

Name	Pranay Singhvi
UID No.	202130026

Experiment 7

HONOR PLEDGE	<div>Date : _____</div> <div>I hereby declare that the documentation, code and output attached with this lab experiment has been completed by me in accordance with highest standards of honesty. I confirm that I have not plagiarized or used unauthorized material or given or received illegitimate help for completing this experiment. I will uphold equity and honesty in the evaluation on my work, and if found guilty of plagiarism or dishonesty, will bear the consequences as outlined in the 'integrity' section of the lab rubrics. I am doing so in order to maintain a community built around this code of honour</div> <div>Pranay Pranay Singhvi</div>
PROBLEM STATEMENT	<p>Title: Big data analysis using Hive/ Use of Graph database</p> <ol style="list-style-type: none">1. Import the batch-specific data and store it as a Hive Table2. Perform a Hive query on your uploaded dataset3. Pull data into a spark dataframe and repeat the query using a spark dataframe4. Link neo4j to your underlying datastore and run a graph query
THEORY	<div><p>1. Hadoop Overview:</p><p>Hadoop is an open-source framework designed for distributed storage and processing of large data sets across clusters of commodity hardware. It provides a distributed file system (HDFS) for storage and a framework (MapReduce) for processing and analyzing large datasets.</p><p>Installation:</p><p>Guide Followed: Hadoop-on-Ubuntu</p><ol style="list-style-type: none">1. Create New User for Hadoop: <code>sudo adduser hdoop & su - hdoop</code>2. Set Up SSH Keys: <code>ssh-keygen -t rsa -P "" -f ~/.ssh/id_rsa</code>3. Download the latest stable release of Hadoop from the Apache Hadoop Website.4. Configure Hadoop Environment: Set JAVA_HOME in <code>/usr/local/hadoop/etc/hadoop/hadoop-env.sh</code>5. Update Hadoop's XML configuration files with default config:6. Initialize the Hadoop filesystem namespace: <code>/usr/local/hadoop/bin/hdfs namenode -format</code></div> <div><pre>hdoop@DESKTOP-5UL8JKF:~\$ start-dfs.sh Starting namenodes on [localhost] Starting datanodes Starting secondary namenodes [DESKTOP-5UL8JKF] hdoop@DESKTOP-5UL8JKF:~\$ start-yarn.sh Starting resourcemanager Starting nodemanagers hdoop@DESKTOP-5UL8JKF:~\$</pre></div> <div>7. Start Hadoop, Launch HDFS and YARN services: start-all.sh, verify using: jps</div>


```
hadoop@DESKTOP-5UL8JKF:~$ jps
64678 Jps
935 NameNode
1720 NodeManager
1242 SecondaryNameNode
1051 DataNode
1598 ResourceManager
hadoop@DESKTOP-5UL8JKF:~$
```

8. Check localhost:9870 for Hadoop UI and localhost:8088 for YARN UI:

←→↻localhost:9870/dfshealth.html#tab-overview

HadoopOverviewDatanodesDatanode Volume FailuresSnapshotStartup ProgressUtilities

Overview 'localhost:9000' (✓active)

Started:	Mon Mar 25 08:51:50 +0530 2024
Version:	3.3.6, r1be78238728da9266a4f88195058f08fd012bf9c
Compiled:	Sun Jun 18 13:52:00 +0530 2023 by ubuntu from (HEAD detached at release-3.3.6-RC1)
Cluster ID:	CID-a02619bd-fcb8-4ea9-8b4d-dca1596e1146
Block Pool ID:	BP-330841302-127.0.1.1-1711259157079

Summary

Security is off.


Safemode is off.

14 files and directories, 4 blocks (4 replicated blocks, 0 erasure coded block groups) = 18 total filesystem object(s).

Heap Memory used 126.01 MB of 274 MB Heap Memory. Max Heap Memory is 846.5 MB.

Non Heap Memory used 49.7 MB of 51.36 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

←→↻localhost:8088/cluster



All Applications

Cluster

AboutNodesNode LabelsApplicationsNEWNEW SAVING SUBMITTEDACCEPTEDRUNNINGFINISHEDFAILEDKILLED

Scheduler

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources
0	0	0	0	0	<memory:0 B, vCores:0>

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes
0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:1>

Show 20 entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers
No data available in table												

Showing 0 to 0 of 0 entries

2. Hive

Overview:

Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis. It provides a SQL-like interface (HiveQL) to query and analyze data stored in Hadoop's distributed file system (HDFS).

Installation:

Guide Followed: [Hive-on-Ubuntu](#)

1. Download the latest stable release of Hive from the [Apache Hive website](#).
2. Extract the downloaded archive to a directory on your WSL Ubuntu system.
3. Set up environment variables in your '.bashrc' file to specify the Hive home directory and add Hive's bin directory to your PATH.
4. Configure Hive's XML files ('hive-site.xml', 'hive-default.xml', etc.) according to your Hadoop and metastore setup.
5. Create Hive Directories in HDFS, Create two separate directories to store data in the HDFS layer: tmp & warehouse
6. Initiate Derby Database: \$HIVE_HOME/bin/schematool -dbType derby -initSchema
7. Start Hive services and the metastore using the provided scripts ('hive --service metastore', etc.).

```
hadoop@DESKTOP-5UL8JKF:~/apache-hive-3.1.2-bin/bin$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoop/apache-hive-3.1.2-bin/lib/log4j-slf4j-impl-2.10.0.jar:/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-reload4j-1.7.2.jar:/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = bc63537d-dd02-4292-806a-2bafc4bcee4f

Logging initialized using configuration in jar:file:/home/hadoop/apache-hive-3.1.2-bin/lib/hive-r!hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using the execution engine (i.e. spark, tez) or using Hive 1.X releases.
Hive Session ID = b7729195-232a-434e-ae7e-293192d9374d
hive> show tables;
OK
enron
temp_employee
Time taken: 1.668 seconds, Fetched: 2 row(s)
hive>
```

```
hadoop@DESKTOP-5UL8JKF:~$ cd $HIVE_HOME/bin
hadoop@DESKTOP-5UL8JKF:~/apache-hive-3.1.2-bin/bin$ hive --service metastore
2024-03-25 08:57:18: Starting Hive Metastore Server
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoop/apache-hive-3.1.2-bin/lib/log4j-slf4j-impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
```

3. Spark

Overview:

Apache Spark is an open-source, distributed computing system that provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. It supports various programming languages such as Scala, Java, Python, and R.

Installation:

pip install spark (in python notebook)

← → ↻

localhost:4040/jobs/

☆ 📁 🌐 📄 📄 📄 📄 📄 📄 📄

spark3.5.1

Jobs

Stages

Storage

Environment

Executors

SQL / DataFrame

CSV to Hive appl

Spark Jobs (?)

User: shabbar

Total Uptime: 44 s

Scheduling Mode: FIFO

Completed Jobs: 3

▶ Event Timeline

▼ Completed Jobs (3)

Page: 1

1 Pages. Jump to 1. Show 100 items in a pa

Job Id ▼	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
2	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2024/03/25 09:14:03	0.5 s	1/1	1/1
1	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2024/03/25 09:13:58	4 s	1/1	4/4
0	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2024/03/25 09:13:54	2 s	1/1	1/1

Page: 1

1 Pages. Jump to 1. Show 100 items in a pa

← → ↻

localhost:4040/SQL/execution/?id=1

📄 📄 📄 📄 📄 📄 📄 📄 📄 📄

spark3.5.1

Jobs

Stages

Storage

Environment

Executors

SQL / DataFrame

Details for Query 1

Submitted Time: 2024/03/25 09:14:03

Duration: 1 s

Succeeded Jobs: 2

☐ Show the Stage ID and Task ID that corresponds to the max metric

Scan csv

number of output rows: 21

number of files read: 1

metadata time: 0 ms

size of files read: 106.2 MiB

↓

WholeStageCodegen (1)

duration: 0 ms

Project

↓

CollectLimit

▶ Details

4. Neo4j

Overview:

Neo4j is a graph database management system that provides an efficient and expressive way to store, manage, and query highly connected data. It is optimized for handling graph data and supports powerful querying capabilities using the Cypher query language.

Installation:
Guide Followed: [Neo4j-on-Ubuntu](#)

1. Download the latest stable release of Neo4j from the [Neo4j Website](#)
2. Follow the installation instructions provided for Linux distributions.
3. After installation, start the Neo4j service using the provided startup scripts: `sudo systemctl enable/start/edit/status/stop neo4j`.
4. Configure username and password through cypher-shell: `:server change-password`
5. Access the Neo4j browser interface using a web browser and connect to the Neo4j database.

```
hdoop@DESKTOP-5UL8JKF:~$ sudo systemctl restart neo4j
hdoop@DESKTOP-5UL8JKF:~$ cypher-shell -a 'neo4j://127.0.0.1:7687'
username: neo4j
password:
Password change required
new password:
confirm password:
A password must be at least 8 characters.
hdoop@DESKTOP-5UL8JKF:~$ cypher-shell -a 'neo4j://127.0.0.1:7687'
username: neo4j
password:
Password change required
new password:
confirm password:
Connected to Neo4j using Bolt protocol version 5.4 at neo4j://127.0.0.1:7687
Type :help for a list of available commands or :exit to exit the shell.
Note that Cypher queries must end with a semicolon.
neo4j@neo4j>
```

```
hdoop@DESKTOP-5UL8JKF:~$ cypher-shell -a 'neo4j://127.0.0.1:7687'
username: neo4j
password:
Connected to Neo4j using Bolt protocol version 5.4 at neo4j://127.0.0.1:7687 as user neo4j.
Type :help for a list of available commands or :exit to exit the shell.
Note that Cypher queries must end with a semicolon.
neo4j@neo4j> CREATE (:Shark {name: 'Great White'});
0 rows
ready to start consuming query after 542 ms, results consumed after another 0 ms
Added 1 nodes, Set 1 properties, Added 1 labels
neo4j@neo4j> CREATE
    (:Shark {name: 'Hammerhead'})-[:FRIEND]→
    (:Shark {name: 'Sammy'})-[:FRIEND]→
    (:Shark {name: 'Megalodon'});
0 rows
ready to start consuming query after 250 ms, results consumed after another 0 ms
Added 3 nodes, Created 2 relationships, Set 3 properties, Added 3 labels
neo4j@neo4j> MATCH (a:Shark),(b:Shark)
    WHERE a.name = 'Sammy' AND b.name = 'Megalodon'
    CREATE (a)-[r:ORDER { name: 'Lamniformes' } ]→(b)
    RETURN type(r), r.name;
+-----+
| type(r) | r.name |
+-----+
| "ORDER" | "Lamniformes" |
+-----+
```

1. Importing Libraries & Dataset

```
In [ ]: import re
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from neo4j import GraphDatabase
from pyspark.sql import SparkSession
# read csv file
df = pd.read_csv('../Datasets/enron.csv', index_col=0, low_memory=False)
df.head()
```

```
Out [ ]:
```

	Message-ID	Date	From	To	Subject	X-From	X-To	X-cc	X-bcc	X-F
0	<18782981.1075855378110.JavaMail.evans@thyme>	2001-05-14 23:39:00	frozenset({'phillip.allen@enron.com'})	frozenset({'tim.belden@enron.com'})	NaN	Phillip K Allen	Tim Belden <Tim Belden/Enron@EnronXGate>	NaN	NaN	\\Phillip_Allen_Jan2002_1\\Phillip K.
1	<15464986.1075855378456.JavaMail.evans@thyme>	2001-05-04 20:51:00	frozenset({'phillip.allen@enron.com'})	frozenset({'john.lavorato@enron.com'})	Re:	Phillip K Allen	John J Lavorato <John J Lavorato/ENRON@enronXg...	NaN	NaN	\\Phillip_Allen_Jan2002_1\\Phillip K.
2	<24216240.1075855687451.JavaMail.evans@thyme>	2000-10-18 10:00:00	frozenset({'phillip.allen@enron.com'})	frozenset({'leah.arsdall@enron.com'})	Re: test	Phillip K Allen	Leah Van Arsdall	NaN	NaN	\\Phillip_Allen_Dec2000\\Folders\\sen
3	<13505866.1075863688222.JavaMail.evans@thyme>	2000-10-23 13:13:00	frozenset({'phillip.allen@enron.com'})	frozenset({'randall.gay@enron.com'})	NaN	Phillip K Allen	Randall L Gay	NaN	NaN	\\Phillip_Allen_Dec2000\\Folders\\sen
4	<30922949.1075863688243.JavaMail.evans@thyme>	2000-08-31 12:07:00	frozenset({'phillip.allen@enron.com'})	frozenset({'greg.piper@enron.com'})	Re: Hello	Phillip K Allen	Greg Piper	NaN	NaN	\\Phillip_Allen_Dec2000\\Folders\\sen

5 rows x 51 columns

2. Preprocessing Data

```
In [ ]: def sanitize_column_name(name):
        return re.sub(r'^A-Za-z0-9_', '_', name)

new_columns = [sanitize_column_name(col) for col in df.columns]
df.columns = new_columns
df.to_csv("../Datasets/cleaned_enron.csv", index=False)

In [ ]: df_new = pd.read_csv("../Datasets/cleaned_enron.csv", index_col=0, low_memory=False)
df_new.shape[0]
df_subset = df.iloc[:50000]
df_subset.to_csv("../Datasets/enron_chunk.csv", index=False)

In [ ]: df_subset = pd.read_csv("../Datasets/enron_chunk.csv", index_col=0, low_memory=False)
print(df_subset.shape[0])
df_subset.head()
```

50000

Out[]:

Message_ID	Date	From	To	Subject	X_From	X_To	X_cc	X_bcc	
<18782981.1075855378110.JavaMail.evans@thyme>	2001-05-14 23:39:00	frozenset({'phillip.allen@enron.com'})	frozenset({'tim.belden@enron.com'})	NaN	Phillip K Allen	Tim Belden <Tim Belden/Enron@EnronXGate>	NaN	NaN	\Phillip_Allen_Jan2001
<15464986.1075855378456.JavaMail.evans@thyme>	2001-05-04 20:51:00	frozenset({'phillip.allen@enron.com'})	frozenset({'john.lavorato@enron.com'})	Re:	Phillip K Allen	John J Lavorato <John J Lavorato/ENRON@enronXgate>	NaN	NaN	\Phillip_Allen_Jan2001
<24216240.1075855687451.JavaMail.evans@thyme>	2000-10-18 10:00:00	frozenset({'phillip.allen@enron.com'})	frozenset({'leah.arsdall@enron.com'})	Re: test	Phillip K Allen	Leah Van Arsdall	NaN	NaN	\Phillip_Allen_Dec2000
<13505866.1075863688222.JavaMail.evans@thyme>	2000-10-23 13:13:00	frozenset({'phillip.allen@enron.com'})	frozenset({'randall.gay@enron.com'})	NaN	Phillip K Allen	Randall L Gay	NaN	NaN	\Phillip_Allen_Dec2000
<30922949.1075863688243.JavaMail.evans@thyme>	2000-08-31 12:07:00	frozenset({'phillip.allen@enron.com'})	frozenset({'greg.piper@enron.com'})	Re: Hello	Phillip K Allen	Greg Piper	NaN	NaN	\Phillip_Allen_Dec2000

5 rows x 50 columns

3. Loading Dataset to a Spark Dataframe

```
In [ ]: # Create SparkSession
spark = SparkSession.builder \
    .appName("CSV to Hive") \
    .config("hive.metastore.uris", "thrift://localhost:9083") \
    .enableHiveSupport() \
    .getOrCreate()

# Read CSV file into DataFrame
csv_file_path = "./eron2.csv"
df1 = spark.read.csv(csv_file_path, header=True, inferSchema=True)
# Define a function to sanitize column names
# def sanitize_column_name(name):
#     return re.sub(r'[^A-Za-z0-9_]', '_', name)

# new_columns = [sanitize_column_name(col) for col in df1.columns]
# df1 = df1.toDF(*new_columns)
df1.show()
```

your 131072x1 screen size is bogus. expect trouble
Warning: Ignoring non-Spark config property: hive.metastore.uris
24/03/24 11:55:07 WARN Utils: Your hostname, DESKTOP-5UL8JKF resolves to a loopback address: 127.0.1.1; using 172.27.122.177 instead (on interface eth0)
24/03/24 11:55:07 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/03/24 11:55:10 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
24/03/24 11:55:30 WARN SparkStringUtils: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.

only showing top 20 rows

4. Using Spark to load Data into Hive Table

```
In [ ]: # Extract schema from DataFrame
schema = df1.schema
print(schema)

StructType([StructField('Message_ID', StringType(), True), StructField('Date', StringType(), True), StructField('From', StringType(), True), StructField('To', StringType(), True), StructField('Subject', StringType(), True), StructField('X_From', StringType(), True), StructField('X_To', StringType(), True), StructField('X_cc', StringType(), True), StructField('X_bcc', StringType(), True), StructField('X_Folder', StringType(), True), StructField('X_Origin', StringType(), True), StructField('X_FileName', StringType(), True), StructField('content', StringType(), True), StructField('user', StringType(), True), StructField('Cat_1_level_1', StringType(), True), StructField('Cat_1_level_2', StringType(), True), StructField('Cat_1_weight', StringType(), True), StructField('Cat_2_level_1', StringType(), True), StructField('Cat_2_level_2', StringType(), True), StructField('Cat_2_weight', StringType(), True), StructField('Cat_3_level_1', StringType(), True), StructField('Cat_3_level_2', StringType(), True), StructField('Cat_3_weight', StringType(), True), StructField('Cat_4_level_1', StringType(), True), StructField('Cat_4_level_2', StringType(), True), StructField('Cat_4_weight', StringType(), True), StructField('Cat_5_level_1', StringType(), True), StructField('Cat_5_level_2', StringType(), True), StructField('Cat_5_weight', StringType(), True), StructField('Cat_6_level_1', StringType(), True), StructField('Cat_6_level_2', StringType(), True), StructField('Cat_6_weight', StringType(), True), StructField('Cat_7_level_1', StringType(), True), StructField('Cat_7_level_2', StringType(), True), StructField('Cat_7_weight', StringType(), True), StructField('Cat_8_level_1', StringType(), True), StructField('Cat_8_level_2', StringType(), True), StructField('Cat_8_weight', StringType(), True), StructField('Cat_9_level_1', StringType(), True), StructField('Cat_9_level_2', StringType(), True), StructField('Cat_9_weight', StringType(), True), StructField('Cat_10_level_1', StringType(), True), StructField('Cat_10_level_2', StringType(), True), StructField('Cat_10_weight', StringType(), True), StructField('Cat_11_level_1', StringType(), True), StructField('Cat_11_level_2', StringType(), True), StructField('Cat_11_weight', StringType(), True), StructField('Cat_12_level_1', StringType(), True), StructField('Cat_12_level_2', StringType(), True), StructField('Cat_12_weight', StringType(), True), StructField('labeled', StringType(), True)])

In [ ]: # Create Hive table with extracted schema
table_name = "enron"
schema_ddl = ','.join([f'{field.name} {field.dataType.simpleString()}' for field in schema])
create_table_query = f"CREATE TABLE IF NOT EXISTS {table_name} ({schema_ddl}) STORED AS PARQUET"
spark.sql(create_table_query)

24/03/24 11:46:32 WARN SessionState: METASTORE_FILTER_HOOK will be ignored, since hive.security.authorization.manager is set to instance of HiveAuthorizerFactory.

Out [ ]: DataFrame[]

In [ ]: # Load data into Hive table
df1.write.mode("overwrite").saveAsTable(table_name)

In [ ]: spark.stop()
```


5. Performing Hive Queries

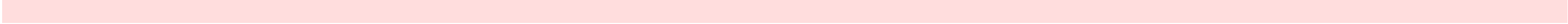
```
In [ ]: # Create SparkSession
spark = SparkSession.builder \
    .appName("Load Hive Table") \
    .config("hive.metastore.uris", "thrift://localhost:9083") \
    .enableHiveSupport() \
    .getOrCreate()

# Load Hive table into DataFrame
df = spark.sql("SELECT * FROM enron")

# Show DataFrame
df.show()
```

24/03/24 11:55:41 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.
[Stage 3:> (0 + 1) / 1]

NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
<15691897.1075854... 2000-05-08 11:46:00 frozenset({'eric... frozenset({'earl... Daily Throughput ...						Eric Bass	Earl Tisdale	NULL	NUL
L \Eric_Bass_Dec200...	Bass-E	ebass.nsf Earl, James McKay...	bass-e	NULL	NULL	NULL	NULL	NULL	N
ULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	
NULL	NULL	NULL	NULL	NULL	NULL	False			
+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
-+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
-+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
-+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
-+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
only showing top 20 rows									



6. Linking Neo4j to the Underlying Datastore

```
In [ ]: # connect to neo4j
uri = "bolt://localhost:7687"
neo4j_user = "neo4j"
neo4j_password = "password"
driver = GraphDatabase.driver(uri, auth=("neo4j", "password"))

# write data to neo4j
df.write \
    .format("org.neo4j.spark.DataSource") \
    .option("url", uri) \
    .option("authentication.basic.username", neo4j_user) \
    .option("authentication.basic.password", neo4j_password) \
    .option("labels", ":Enron") \
    .option("node.keys", "id") \
    .mode("Overwrite") \
    .save()
```

CONCLUSION

This experiment provided us with hands-on experience in leveraging Hive for data storage and querying within a distributed environment. Additionally, we gained proficiency in utilizing Spark to ingest data into Hive tables and execute queries. Moreover, we acquired skills in employing Neo4j for conducting graph-based queries on our dataset. Through this experiment, we gained valuable insights into the capabilities of big data analysis and its potential for extracting meaningful insights from vast datasets.