	quickstart.cloudera:33173	quickstart.cloudera:8042	Wed Nov 22 14:45:38 +0000 2023		0	0 B	8 GB	0	8	2.6.0-cdh5.7.0	

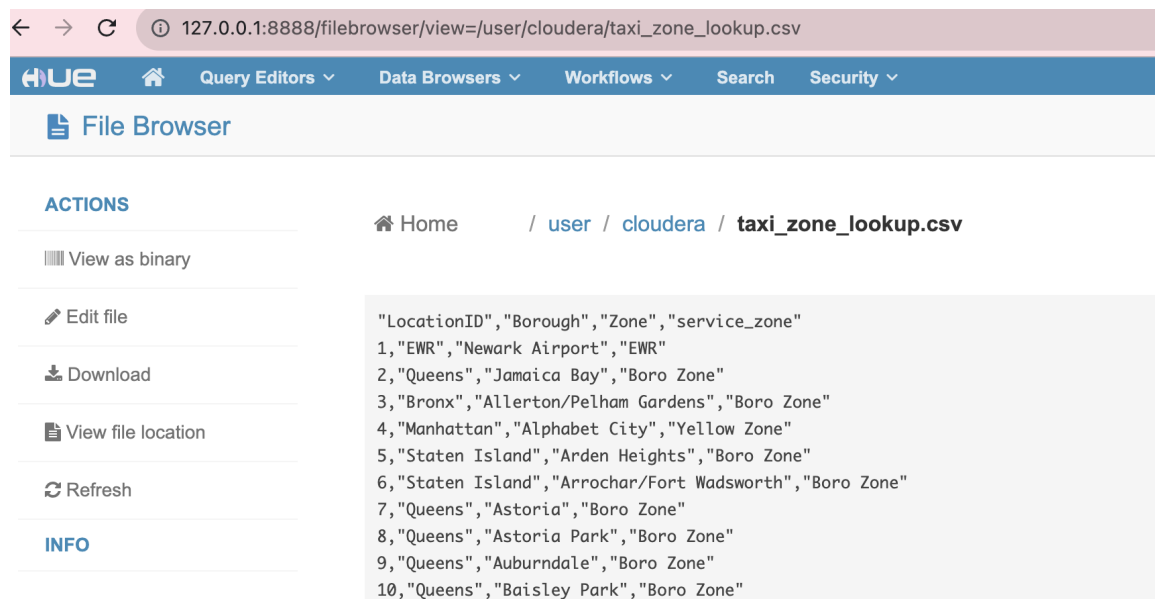
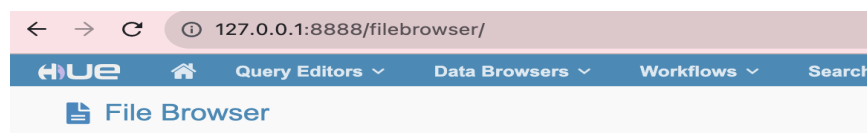
Showing 1 to 1 of 1 entries

First Previous 1 Next Last

The fault-tolerant and distributed characteristics of the Hadoop Distributed File System (HDFS) are well-known. A client requests that a file be uploaded from the NameNode during a write operation. After confirming the file's existence and approving the upload, the NameNode points the client towards a DataNode to find out which server the block ought to be kept on. After that, the client uploads the file to the designated DataNode, sending the matching block to the DataNode server to finish the write request. When performing a read operation, the client applies to the NameNode in order to obtain metadata. The NameNode then locates the DataNode server in the cluster where the file is located and arbitrarily chooses one. There are various tools and technologies' interfaces available in this Ecosystem and we will look at a few of those.

Taking input data of yellow\_taxi\_cab and putting it on hdfs by using the put command:-

```
[root@quickstart ~]#  
[root@quickstart ~]# hdfs dfs -put /user/sde_project/taxi_zone_lookup.csv /user/cloudera/
```



## 1.3 HIVE

Hive, built upon Hadoop, functions as a data warehouse tool capable of transforming, loading, and extracting data. It serves as a mechanism for the storage, retrieval, and analysis of extensive datasets within Hadoop. The tool efficiently processes queries, swiftly generating query results. Similar to MySQL, Hive executes statements [10]. Upon user task submission, the compiler takes charge of the user's task. It retrieves metadata information from the meta-store, compiles the task, selects the optimal strategy, and ultimately delivers the results. The subsequent pseudocode outlines Hive's process for data transformation, loading, and extraction.

Creating a database and table to process the file using Hive :-

The screenshot displays the Hive Editor interface in a web browser. The top navigation bar includes 'Query Editors', 'Data Browsers', 'Workflows', 'Search', and 'Security'. The main content area is divided into a left sidebar and a central query editor.

**Query Editor (Top):** The query editor shows a single query: `1 CREATE DATABASE sde_project`. The left sidebar shows the 'Databases' section with 'default' and 'sde\_project' listed.

**Query Editor (Bottom):** The query editor shows a single query: `1 USE sde_project`. Below the query editor are buttons for 'Execute', 'Save as...', 'Explain', 'Format', and 'New query'.

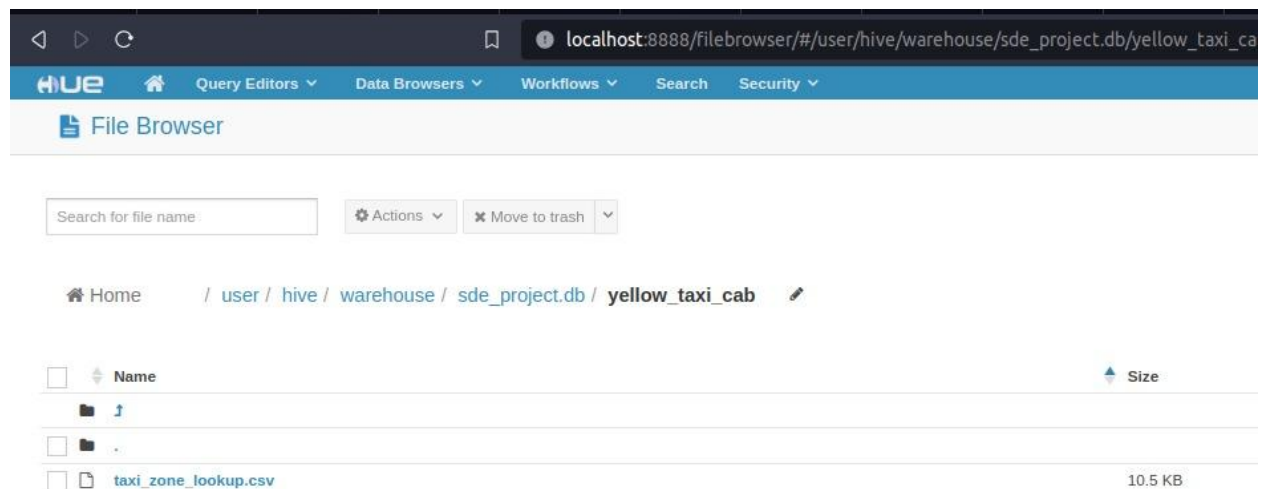
**Table Browser (Bottom Left):** The table browser shows the 'sde\_project' database selected, with a list of tables including 'yellow\_taxi\_data'.

**Query Editor (Bottom Right):** The query editor shows a single query: `1 CREATE TABLE IF NOT EXISTS sde_project.yellow_taxi_data (`  
`2 LocationID INT,`  
`3 Borough STRING,`  
`4 Zone STRING,`  
`5 service_zone STRING`  
`6 )`  
`7 ROW FORMAT DELIMITED`  
`8 FIELDS TERMINATED BY ','`  
`9 STORED AS TEXTFILE`  
`10 TBLPROPERTIES("skip.header.line.count"="1")`

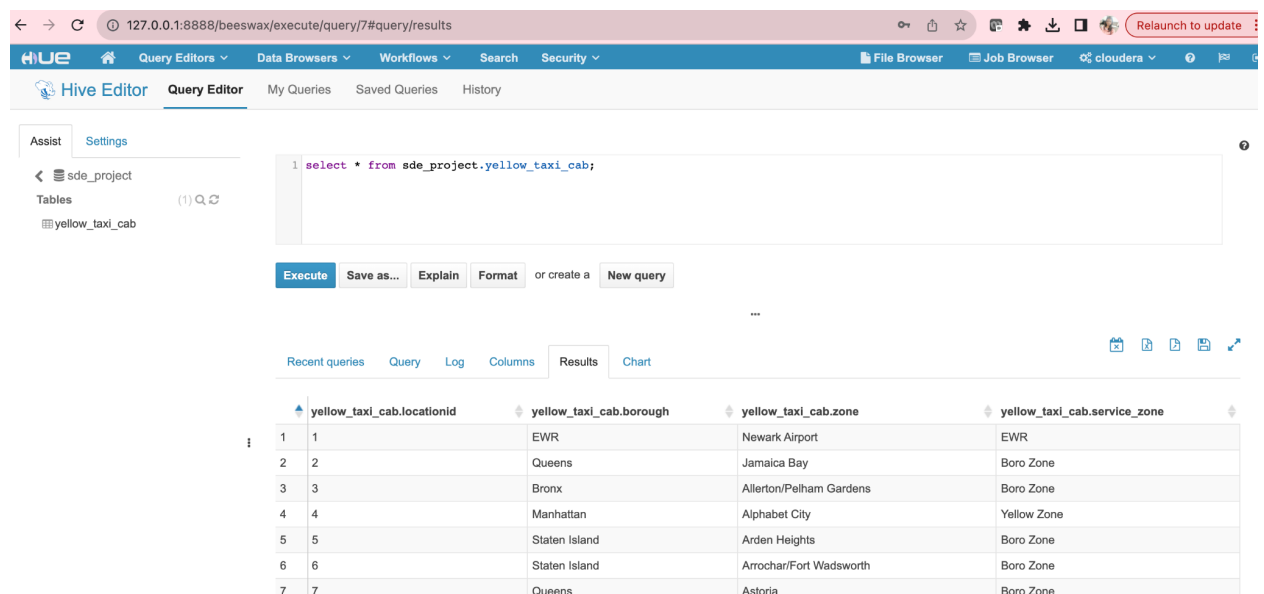
**Query Editor (Bottom):** The query editor shows a single query: `1 LOAD DATA INPATH 'hdfs:///user/cloudera/taxi_zone_lookup.csv' INTO TABLE sde_project.yellow_taxi_cab;`

Why do files move from the Table's loading directory to the Hive Table directory? "- A Hive table can have data loaded from an HDFS (Hadoop Distributed File System) path using the LOAD DATA INPATH command. After running this command, the data files are moved into the Hive table directory in order to guarantee that Hive has management and control over the data inside the Hive table.

Hive integrates with the Hive Metastore to store table metadata, such as location and schema. Hive changes its metastore with the new data location when we load data into a table using LOAD DATA INPATH. Data is arranged by Hive within HDFS according to a particular directory layout. The HDFS directory that each Hive table usually relates to holds the data files that are related to that table.



Now checking if the data is available via performing queries on the table :-





## 1.4 Performing query that requires MapReduce task to be performed :-

```
1 SELECT Borough, COUNT(*) AS row_count FROM sde_project.yellow_taxi_cab GROUP BY Borough;
```

Execute Save as... Explain Format or create a New query

Recent queries Query Log Columns Results Chart

	borough	row_count
1	Bronx	43
2	Brooklyn	61
3	EWR	1
4	Manhattan	69
5	Queens	69
6	Staten Island	20

Behind the curtains tasks:-

Details about the Application Master in the process of executing a Hive query using MapReduce on a Cloudera Distribution for Hadoop (CDH) cluster:

1. **Query Submission:** The HiveServer assembles the Hive SQL query into a number of MapReduce jobs when we submit it.
2. **Job Submission:** The MapReduce jobs are submitted to the YARN ResourceManager.
3. **Application Master Creation:** YARN ResourceManager initiates an Application Master (AM) for every job that is submitted. The Application Master is in charge of negotiating resources with the ResourceManager, organizing the completion of tasks, and keeping track of the status of jobs.

hadoop All Applications Logged in as:

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Reboot Node
1	0	1	0	1	2 GB	8 GB	0 B	1	8	0	1	0	0	0	0

User Metrics for dr.who

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved	VCores Used	VCores Pending	VCores Reserved
0	0	1	0	0	0	0	0 B	0 B	0 B	0	0	0

Show 20 entries

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCoers	Allocated Memory MB	Progress	Tracking
application_1700656538094_0001	cloudera	SELECT Borough, COUNT(*) AS row_co...Borough(Stage-1)	MAPREDUCE	root.cloudera	Wed Nov 22 18:51:24 +0550 2023	N/A	ACCEPTED	UNDEFINED	1	1	2048		UNASSIG

Showing 1 to 1 of 1 entries

The screenshot shows the Hadoop web interface at localhost:8088. The left sidebar contains navigation links for Cluster, About Nodes, Applications, NEW, NEW\_SAVING, SUBMITTED, ACCEPTED, RUNNING, FINISHED, FAILED, KILLED, Scheduler, and Tools. The main content area displays the 'Application Overview' for a running job.

**Application Overview**

User:	cloudera
Name:	SELECT Borough, COUNT(*) AS row_co...Borough(Stage-1)
Application Type:	MAPREDUCE
Application Tags:	
State:	RUNNING
FinalState:	UNDEFINED
Started:	Wed Nov 22 13:21:24 +0000 2023
Elapsed:	50sec
Tracking URL:	ApplicationMaster
Diagnostics:	

**Application Metrics**

Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	110033 MB-seconds, 57 vcore-seconds

**ApplicationMaster**

Attempt Number	Start Time	Node	Logs
1	Wed Nov 22 13:21:24 +0000 2023	quickstart.cloudera:8042	logs

The screenshot shows the Hadoop web interface at localhost:8088, displaying the 'FINISHED Applications' page. A notification at the top states 'Screenshot captured. You can paste the image from the clipboard.' The left sidebar is identical to the previous screenshot. The main content area shows cluster metrics and a table of finished applications.

**Cluster Metrics**

	Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
1	0	0	0	1	0	0 B	8 GB	0 B	0	8	0	1	0	0	0	0

**User Metrics for dr.who**

	Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved	VCores Used	VCores Pending	VCores Reserved
0	0	0	0	1	0	0	0	0 B	0 B	0 B	0	0	0

Show 20 entries

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCoers	Allocated Memory MB	Progress	Tracking UI
application_1700656538094_0001	cloudera	SELECT Borough, COUNT(*) AS row_co...Borough(Stage-1)	MAPREDUCE	root.cloudera	Wed Nov 22 18:51:24 +0550 2023	Wed Nov 22 18:52:20 +0550 2023	FINISHED	SUCCEEDED	N/A	N/A	N/A		History

Showing 1 to 1 of 1 entries

First Previous 1 Next Last

- Application ID Creation:** The submitted query receives an Application ID from YARN that is specific to it and linked to the Application Master.
- Task Assignment and Execution:** Resources (containers) on different cluster nodes are negotiated by the Application Master in conjunction with the ResourceManager. After that, it assigns tasks—including map and reduce tasks—in collaboration with NodeManagers.
- Map Phase:** The map step involves reading data from HDFS and processing it concurrently throughout the cluster. Keys are used by the MapReduce system to sort and shuffle intermediate data.

7. **Reduce Phase:** The Application Master organizes the aggregation of intermediate data using keys during the reduction phase. The findings are written back to HDFS and the final output is generated.
8. **Task and Job Monitoring:** The Application Master keeps an eye on each task's development as well as the overall completion of the assignment. It updates status by communication with the ResourceManager.
9. **Job Completion:** The MapReduce job is concluded when the Application Master notifies the ResourceManager that all tasks have been successfully performed.
10. **History Server:** The history server holds comprehensive historical data, including as logs, counters, and other metadata, regarding finished MapReduce jobs. Performance analysis and troubleshooting can benefit from this information.

## 1.5 Oozie Workflows

Various other processing jobs can be done using the CDH but how do these tasks connect with each other? Answer is the Oozie workflows and coordinators of the CDH. Below is example:-

Task1. Workflow to create HDFS directories and files (action name : fs-12c7) :-

```
<workflow-app name="My Workflow" xmlns="uri:oozie:workflow:0.5">
  <start to="fs-12c7"/>
  <kill name="Kill">
    <message>Action failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
  </kill>
  <action name="fs-12c7">
    <fs>
      <touchz path='${nameNode}/user/cloudera/test' />
    </fs>
    <ok to="End"/>
    <error to="Kill"/>
  </action>
  <end name="End"/>
</workflow-app>
```

cloudera	<b>Name</b>	<b>Value</b>
<b>STATUS</b>	dryrun	False
<b>SUCCEEDED</b>	hue-id-w	3
<b>PROGRESS</b>	jobTracker	localhost:8032
<b>100%</b>	mapreduce.job.user.name	cloudera
<b>ID</b>	nameNode	hdfs://quickstart.cloudera:8020
0000001-231122121616918-oozie-oozi-W	oozie.use.system.libpath	True
<b>VARIABLES</b>	oozie.wf.application.path	hdfs://quickstart.cloudera:8020/user/hue/oozie/workspaces/hue-oozie-1700655912.35
	security_enabled	False
	user.name	cloudera
	<b>Back</b>	

Task2. Workflow to run shell script checking availability of file on hdfs :-

```
<workflow-app name="Test" xmlns="uri:oozie:workflow:0.5">
  <start to="shell-869a"/>
  <kill name="Kill">
    <message>Action failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
  </kill>
  <action name="shell-869a">
    <shell xmlns="uri:oozie:shell-action:0.1">
      <job-tracker>${jobTracker}</job-tracker>
      <name-node>${nameNode}</name-node>
      <exec>/user/cloudera/file_avail_check.sh</exec>
      <capture-output/>
    </shell>
    <ok to="End"/>
    <error to="Kill"/>
  </action>
  <end name="End"/>
</workflow-app>
```

The screenshot shows the Hue Job Browser interface. On the left, a sidebar contains navigation links for JOB ID, TYPE, USER, STATUS, LOGS, MAPS, REDUCES, and DURATION. The main area displays job details for the job ID 'oozie:launcher:T=shell:W=Test:A=shell-b9e4:ID=0000000-231120194824389-oozie-oozi-W'. The job is in a 'SUCCEEDED' state. A table lists various configuration parameters and their values, such as 'dfs.block.access.key.update.interval' set to 600 and 'dfs.blockreport.intervalMsec' set to 21600000.

Name	Value
ID	1700509648185_0001
User	cloudera
Maps	1 of 1
Reduces	0 of 0
Started	11/20/23 13:38:14
Ended	11/20/23 13:38:25
Duration	10747
Status	SUCCEEDED
dfs.block.access.key.update.interval	600
dfs.block.access.token.enable	false
dfs.block.access.token.lifetime	600
dfs.block.scanner.volume.bytes.per.second	1048576
dfs.blockreport.initialDelay	0
dfs.blockreport.intervalMsec	21600000
dfs.blockreport.split.threshold	1000000
dfs.blocksize	134217728
dfs.bytes-per-checksum	512

## Platform 2 : EDA on Local Machine

Now, let's look at one of the processes on the local machine platform and observe the memory occupancies.

We used a dataset called creditcard.csv for EDA. The primary objective of exploratory data analysis, a crucial stage of data analysis, is to enumerate the salient features, patterns, and connections found in a dataset. To comprehend the structure of the data, find any patterns or anomalies, and get insights into it, statistical and visual aids are used. EDA frequently comes first in the data analysis process and serves as a roadmap for later investigations.

EDA is done using two ways - with and without Profiling.

## 2.1 Traditional way using standard libraries, numpy, matplotlib, pandas, seaborn

Without pandas\_profiling method:-

Steps performed:-

### 2.1.1 Data Cleaning:-

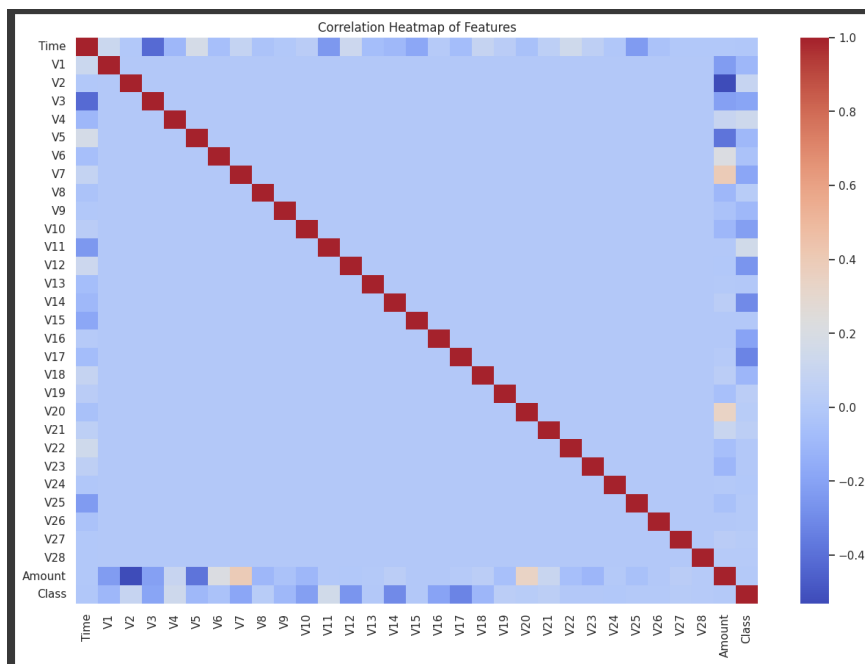
```
2 df.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0

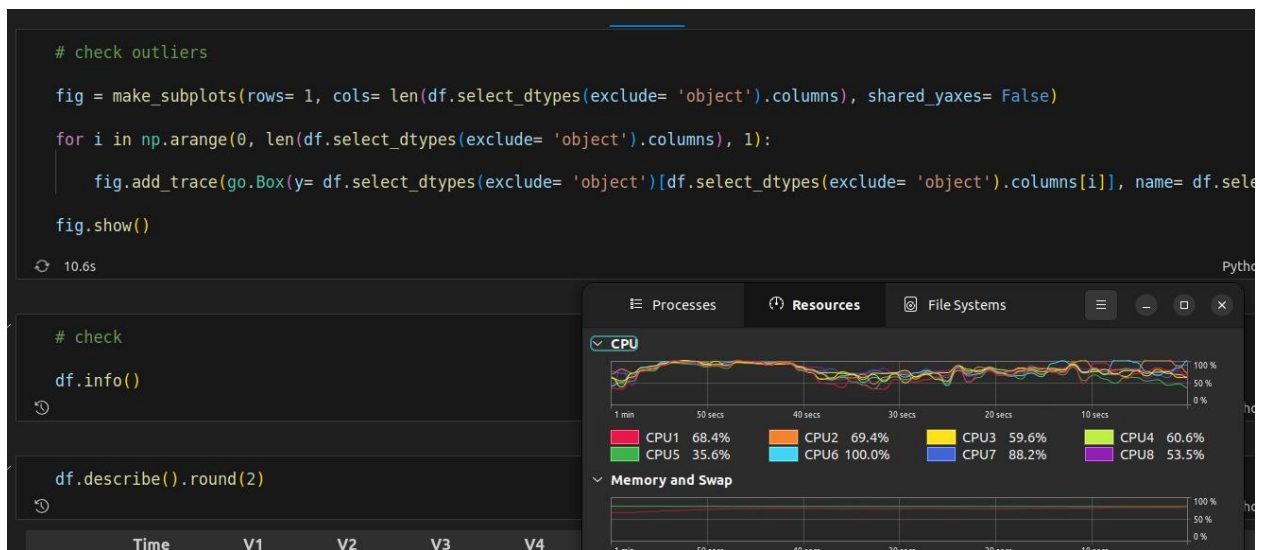
### 2.1.2 Understanding the Dataset:-

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...
count	283726.00	283726.00	283726.00	283726.00	283726.00	283726.00	283726.00	283726.00	283726.00	283726.00	...
mean	94811.08	0.01	-0.00	0.00	-0.00	0.00	-0.00	0.00	-0.00	-0.00	...
std	47481.05	1.95	1.65	1.51	1.41	1.38	1.33	1.23	1.18	1.10	...
min	0.00	-56.41	-72.72	-48.33	-5.68	-113.74	-26.16	-43.56	-73.22	-13.43	...
25%	54204.75	-0.92	-0.60	-0.89	-0.85	-0.69	-0.77	-0.55	-0.21	-0.64	...
50%	84692.50	0.02	0.06	0.18	-0.02	-0.05	-0.28	0.04	0.02	-0.05	...
75%	139298.00	1.32	0.80	1.03	0.74	0.61	0.40	0.57	0.33	0.60	...
max	172792.00	2.45	22.06	9.38	16.88	34.80	73.30	120.59	20.01	15.59	...

### 2.1.3 Correlation Analysis:-



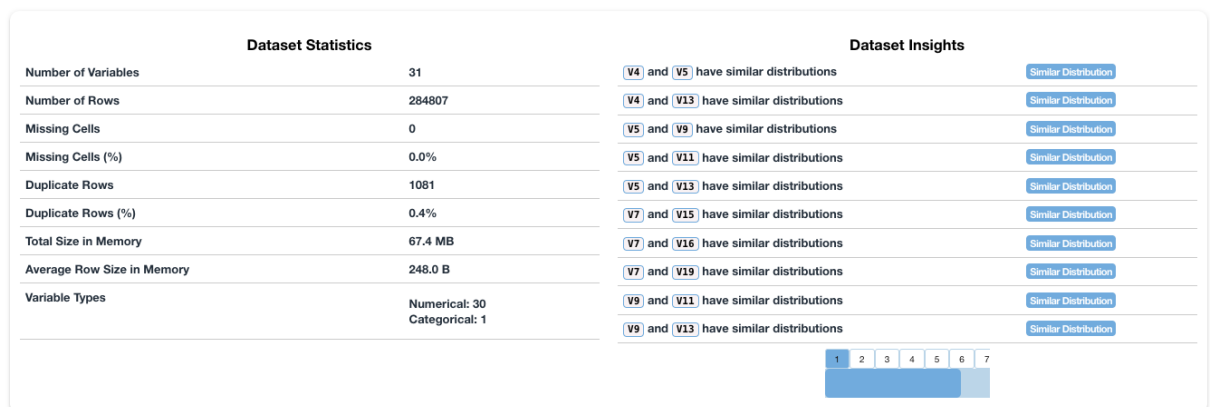
## 2.1.4 CPU Utilization (All CPU cores are used at the same time):-



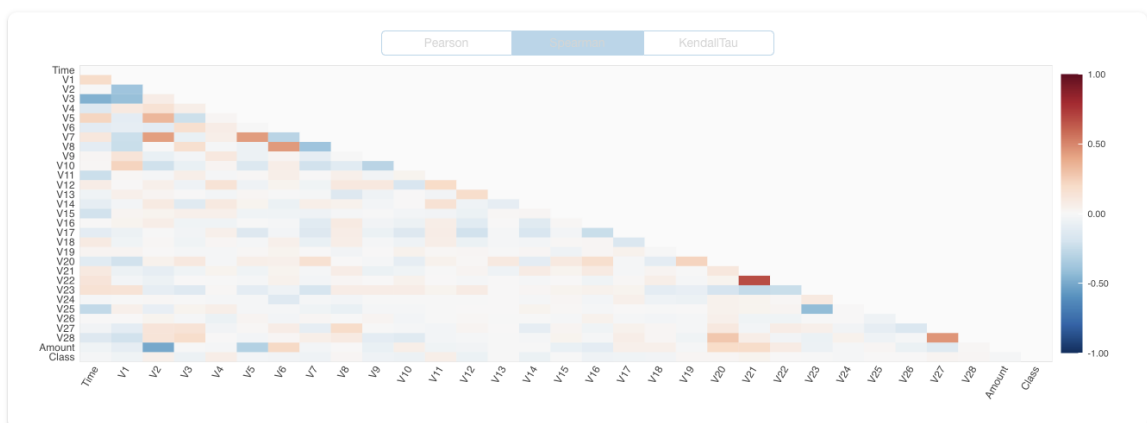
## 2.2 Using EDA library “dataprep” and profiling Method:-

### 2.2.1 Using the create\_report method of dataprep to perform profiling.

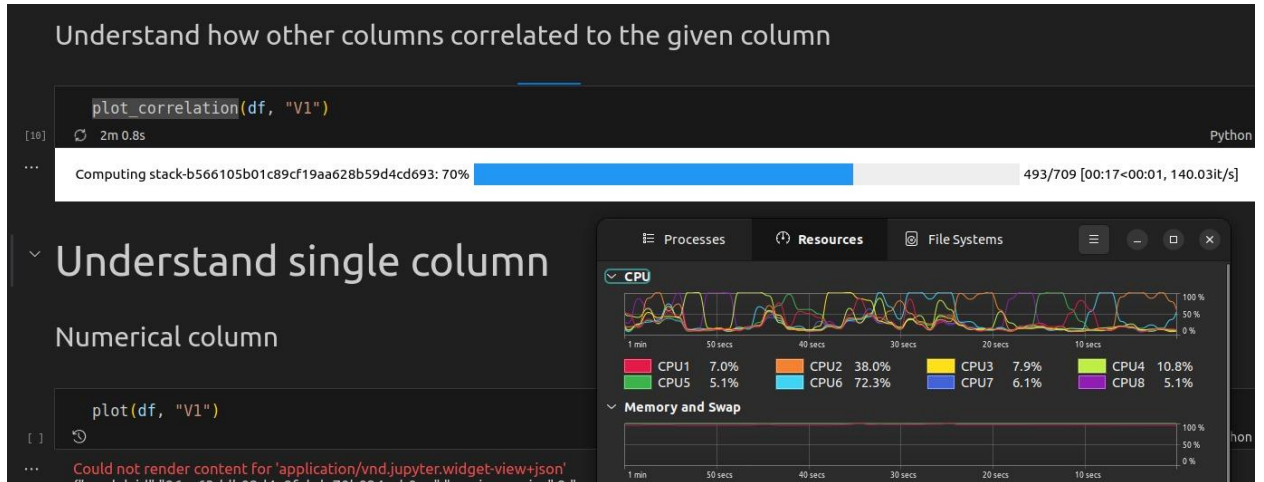
#### Overview



#### Correlations



### 2.2.2 CPU Utilisation after data profiling:-



### 2.3 CPU Utilization is reduced in every single core.

Below are the inferences:-

CPU Core	EDA with dataprep in %	EDA with standard lib in %
Core 1	7	68.4
Core 2	38	69.4
Core 3	7.9	59.6
Core 4	10.8	60.6
Core 5	5.1	35.6
Core 6	72.3	100
Core 7	6.1	88.2
Core 8	5.1	53.5
Average	19.0375	66.9125

## **Conclusion:**

This study set out to conduct a thorough investigation of large data using the local machine Exploratory Data Analysis (EDA) and CDH platform architecture as lenses. The need for strong big data ecosystems is highlighted by the rise in the world's capacity for data storage, as reported by IDC. A practical overview of CDH, local machine EDA was given, emphasizing the subtle differences in the processing methods used by each platform. The flexibility of big data processing was demonstrated by the integration of Hadoop tools, the Hive data warehouse, and Oozie workflows in CDH. Conventional libraries and profiling techniques were used in local machine EDA, showing different CPU utilization. The project offers a comprehensive overview of big data platforms and tools, setting the groundwork for upcoming initiatives in the rapidly changing fields of data engineering and analytics.

## **CODE LINK:-**

[https://github.com/Bhawna-Bhoria/SDE\\_Project](https://github.com/Bhawna-Bhoria/SDE_Project)

## **References:-**

<https://blog.clairvoyantsoft.com/cloduera-quickstart-vm-using-docker-on-mac-2308acd196f2>

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL>

<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9463522&tag=1>

<https://github.com/sfu-db/dataprep>