PERIYAR UNIVERSITY

(NAAC 'A++' Grade with CGPA 3.61 (Cycle - 3)

State University - NIRF Rank 56 - State Public University Rank 25)

SALEM - 636 011

CENTRE FOR DISTANCE AND ONLINE EDUCATION (CDOE)

MASTER OF COMPUTER APPLICATIONS

SEMESTER - II

CORE COURSE: BIG DATA ANALYTICS LAB

(Candidates admitted from 2024 onwards)

PERIYAR UNIVERSITY

CENTRE FOR DISTANCE AND ONLINE EDUCATION (CDOE)

M. C. A. 2024 admission onwards

Core Lab : BIG DATA ANALYTICS LAB

Prepared by:

Centre for Distance and Online Education (CDOE)

Periyar University, Salem - 636011

Credits: 2

Big Data Analytics Lab

Course Objectives

● To teach the fundamental techniques for handling the big data tools.

● To familiarize the tools required to manage big data.

● To analyse big data using Hadoop, MapReduce, Hive, and Pig

● To teach the fundamental principles in achieving big data

analytics with scalability and streaming capability

● To enable students to have skills that will help them to solve complex.

List of Experiments

1. Implement File System Shell Commands for HDFS in

Hadoop Environment

2. Write a Mapreduce program using single reduce function for

finding Maximum and Minimum Number

3. Write a Mapreduce program using multiple reduce function for

Word Count in an given Text document

4. Implement the following using Pig Latin Input and Output

Operations Relational Operations

5. Implement the following using Pig Latin User Defined

Functions Advanced Relational Operations

6. Write a Word Count program using Pig Latin Script

7. Write a program to find a maximum temperature using Pig Latin Script

8. Implement the following using Hive commands Handling the

Database Creating and Manipulating table

9. Implement Simple Queries for database using MongoDB

10. Implement Simple Queries for collections using MongoDB

Course Outcomes

On the successful completion of the course, students will be able to

CO1:	Understand conceptually how Big Data is stored and implement it using different tools	K1- K6
CO2:	Comprehend and implement programs for data storage in HDFS and table manipulation using Big Data tools in Hadoop environment	
CO3:	Critically analyse and examine existing Big Data datasets and implementations the solutions for it using MongoDB	
CO4:	Understand and examine existing Big Data datasets and implementations the solutions using HIVE database.	
CO5:	Comprehend and review existing datasets and implementations the solutions to handle it using PIG	

K1- Remember, K2- Understand, K3- Apply , K4- Analyse, K5- Evaluate, K6- Create

Mapping with Programme Outcomes

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10
CO1	H	H	M	H	M	H	M	H	H	-
CO2	H	M	H	H	H	M	M	L	L	M
CO3	H	H	H	H	H	H	H	M	M	H
CO4	H	M	H	H	H	M	H	M	M	H
CO5	H	H	H	H	H	H	H	H	H	H

H- High; M-Medium; L-Low

Contents

Lab 1: Implement File System Commands for HDFS in Hadoop Environment

Objective: Learn how to work with the Hadoop Distributed File System (HDFS) using command-line tools.

Prerequisites: Hadoop cluster setup and running.

Task 1: List Files in HDFS

In this task, you will use HDFS commands to list files in an HDFS

directory. # List files in an HDFS directory

hdfs dfs -ls /user/your_username/your_hdfs_directory

Replace /user/your_username/your_hdfs_directory with the actual HDFS directory you want to list.

Task 2: Upload a File to HDFS

In this task, you will use HDFS commands to upload a local file to

HDFS. # Upload a local file to HDFS

hdfs dfs -copyFromLocal /path/to/local/file.txt /user/your_username/your_hdfs_directory/

Replace /path/to/local/file.txt with the path to your local file and /user/your_username/your_hdfs_directory/ with the target HDFS destination.

Task 3: Download a File from HDFS

In this task, you will use HDFS commands to download a file from HDFS to your local file system.

 # Download a file from HDFS to the local file system

hdfs dfs -copyToLocal /user/your_username/your_hdfs_directory/file.txt /path/to/local/downloaded_file.txt

Adjust the HDFS file path and local destination path as needed.

Task 4: Delete a File in HDFS

In this task, you will use HDFS commands to delete a file in HDFS.

 # Delete a file in HDFS

hdfs dfs -rm /user/your_username/your_hdfs_directory/file.txt

Replace /user/your_username/your_hdfs_directory/file.txt with the path to the file you want to delete.

Similarly try the followings commands in the Hadoop environment to get familiarize it

Command	Description	Example
start-all.sh	Start the entire Hadoop cluster, including HDFS and YARN.	start-all.sh
stop-all.sh	Stop the entire Hadoop cluster.	stop-all.sh
start-dfs.sh	Start the Hadoop Distributed File System (HDFS) service.	start-dfs.sh
stop-dfs.sh	Stop the HDFS service.	stop-dfs.sh
start-yarn.sh	Start the Yet Another Resource Negotiator (YARN) service.	start-yarn.sh
stop-yarn.sh	Stop the YARN service.	stop-yarn.sh
hdfs dfs -ls /path/to/directory	List files and directories in HDFS.	hdfs dfs -ls /user/hadoop/input
hdfs dfs -mkdir /path/to/new_directory	Create a new directory in HDFS.	hdfs dfs -mkdir /user/hadoop/output
hdfs dfs -copyFromLocal /local/file /hdfs/path	Upload a local file to HDFS.	hdfs dfs -copyFromLocal /local/myfile.txt /user/hadoop/input

hdfs dfs -copyToLocal /hdfs/path /local/destination	Download a file from HDFS.	hdfs dfs -copyToLocal /user/hadoop/output/part-00000 /local/output.txt
hdfs dfs -rm /path/to/file	Delete a file in HDFS.	hdfs dfs -rm /user/hadoop/output/part-00000
yarn logs -applicationId <application_id>	View application logs.	yarn logs -applicationId application_1234567 89_0001
yarn application -list	List running YARN applications.	yarn application -list
hadoop jar <jar-file> <main-class> <input-path> <output-path>	Submit a MapReduce job.	hadoop jar myjob.jar com.example.MyJob /user/hadoop/input /user/hadoop/output
hdfs dfsadmin -report	Check the health and status of the Hadoop cluster.	hdfs dfsadmin -report
hdfs dfs -setrep -w <replication_factor> /path/to/file	Set file replication factor.	hdfs dfs -setrep -w 3 /user/hadoop/input/myfile.txt
hdfs dfs -du -s -h /path/to/directory	Check HDFS disk usage.	hdfs dfs -du -s -h /user/hadoop/output

hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-<version>.jar -mapper <mapper_script> -reducer <reducer_script> -input <input_path> -output <output_path>	Run a MapReduce Streaming job.	hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.1.jar -mapper mapper.py -reducer reducer.py -input /user/hadoop/input -output /user/hadoop/output
hdfs fsck /path/to/file - files -blocks -locations	Get file block locations in HDFS.	hdfs fsck /user/hadoop/input/myfile.txt -files -blocks -locations
Open a web browser and navigate to http://resource_manager_host:8088	Check the Resource Manager Web UI.	Open a web browser and navigate to http://localhost:8088

Hadoop Command Execution and Management in a Big Data Environment (namely rr) : A Quick Reference Guide

Command	Description	Output/Notes
start-dfs.sh	Start HDFS services (NameNode, DataNode, SecondaryNameNode)	Starting namenodes on [localhost] Starting datanodes Starting secondary namenodes [Ubuntu] WARN util.NativeCodeLoader: Unable to load native-rr library...

Command	Description	Output/Notes
start-yarn.sh	Start YARN services (ResourceManager, NodeManager)	Starting resourcemanager... Starting nodemanagers...
jps	List all Java processes related to Hadoop	8979 ResourceManager 9110 NodeManager 12888 DataNode 13098 SecondaryNameNode 13535 Jps 12751 NameNode
hdfs dfs -ls /	List files in the root directory	Found 1 items drwx---------- rr supergroup 0 2023-03-04 22:17 /user
hdfs dfs - mkdir /Sojana	Create a directory named /Sojana	(No output if successful)
hdfs dfs -ls /	List files again to verify directory creation	Found 2 items drwxr-xr-x - rr supergroup 0 2023-03-04 22:33 /Sojana drwx---------- rr supergroup 0 2023-03-04 22:17 /user
hdfs dfs - touchz /Sojana/stud.txt	Create an empty file named stud.txt in /Sojana	(No output if successful)
hdfs dfs -ls /Sojana/	List files in /Sojana	Found 1 items -rw-r--r-- 1 rr supergroup 0 2023-03-04 22:34 /Sojana/stud.txt
hdfs dfs - put DATA.txt /Sojana/	Upload a local file DATA.txt to /Sojana	(No output if successful)

hdfs dfs -ls /Sojana/	List files in /Sojana again to verify upload	Found 2 items -rw-r--r-- 1 rr supergroup 7 2023-03-04 22:36 /Sojana/DATA.txt -rw-r--r-- 1 rr supergroup 0 2023-03-04 22:34 /Sojana/stud.txt
hdfs dfs - cat /Sojana/DATA .txt	Display contents of DATA.txt	Hellow
hdfs dfs -du /Sojana/DATA .txt	Display disk usage of DATA.txt	7 7 /Sojana/DATA.txt
hdfs dfs - setrep -R 4 /Sojana	Set replication factor of all files in /Sojana to 4	Replication 4 set: /Sojana/DATA.txt Replication 4 set: /Sojana/stud.txt

Command	Description	Output/Notes
echo $?	Check return code of the last command	0
hdfs dfs - count -q /Sojana/DATA .txt	Display count and quota information of DATA.txt	none inf none inf 0 1 7 /Sojana/DATA.txt
hdfs dfs - mkdir /temp	Create a directory named /temp	(No output if successful)
hdfs dfs -cp /Sojana /temp	Copy /Sojana directory to /temp	(No output if successful)
hdfs dfs -ls /temp	List files in /temp to verify copy	Found 1 items drwxr-xr-x - rr supergroup 0 2023-03-04 22:49 /temp/Sojana
hdfs dfs -mv /Sojana/DATA .txt /temp/DATA.t xt	Move DATA.txt from /Sojana to /temp	(No output if successful)
hdfs dfs -ls /temp/	List files in /temp again to verify move	Found 2 items -rw-r--r-- 4 rr supergroup 7 2023-03-04 22:36 /temp/DATA.txt drwxr-xr-x - rr supergroup 0 2023-03-04 22:49 /temp/Sojana
hdfs dfs - rmr /temp	Remove /temp directory recursively	rmr: DEPRECATED: Please use '- rm -r' instead. INFO fs.TrashPolicyDefault: Moved: 'hdfs://localhost:54310/temp' to trash at: hdfs://localhost:54310/user/rr/ .Trash/Current/temp

Command	Description	Output/Notes
hdfs dfs -ls /	List files in the root directory to verify the state after deletion	Found 2 items drwxr-xr-x - rr supergroup 0 2023-03-04 22:53 /Sojana drwx----------rr supergroup 0 2023-03-04 22:17 /user
stop-all.sh	Stop all Hadoop services (HDFS and YARN)	WARNING: Stopping all Apache Rr daemons as rr in 10 seconds. Use CTRL-C to abort. Stopping namenodes on [localhost] Stopping datanodes Stopping secondary namenodes [Ubuntu] WARN util.NativeCodeLoader: Unable to load native-rr library...
hdfs dfs - chmod [permissions] [path]	Change file permissions of a file or directory	Example: hdfs dfs -chmod 755 /Sojana
hdfs dfs - chown [owner] [path]	Change the owner of a file or directory	Example: hdfs dfs -chown newowner /Sojana
hdfs dfs - chgrp [group] [path]	Change the group of a file or directory	Example: hdfs dfs -chgrp newgroup /Sojana
hdfs dfs -mv [source] [destination]	Move or rename a file or directory	Example: hdfs dfs -mv /Sojana/oldname.txt /Sojana/newname.txt
hdfs dfs -rm [path]	Remove a file	Example: hdfs dfs -rm /Sojana/oldfile.txt
hdfs dfs -rm -r [path]	Remove a directory and its contents recursively	Example: hdfs dfs -rm -r /Sojana/oldfolder

Command	Description	Output/Notes
hdfs dfs -df -h	Show the amount of free space in the filesystem, in human-readable format	Example Output: Filesystem Size Used Available Use% hdfs://localhost:54310 100G 10G 90G 10%
hdfs dfsadmin -report	Report basic statistics of the HDFS	Includes the number of live and dead DataNodes, total capacity, used space, etc.
hdfs dfs - tail [path]	Display the last kilobyte of the file	Example: hdfs dfs -tail /Sojana/logfile.txt
hdfs dfs -appendToFile [localfile] [hdfsfile]	Append a local file to an existing file in HDFS	Example: hdfs dfs -appendToFile local.txt /Sojana/remote.txt
hdfs dfs - text [path]	Display the contents of a file in human-readable format	Useful for viewing contents of compressed files
hdfs dfs - expunge	Empty the HDFS trash	Moves the current trash checkpoint to a final trash location and then deletes it
hdfs dfs - stat [option] [path]	Display statistics about the file or directory	Example: hdfs dfs -stat %b /Sojana/DATA.txtto display the block size
hdfs dfs -appendToFile [localfile] [hdfsfile]	Append a local file to an existing file in lo HDFS	Example: hdfs dfs -appendToFile cal.txt /Sojana/remote.txt

hdfs dfs - text [path]	Display the contents of a file in human-readable format	Useful for viewing contents of compressed files
hdfs dfs - expunge	Empty the HDFS trash	Moves the current trash checkpoint to a final trash location and then deletes it
hdfs dfs - stat [option] [path]	Display statistics about the file or directory	Example: hdfs dfs -stat %b /Sojana/DATA.txtto display the block size

Lab 2: Write a Mapreduce program using single reduce function for finding Maximum and Minimum Number

Objective: Implement a MapReduce program to find the maximum and minimum numbers in a dataset.

Python Code for Mapper.py and Reducer.py

Mapper (mapper.py)

```
#!/usr/bin/env python
import sys


for line in sys.stdin:
    # Split the line into words
    words = line.strip().split()


    for word in words:
        # Emit word as the key and 1 as the
        value print(f"{word}\t1")
```

Reducer (reducer.py)

```
#!/usr/bin/env python
import sys
current_word = None
current_count = 0


for line in sys.stdin:
    word, count =
     line.strip().split('\t') if
     current_word == word:
        current_count +=
    int(count) else:
        if current_word:
            # Emit the word and its total count
```

```
            print(f"{current_word}\t{current_count}")
        current_word = word
        current_count = int(count)

if current_word:
    # Don't forget the last word if needed
    print(f"{current_word}\t{current_count}")
```

Execute Plan for MapReduce Job in Hadoop

Step 1: Create a directory for your input data on HDFS and upload your input file to it.

For example:

hdfs dfs -mkdir -p /user/your_username/input

hdfs dfs -put /path/to/your/input/file.txt

/user/your_username/input/ **Step 2: Create a directory for the**

output of your MapReduce job: hdfs dfs -mkdir

/user/your_username/output

Run the MapReduce job using the hadoop jar

command: hadoop jar /path/to/hadoop-

streaming.jar \

-input /user/your_username/input \

-output /user/your_username/output \

-mapper /path/to/mapper.py \

-reducer /path/to/reducer.py

Make sure to replace /path/to/hadoop-streaming.jar, /path/to/mapper.py, and

/path/to/reducer.py with the actual paths to these files in your environment. Also, adjust the input and output paths accordingly.

Once the job is finished, you can retrieve the results from the HDFS output directory:

Step 3: hdfs dfs -cat /user/your_username/output/part-00000

This command will display the word counts on the console.

The detailed execution of these commands in the Hadoop environment (rr) is given below:

rr@ubuntu:~$ start-all.sh

WARNING: Attempting to start all Apache Rr daemons as rr in 10 seconds.

WARNING: This is not a recommended production deployment configuration.

WARNING: Use CTRL-C to abort. Starting namenodes on [0.0.0.0] Starting datanodes

Starting secondary namenodes [ubuntu]

2023-03-04 21:29:38,334 WARN util.NativeCodeLoader: Unable to load native-rr library for your platform... using builtin-java classes where applicable

Starting resourcemanager Starting nodemanagers rr@ubuntu:~$ mkdir Bda rr@ubuntu:~$ cd Bda

rr@ubuntu:~/Bda$ gedit minMaxMap.py

import sys

for line in sys.stdin: val = line.strip()

year = val[0:4]temp = val[5:9] if temp != '9999':

print('%s\t%s' % (year, temp)) rr@ubuntu:~/Bda$ gedit minMaxReduce.py from operator import itemgetter

import sys

(last_key, max_val, min_val) = (None, 0, 0) for line in

sys.stdin: (key, val) = line.strip().split("\t") if last_key and

last_key != key:

print('%s\t%s\t%s' % (last_key, max_val, min_val)) (last_key, max_val, min_val) = (key, float(val), float(val))

```
 if last_key:
 else:
 (last_key, max_val, min_val) = (key,max(max_val,
float(val)), min(min_val, float(val))) if min_val ==
0.0: min_val = max_val
 print('%s\t%s\t%s' % (last_key, max_val, min_val))
```

rr@ubuntu:~/Bda$ gedit minmax.txt

```
2011    41
2011    72
2011    55
2011    43
2012    51
2012    62
2014    23
2015    45
2016    47
2016    65
2017    43
2017    23
2018    60
2019    65
2022    78
2022    45
2023    22
```

rr@ubuntu:~/Bda$ ls

minMaxMap.py minMaxReduce.py minmax.txt

rr@ubuntu:~/Bda$ cat minmax.txt | python minMaxMap.py | python minMaxReduce.py

2011 72.0 41.0

2011 55.0 43.0

2012 62.0 51.0

2014 23.0 23.0

2015 45.0 45.0

2016 65.0 47.0

2017 43.0 23.0

2018 60.0 60.0

2019 65.0 65.0

2022 78.0 45.0

2023 22.0 22.0

rr@ubuntu:~/Bda$ hdfs dfs -mkdir /data

2023-03-04 21:44:12,439 WARN util.NativeCodeLoader: Unable to load native-rr library for your platform... using builtin-java classes where applicable

rr@ubuntu:~/Bda$ hdfs dfs -ls /

2023-03-04 21:44:25,881 WARN util.NativeCodeLoader: Unable to load native-rr library for your platform... using builtin-java classes where applicable

Found 2 items

drwxr-xr-x - rr supergroup0 2023-03-04 21:44

/data drwxr-xr-x - rr supergroup0 2023-03-04 21:43

/Sojana rr@ubuntu:~/Bda$ hdfs dfs -put

minMaxReduce.py /data

2023-03-04 21:45:16,642 WARN util.NativeCodeLoader: Unable to load native-rr library for your platform... using builtin-java classes where applicable

rr@ubuntu:~/Bda$ hdfs dfs -put minMaxMap.py /data

2023-03-04 21:45:40,287 WARN util.NativeCodeLoader: Unable to load native-rr library for your platform... using builtin-java classes where applicable

rr@ubuntu:~/Bda$ hdfs dfs -put minmax.txt /data

2023-03-04 21:46:10,971 WARN util.NativeCodeLoader: Unable to load native-rr library for your platform... using builtin-java classes where applicable

rr@ubuntu:~/Bda$ hdfs dfs -ls /data/

2023-03-04 21:46:26,374 WARN util.NativeCodeLoader: Unable to load native-rr library for your platform... using builtin-java classes where applicable

Found 3 items

-rw-r--r-- 1 rr supergroup 161 2023-03-04 21:45

/data/minMaxMap.py

-rw-r--r-- 1 rr supergroup 554 2023-03-04 21:45

/data/minMaxReduce.py

-rw-r--r-- 1 rr supergroup 136 2023-03-04 21:46 /data/minmax.txt

rr@ubuntu:~/Bda$ rr jar /usr/local/rr/share/rr/

tools/lib/rr-streaming-*.jar -file minMaxMap.py -mapper "python minMaxMap.py" -file minMaxReduce.py -reducer "python minMaxReduce.py" -input /data/minmax.txt -output /data/P02RESULT

rr@ubuntu:~/Bda$ hdfs dfs -ls /data/

2023-03-04 22:01:01,336 WARN util.NativeCodeLoader: Unable to load native-rr library for your platform... using builtin-java classes where applicable

Found 4 items

drwxr-xr-x - rr supergroup0 2023-03-04 21:58

/data/P02RESULT

-rw-r--r-- 1 rr supergroup 161 2023-03-04 21:45

/data/minMaxMap.py

-rw-r--r-- 1 rr supergroup 554 2023-03-04 21:45

/data/minMaxReduce.py

-rw-r--r-- 1 rr supergroup 136 2023-03-04 21:46 /data/minmax.txt

rr@ubuntu:~/Bda$ hdfs dfs -ls /data/P02RESULT/

2023-03-04 22:02:41,070 WARN util.NativeCodeLoader: Unable to load native-rr library for your platform. using builtin-java classes

where applicable Found 2 items

-rw-r--r-- 1 rr supergroup 0 2023-03-04 21:58

/data/P02RESULT/_SUCCESS

-rw-r--r-- 1 rr supergroup 165 2023-03-04 21:58

/data/P02RESULT/part-00000

rr@ubuntu:~/Bda$ hdfs dfs -cat /data/P02RESULT/part-*

2023-03-04 22:03:05,017 WARN util.NativeCodeLoader: Unable to load native-rr library for your platform... using builtin-java classes where applicable

2011 72.0 41.0

2011 55.0 43.0

2012 62.0 51.0

2014 23.0 23.0

2015 45.0 45.0

2016 65.0 47.0

2017 43.0 23.0

2018 60.0 60.0

2019 65.0 65.0

2022 78.0 45.0

2023 22.0 22.0

rr@ubuntu:~/Bda$ stop-all.sh

WARNING: Stopping all Apache Rr daemons as rr in 10 seconds.
WARNING: Use CTRL-C to abort.

Stopping namenodes on [0.0.0.0] Stopping

datanodes Stopping secondary namenodes

[ubuntu]

Lab 3: Write a Mapreduce program using multiple reduce function for Word Count in an given Text document

Objective: Implement a MapReduce program to perform word count on a given text document using multiple reduce function.

Python code : Mapper and Reducer for Word Count

Wordcount_Mapper.py

```
#!/usr/bin/env python
import sys
# Input: Text
document for line in
sys.stdin:
    # Tokenize the line into
    words words =
    line.strip().split()

    # Emit each word with a count of
    1 for word in words:
        print(f"{word}\t1")
```

Wordcount_Reducer.py

```
#!/usr/bin/env python
import sys
current_word = None
current_count = 0

# Input: Key-Value pairs from the
Mapper for line in sys.stdin:
    word, count = line.strip().split('\t')

    # Convert the count to an
    integer count = int(count)
```

```
    if current_word ==
        word: current_count
        += count
    else:
        if current_word:
            # Emit the word and its total count
            print(f"{current_word}\t{current_count}")
        current_word = word
        current_count = count

# Don't forget the last word if
needed if current_word:
    print(f"{current_word}\t{current_count}")
```

Executing the MapReduce job

To execute the Word Count MapReduce job on a given text document, follow these steps:

Step 1: Create a directory for your input data on HDFS and upload your text document to it. For example:

hdfs dfs -mkdir -p /user/your_username/input

hdfs dfs -put /path/to/your/input/text.txt

/user/your_username/input/ **Step 2: Create a directory for the**

output of your MapReduce job: hdfs dfs -mkdir output

Run the MapReduce job using the hadoop jar command.

hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-<version>.jar \

-input input \

-output output \

-mapper /path/to/mapper.py \

-reducer /path/to/reducer.py \

-file /path/to/mapper.py \

-file /path/to/reducer.py \

-numReduceTasks <num_reducers>

Replace /path/to/mapper.py and /path/to/reducer.py with the actual paths to your Mapper and Reducer Python scripts. Adjust the number of reducers (<num_reducers>) as needed.

This command will display the result of the matrix multiplication on the console

Step 3: Retrieve the Word Count results from the HDFS output directory:

hdfs dfs -cat /user/your_username/output/part-

00000 This will display the word counts on the

console.

The detailed execution of these commands are as follows:

rr@ubuntu:~$ start-all.sh

WARNING: Attempting to start all Apache Rr daemons as rr in 10 seconds.

WARNING: This is not a recommended production deployment configuration.

WARNING: Use CTRL-C to abort. Starting namenodes on [0.0.0.0] Starting datanodes

Starting secondary namenodes [ubuntu]

2023-03-05 04:56:47,216 WARN util.NativeCodeLoader: Unable to load native-rr library for your platform... using builtin-java classes where applicable

Starting resourcemanager Starting nodemanagers rr@ubuntu:~$ gedit words.txt

Big data analytics is the use of advanced analytic techniques against very large, diverse data sets that include structured, semi-structured and unstructured data, from different sources, and in different sizes from terabytes to zettabytes

rr@ubuntu:~$ gedit wordMap.py

import sys

for line in sys.stdin: line=line.strip()

words=line.split() for word in words:

print('%s\t%s'%(word,"1")) rr@ubuntu:~$ gedit

wordReduce.py

import sys wordcount = {}

for line in sys.stdin: line =

line.strip() word, count =

line.split() try:

wordcount[word] = wordcount[word] + int(count) except:

wordcount[word] = int(count) for word in wordcount.keys():

print('%s\t%s' % (word, wordcount[word]))

rr@ubuntu:~$ hdfs dfs -mkdir -p /bigdata2

2023-03-05 05:19:47,272 WARN util.NativeCodeLoader: Unable to load native-rr library for your platform... using builtin-java classes where applicable

rr@ubuntu:~$ hdfs dfs -ls /

2023-03-05 05:20:00,556 WARN util.NativeCodeLoader: Unable to load native-rr library for your platform... using builtin-java classes where applicable

Found 4 items

drwxr-xr-x - rr supergroup0 2023-03-04 21:57 /data

drwxr-xr-x - rr supergroup0 2023-03-05 05:19

/bigdata2

drwxr-xr-x - rr supergroup0 2023-03-04 21:43

/Sojana drwx------ - rr supergroup0 2023-03-04

21:52 /tmp **rr@ubuntu:~$ hdfs dfs -chmod 700**

/bigdata2

2023-03-05 05:20:26,829 WARN util.NativeCodeLoader: Unable to load native-rr library for your platform... using builtin-java classes where applicable

rr@ubuntu:~$ hdfs dfs -put words.txt /bigdata2

2023-03-05 05:21:14,052 WARN util.NativeCodeLoader: Unable to load native-rr library for your platform... using builtin-java classes where applicable

rr@ubuntu:~$ hdfs dfs -put wordMap.py /bigdata2

2023-03-05 05:21:33,014 WARN util.NativeCodeLoader: Unable to load native-rr library for your platform... using builtin-java classes where applicable

rr@ubuntu:~$ hdfs dfs -put wordReduce.py /bigdata2

2023-03-05 05:21:47,709 WARN util.NativeCodeLoader: Unable to load native-rr library for your platform... using builtin-java classes where applicable

rr@ubuntu:~$ hdfs dfs -ls /bigdata2

2023-03-05 05:22:00,937 WARN util.NativeCodeLoader: Unable to load native-rr library for your platform... using builtin-java classes where applicable

Found 3 items

-rw-r--r-- 1 rr supergroup 159 2023-03-05 05:21

/bigdata2/wordMap.py

-rw-r--r-- 1 rr supergroup 342 2023-03-05 05:21

/bigdata2/wordReduce.py

-rw-r--r-- 1 rr supergroup 242 2023-03-05 05:21

/bigdata2/words.txt

rr@ubuntu: ~$ cat words.txt | python wordMap.py | sort | python wordReduce.py

and 2

structured, 1

semi-structured 1

is 1

terabytes 1

analytics 1

diverse 1

in 1

techniques 1

data, 1

different 2

from 2

to 1

Big 1

analytic 1

large,

1

include

1 that

1

very 1

use 1

unstructured 1

data 2

advanced 1

zettabytes 1

sources, 1

sizes 1

of 1

against 1

sets 1

the 1

rr@ubuntu: ~$ rr jar /usr/local/rr/share/rr/

tools/lib/rr-streaming-*.jar -D mapred.reduce.tasks=2 -file wordMap.py -mapper "python wordMap.py" -file wordReduce.py - reducer "python wordReduce.py" -input /bigdata2/words.txt -output /bigdata2/WCResult rr@ubuntu: ~$ hdfs dfs -ls /bigdata2/WCResult

2023-03-05 05:30:11,919 WARN util.NativeCodeLoader: Unable to load native-rr library for your platform. using builtin-java classes where
applicable

Found 3 items

-rw-r--r-- 1 rr supergroup 0 2023-03-05 05:29

/bigdata2/WCResult/_SUCCESS

-rw-r--r-- 1 rr supergroup 141 2023-03-05 05:29

/bigdata2/WCResult/part-00000

-rw-r--r-- 1 rr supergroup 135 2023-03-05 05:29

/bigdata2/WCResult/part-00001

rr@ubuntu:~$ hdfs dfs -cat /bigdata2/WCResult/part*

2023-03-05 05:31:28,786 WARN util.NativeCodeLoader: Unable to load native-rr library for your platform. using builtin-java classes where applicable

and 2

different 2

zettabytes 1

structured, 1

to 1

that 1

of 1

terabytes 1

sets 1

use 1

against 1

analytic 1

large, 1

in 1

the 1

techniques 1

data, 1

from 2

sources, 1

sizes 1

very 1

Big 1

is 1

unstructured 1

analytics 1

diverse 1

semi-structured 1

include

1 data 2

advanced 1

rr@ubuntu:~$ stop-all.sh

WARNING: Stopping all Apache Rr daemons as rr in 10 seconds.
WARNING: Use CTRL-C to abort.

Stopping namenodes on [0.0.0.0] Stopping

datanodes Stopping secondary namenodes

[ubuntu]

2023-03-05 05:32:11,114 WARN util.NativeCodeLoader: Unable to load native-rr library for your platform using builtin-java classes where applicable

Stopping nodemanagers Stopping resourcemanager

Lab 4: Implement the following using Pig Latin : Input and Output Operations and Relational Operations

Objective: Learn Pig Latin basics, including input/output and relational operations.

Part-I :Input and Output Operations in Pig Latin

1. Load Data (Load):

The LOAD statement is used to load data from various sources like HDFS, local file systems, or other storage systems.

Example: Loading data from an HDFS file.

```
data = LOAD 'hdfs:///user/hadoop/data.txt' USING PigStorage(',') AS (name:chararray, age:int, city:chararray);
```

2. Store Data (Store):

The STORE statement is used to store the results of Pig Latin operations into a specified location.

Example: Storing data in HDFS.

```
STORE data INTO 'hdfs:///user/hadoop/output' USING PigStorage(',');
```

Part II:Relational Operations in Pig Latin

3. Filter (FILTER):

The FILTER operator is used to select rows from a relation based on a specified condition.

Example: Filtering data to select people older than 30.

```
filtered_data = FILTER data BY age > 30;
```

4. Group (GROUP):

The GROUP operator groups data based on one or more columns.

Example: Grouping data by the 'city' column.

grouped_data = GROUP data BY city;

5. Foreach (FOREACH):

The FOREACH operator is used to apply transformations to the data within each group.

Example: Calculating the average age in each city group.

avg_age = FOREACH grouped_data GENERATE group AS city, AVG(data.age) AS avg_age;

Part III: Advanced Relational Operator

6. Join (JOIN):

The JOIN operator combines two or more relations based on a common column.

Example: Joining data from two relations.

joined_data = JOIN data1 BY id, data2 BY

id;

7. Union (UNION):

The UNION operator combines two or more relations with the same

schema. Example: Combining data from two relations with the same

schema. combined_data = UNION data1, data2;

8. Order (ORDER):

The ORDER operator sorts the data based on one or more

columns. Example: Sorting data by age in descending order.

sorted_data = ORDER data BY age DESC;

The detailed execution of these commands in the pig environment is given below:

hadoop@ubuntu:~$ start-all.sh

WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds.

WARNING: This is not a recommended production deployment configuration.

WARNING: Use CTRL-C to abort. Starting namenodes on [0.0.0.0] Starting datanodes

Starting secondary namenodes [ubuntu]

2023-04-12 07:26:04,750 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Starting resourcemanager Starting nodemanagers

hadoop@ubuntu:~$ gedit dels.txt

1,Elan,25,A

2,Dhanush,22,B

3,Lachu,30,C

4,Sanjai,27,B

5,Akil,24,A

hadoop@ubuntu:~$ hdfs dfs -mkdir /BDA

2023-04-12 06:42:59,540 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

hadoop@ubuntu:~$ hdfs dfs -chmod 755 /BDA

2023-04-12 06:43:37,776 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

hadoop@ubuntu:~$ hdfs dfs -put dels.txt /BDA

2023-04-12 07:34:08,843 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

hadoop@ubuntu:~$ hdfs dfs -ls /BDA

2023-04-12 07:34:30,977 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Found 4 items

-rw-r--r-- 1 hadoop supergroup 66 2023-04-12 07:24 /BDA/dels.txt
hadoop@ubuntu:~$ pig

grunt> A = LOAD '/BDA/dels.txt ' USING PigStorage(',') as (id:int,name:chararray,age:int,grade:chararray);

grunt> DUMP A;

(1,Elan,25,A)

(2,Dhanush,22,B)

(3,Lachu,30,C)

(4,Sanjai,27,B)

(5,Akil,24,A)

grunt> fewrecords = LIMIT A 2; grunt> DUMP fewrecords;

(1,Elan,25,A)

(2,Dhanush,22,B)

grunt> agelimit = FILTER A BY age>25; grunt> DUMP agelimit;

(3,Lachu,30,C)

(4,Sanjai,27,B)

grunt> DESCRIBE A;

A: {id: int,name: chararray,age: int,grade: chararray}

grunt> groupbyage = GROUP A BY age; grunt> DUMP groupbyage;

(22,{(2,Dhanush,22,B)})

(24,{(5,Akil,24,A)})

(25,{(1,Elan,25,A)})

(27,{(4,Sanjai,27,B)})

(30,{(3,Lachu,30,C)})

grunt> maxage = FOREACH groupbyage GENERATE(A.id, A.age), MAX(A.age);

grunt> DUMP maxage;

(({(2)},{(22)}),22)

(({(5)},{(24)}),24)

(({(1)},{(25)}),25)

(({(4)},{(27)}),27)

(({(3)},{(30)}),30)

grunt> allagevalues = GROUP A ALL;

grunt> DUMP allagevalues;

(all,{(5,Akil,24,A),(4,Sanjai,27,B),(3,Lachu,30,C),(2,Dhan
ush,22,B),(1,Elan,25,A)})

grunt> sumofage = FOREACH allagevalues GENERATE group,SUM(A.age);

grunt> dump sumofage;

(all,128)

grunt> dist = DISTINCT A;

grunt> DUMP dist;

(1,Elan,25,A)

(2,Dhanush,22,B)

(3,Lachu,30,C)

(4,Sanjai,27,B)

(5,Akil,24,A)

grunt> quit

hadoop@ubuntu:~$ stop-

all.sh

WARNING: Stopping all Apache Hadoop daemons as hadoop in 10 seconds. WARNING: Use CTRL-C to abort.

Stopping namenodes on [0.0.0.0] Stopping

datanodes Stopping secondary namenodes

[ubuntu]

2023-04-12 08:11:41,223 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Stopping nodemanagers Stopping resourcemanager

Lab 5: Implement the following using Pig Latin: User Defined Functions and Advanced Relational Operations

Objective: Explore more advanced operations and user-defined functions with Pig Latin.

Part1: User Defined Function in pig

Step 1: Start Hadoop Services

1. **Command:**

   hadoop@ubuntu:~$ start-all.sh

2. **Output:**

   WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds.
   WARNING: This is not a recommended production deployment configuration.
   WARNING: Use CTRL-C to abort.
   Starting namenodes on [0.0.0.0]
   Starting datanodes
   Starting secondary namenodes [ubuntu]
   2023-04-12 06:37:01,144 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
   Starting
   resourcemanager
   Starting
   nodemanagers

Step 2: Create and Edit Input Files

1. **Create employee.txt File:**

   hadoop@ubuntu:~$ gedit employee.txt

2. **Content of employee.txt:**

10012,Santhosh,25,35000
10021,Elanchezhian,26,45250
12045,Suresh,31,85000
14502,Karthi,23,65000
18545,Selva,21,15000

3. **Create udf.py File:**

   hadoop@ubuntu:~$ gedit udf.py

4. **Content of udf.py:**

   @outputSchema('num:long
   ') def get_length(data):
      str_data = ''.join([chr(x) for x in
      data]) return len(str_data)

Step 3: Upload Files to HDFS

1. **Upload employee.txt to HDFS:**

   hdfs dfs -put employee.txt /BDA

2. **Output:**

   2023-04-12 06:43:15,455 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

3. **Upload udf.py to HDFS:**

   hdfs dfs -put udf.py /BDA

4. **Output:**

   2023-04-12 06:43:26,792 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

5. **List Files in HDFS:**

   hdfs dfs -ls /BDA

6. **Output:**

   2023-04-12 06:44:01,322 WARN util.NativeCodeLoader:
   Unable to load native-hadoop library for your platform... using
   builtin-java classes where applicable
   Found 2 items
   -rw-r--r-- 1 hadoop supergroup 117 2023-04-12 06:43
   /BDA/employee.txt
   -rw-r--r-- 1 hadoop supergroup 120 2023-04-12 06:43 /BDA/udf.py

Step 4: Execute Pig Script with UDF

1. **Open Pig:**

   hadoop@ubuntu:~$ pig

2. **Pig Grunt Shell Commands:**

   grunt> REGISTER 'udf.py' USING jython as pyudf;
   grunt> A = LOAD '/BDA/employee.txt' USING PigStorage(',');
   grunt> B = FOREACH A GENERATE $0, pyudf.get_length($0);
   grunt> DUMP B;

3. **Expected Output:**

   (10012,Santhosh,25,35000,23)
   (10021,Elanchezhian,26,45250,27)
   (12045,Suresh,31,85000,21)
   (14502,Karthi,23,65000,21)
   (18545,Selva,21,15000,20)

4. **Exit Pig:**

   grunt> quit;

Part II: Advanced Relational Operators

hadoop@ubuntu:~$ gedit emp_Del.txt

10012,Salem,MCA,Developer

10021,Erode,MCA,CEO

12045,Trichy,MCA,Manager 14502,Rahul,B.E,GM

18545,Rehan,M.E,MD

hadoop@ubuntu:~$ hdfs dfs -put emp_Del.txt /BDA
hadoop@ubuntu:~$ pig

grunt> A = LOAD '/BDA/employee.txt' USING PigStorage(',') AS (id:int,name:chararray,age:int,Salary:int);

grunt> B = LOAD '/BDA/emp_Del.txt ' USING PigStorage(',') AS (id:int,location:chararray,Qualification:chararray,Designation:charray);

grunt> join_data = JOIN A by id,B by id;

(10012,Santhosh,25,35000,10012,Salem,MCA,Developer)
(10021,Elanchezhian,26,45250,10021,Erode,MCA,CEO)
(12045,Suresh,31,85000,12045,Trichy,MCA,Manager)
(14502,Karthi,23,65000,14502,Rahul,B.E,GM)
(18545,Selva,21,15000,18545,Rehan,M.E,MD)

grunt> SPLIT A INTO X IF age==25, Y IF age<=22;

grunt> dump X;

(10012,Santhosh,25,35000)

grunt> dump Y;

(18545,Selva,21,15000)

grunt> grade_grp = GROUP A BY name;

grunt> grade_avg = FOREACH grade_grp GENERATE A.name, A.age, AVG(A.Salary);

grunt> dump grade_avg;

({(Selva)},{(21)},15000.0)

({(Karthi)},{(23)},65000.0)

({(Suresh)},{(31)},85000.0)

({(Santhosh)},{(25)},35000.0)

({(Elanchezhian)},{(26)},45250.0)

grunt> cogroup_data = COGROUP A BY id,B BY

id; grunt> dump cogroup_data;

(10012,{(10012,Santhosh,25,35000)},{(10012,Salem,MCA,Developer)})
(10021,{(10021,Elanchezhian,26,45250)},{(10021,Erode,MCA,CEO)})
(12045,{(12045,Suresh,31,85000)},{(12045,Trichy,MCA,Manager)})
(14502,{(14502,Karthi,23,65000)},{(14502,Rahul,B.E,GM)})

(18545,{(18545,Selva,21,15000)},{(18545,Rehan,M.E,MD)})

grunt> M = FOREACH A GENERATE TOMAP(name, age); grunt> dump M;

([Santhosh#25])

([Elanchezhian#26])

([Suresh#31]) ([Karthi#23])

([Selva#21])

grunt> quit; hadoop@ubuntu:~$ stop- all.sh

WARNING: Stopping all Apache Hadoop daemons as hadoop in 10 seconds. WARNING: Use CTRL-C to abort.

Stopping namenodes on [0.0.0.0] Stopping datanodes

Stopping secondary namenodes [ubuntu]

2023-04-12 07:24:45,390 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Stopping nodemanagers Stopping resourcemanager

Lab 6: Write a Word Count program using Pig Latin Script

Objective: To learn and practice the basics of word counting in Hadoop using Pig Latin.

Create a Pig Latin Script

Step 1: Start Pig Latin

Open your terminal and start Pig Latin by running the pig command.

Step 2: Write the Pig Latin Script

Create a Pig Latin script to find the word count in the given text.

wordcount.pig

-- Load the input text file

lines = LOAD 'input.txt' AS (line: chararray);

-- Tokenize the lines into words

words = FOREACH lines GENERATE FLATTEN(TOKENIZE(line)) AS word;

-- Group and count the words

word_counts = GROUP words BY

word;

word_count = FOREACH word_counts GENERATE group AS word, COUNT(words) AS count;

-- Order the results by word count in descending order

sorted_word_count = ORDER word_count BY count DESC;

-- Display the result

DUMP sorted_word_count;

Note: Save this script to a .pig file, and then you can execute it using the Pig Latin interpreter. This script will count the occurrences of each word in the input text file and display the results.

Step 3: Run the Pig Script

Execute your Pig Latin script using the pig command. Provide the script filename as an argument.

pig wordcount.pig

The detailed execution of these commands in the pig environment is given below:

hadoop@ubuntu:~$ start-all.sh

WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds.

WARNING: This is not a recommended production deployment configuration.

WARNING: Use CTRL-C to abort. Starting namenodes on [0.0.0.0] Starting datanodes

Starting secondary namenodes [ubuntu]

2023-04-05 07:39:45,309 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Starting resourcemanager Starting nodemanagers

hadoop@ubuntu:~$ gedit wCount.txt

Pig Represents Big Data as data flows.

Pig is a high-level platform or tool which is used to process the large

datasets. It provides a high-level of abstraction for processing over the

MapReduce.

It provides a high-level scripting language, known as Pig Latin which is used to develop the data analysis codes**.**

hadoop@ubuntu:~$ hdfs dfs -put wCount.txt /elan

2023-04-05 07:44:11,347 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

hadoop@ubuntu:~$ hdfs dfs -ls /elan

2023-04-05 07:44:25,000 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Found 1 items

-rw-r--r-- 1 hadoop supergroup 313 2023-04-05 07:44 /elan/wCount.txt

hadoop@ubuntu:~$ gedit wCount.pig

Input_Data = LOAD '/elan/wCount.txt' AS (line:chararray); tozkenized_data = FOREACH Input_Data GENERATE FLATTEN(TOKENIZE(line)) AS word;

grouped_data = GROUP tozkenized_data BY word;

word_count = FOREACH grouped_data GENERATE group AS word, COUNT(tozkenized_data) AS count;

DUMP ordered_data;

hadoop@ubuntu:~$ pig

wCount.pig (a,3)

(It,2)

(as,2)

(is,3)

(of,1)

(or,1)

(to,2)

(Big,1)

(Pig,3)

(for,1)

(the,3)

(Data,1)

(data,2)

(over,1)

(tool,1)

(used,2)

(Latin,1)

(known,1)

(large,1)

(which,2)

(codes.,1)

(flows.,1) (develop,1) (process,1)

(analysis,1) (language,1) (platform,1) (provides,2) (datasets.,1) (scripting,1) (MapReduce.,1) (Represents,1) (high-level,3) (processing,1) (abstraction,1)

hadoop@ubuntu:~$ stop-all.sh

WARNING: Stopping all Apache Hadoop daemons as hadoop in 10 seconds. WARNING: Use CTRL-C to abort.

Stopping namenodes on [0.0.0.0] Stopping

datanodes Stopping secondary namenodes

[ubuntu]

2023-04-05 07:39:45,309 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Stopping nodemanagers Stopping resourcemanager

Lab 7: Write a program to find a maximum temperature using Pig Latin Script

Objective: To learn and practice to find the maximum temperature from a dataset using Pig Latin.

Create a Pig Latin Script

Step 1: Start Pig Latin

Open your terminal and start Pig Latin by running the pig command.

Step 2: Write the Pig Latin Script

Create a Pig Latin script (e.g., max_temperature.pig) to find the maximum temperature in the dataset.

-- Load the input dataset

data = LOAD 'temperature_data.txt' USING PigStorage(',') AS (date:chararray, temperature:float);

-- Find the maximum temperature

max_temp = FOREACH (GROUP data ALL) GENERATE MAX(data.temperature) as max_temperature;

-- Display the maximum

temperature DUMP max_temp;

Save the Pig script.

Step 3: Run the Pig Script

Execute your Pig Latin script using the pig command. Provide the script filename as an argument.

pig max_temperature.pig

The detailed execution of these commands is given below:

hadoop@ubuntu:~$ start-all.sh

WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds.

WARNING: This is not a recommended production deployment configuration.

WARNING: Use CTRL-C to abort. Starting namenodes on [0.0.0.0] Starting datanodes

Starting secondary namenodes [ubuntu]

2023-04-05 07:39:45,309 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Starting resourcemanager Starting nodemanagers

hadoop@ubuntu:~$ gedit minMax.txt

2000 45

2000 15

2001 26

2002 24

2003 21

2003 36

2004 26

hadoop@ubuntu:~$ hdfs dfs -put minMax.txt /BigData

2023-04-05 08:20:55,357 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

hadoop@ubuntu:~$ hdfs dfs -ls /BigData

2023-04-05 08:28:38,199 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Found 6 items

drwxr-xr-x - hadoop supergroup

/BigData/P02RESULT 0 2023-03-04 21:58

-rw-r--r-- 1 hadoop supergroup 65 2023-04-05 08:20

/BigData/minMax.txt

-rw-r--r-- 1 hadoop

supergroup 161 2023-03-04

21:45

/BigData/minMaxMapper.py

-rw-r--r-- 1 hadoop

supergroup 554 2023-03-04

21:45

/BigData/minMaxReducer.py

-rw-r--r-- 1 hadoop supergroup 136 2023-03-04 21:46

/BigData/temp.txt

-rw-r--r-- 1 hadoop

supergroup 131 2023-04-05

07:11

/BigData/wordCount.txt

hadoop@ubuntu:~$ gedit temp.pig

```
A = LOAD '/BigData/minMax.txt' using PigStorage (' ') AS
(year:int,temp:int);

B = GROUP A BY year;

C = FOREACH B GENERATE group, MAX(A.temp), MIN(A.temp);
DUMP C;
```

hadoop@ubuntu:~$ pig temp.pig

(2000,45,15)

(2001,26,26)

(2002,24,24)

(2003,36,21)

(2004,26,26)

(2005,37,37)

hadoop@ubuntu:~$ stop-all.sh

WARNING: Stopping all Apache Hadoop daemons as hadoop in 10 seconds. WARNING: Use CTRL-C to abort.

Stopping namenodes on [0.0.0.0] Stopping

datanodes Stopping secondary namenodes

[ubuntu]

2023-04-05 07:39:45,309 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Stopping nodemanagers Stopping resourcemanager

Lab 8: Implement the following using Hive commands : i) Handling the Database ii) Creating and Manipulating table

Objective: Learn to use Hive commands to handle databases and create/manipulate tables.

i. Handling the Database

Step 1: Start Hive

Open your terminal and start Hive by running the hive command.

Step 2: Create a new database using the CREATE DATABASE command:

CREATE DATABASE mydatabase;

Step 3: Perform the following commands in the created

database. Show Databases:

To list all available databases in Hive, you can use the SHOW DATABASES;

SHOW DATABASES;

Use a Database:

To switch to a specific database for your session, use the USE command.

USE mydatabase;

Show Tables:

To list all tables in the current database, use the SHOW TABLES; command.

SHOW TABLES;

ii. Creating and Manipulating table

Hive Table Creation and Management Create a Table:

To create a new table, use the CREATE TABLE command, specifying the table name, columns, and their data types.

```
CREATE TABLE mytable
  ( id INT,
  name
  STRING, age
  INT
);
```

Describe a Table:

To view the table schema, use the DESCRIBE command.

```
DESCRIBE mytable;
```

Load Data into a Table:

You can load data into a table from an existing file or directory using the LOAD DATA command.

```
LOAD DATA INPATH '/path/to/data/file' INTO TABLE mytable;
```

Insert Data into a Table:

To insert data into a table using INSERT INTO, you can use a SELECT statement to specify the data source.

```
INSERT INTO TABLE mytable SELECT * FROM myothertable;
```

Hive Table :Querying Select the data:

SELECT name, age FROM mytable WHERE age > 30;

Aggregate Data:

Hive supports aggregate functions like SUM, AVG, COUNT, and more for data analysis.

SELECT AVG(age) AS avg_age FROM mytable;

Join Tables:

You can perform JOIN operations to combine data from multiple

tables. SELECT t1.name, t2.city FROM table1 t1 JOIN table2 t2 ON

t1.id = t2.id;

Hive Table and Database Management

Alter a Table:

Use the ALTER TABLE command to make changes to an existing table, such as adding, dropping, or renaming columns.

ALTER TABLE mytable ADD COLUMNS (email STRING);

Drop a Table:

To delete a table, use the DROP TABLE

command. DROP TABLE mytable;

Drop a Database:

To remove a database, including all its tables, use the DROP DATABASE command.

DROP DATABASE mydatabase;

The detailed execution of these Hive commands is given below:

rr@ubuntu:~$ start-all.sh

WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds.

WARNING: This is not a recommended production deployment configuration.

WARNING: Use CTRL-C to abort. Starting namenodes on [0.0.0.0] Starting datanodes

Starting secondary namenodes [ubuntu]

2023-03-04 21:29:38,334 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Starting resourcemanager Starting nodemanagers

rr@ubuntu:~$ cd BDA rr@ubuntu:~/BDA$ gedit

emp.txt Elanchezhian,20,100000,CEO

Sivakumar,21,50000,Software_Devloper

Magudeshwaren,23,10000,Designer

Dhasush,22,80000,HR

Hariharan,23,4000,Cleaner

Sanjai,44,70000,Recptionist

Lachu,24,10000,Security

rr@ubuntu:~/BDA$ hive

hive> CREATE DATABASE IF NOT EXISTS ELAN;

OK

Time taken: 0.361 seconds

hive> SHOW DATABASES;

OK

default elan

kera_company

Time taken: 0.571 seconds, Fetched: 3 row(s)

hive> ALTER DATABASE ELAN SET DBPROPERTIES('CREATED BY'='ELANCHEZHIAN');

OK

Time taken: 0.295 seconds

hive> CREATE TABLE EMP_CAT(EMP_NAME STRING, AGE INT, SALARY FLOAT,EMP_ROLL STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;

OK

Time taken: 0.138 seconds

hive> LOAD DATA LOCAL INPATH '/home/rr/BDA/emp.txt' INTO TABLE EMP_CAT;

Loading data to table default.emp_cat

Table default.emp_cat stats: [numFiles=1,

totalSize=193] OK

Time taken: 0.424 seconds

hive> SELECT * FROM EMP_CAT;

OK

Elanchezhian 20 100000.0 CEO

Sivakumar 21 50000.0 Software_Devloper

Magudeshwaren 23 10000.0 Designer

Dhasush 22 80000.0 HR

Hariharan 23 4000.0 Cleaner

Sanjai 44 70000.0 Recptionist

Lachu 24 10000.0 Security

Time taken: 0.186 seconds, Fetched: 7 row(s)

hive> CREATE TABLE EMP_CAT2 LIKE EMP_CAT;

OK

Time taken: 0.177 seconds

hive> INSERT INTO TABLE EMP_CAT2 SELECT * FROM EMP_CAT;

Query ID = rr_20230327001239_f8973b2d-95eb-4d6d-9b68-473c3c68bd1e Total jobs = 3

Launching Job 1 out of 3

OK

Time taken: 27.428 seconds

hive> SELECT * FROM EMP_CAT2;

OK

Elanchezhian 20 100000.0 CEO

Sivakumar 21 50000.0 Software_Devloper

Magudeshwaren 23 10000.0 Designer

Dhasush 22 80000.0 HR

Hariharan 23 4000.0 Cleaner

Sanjai 44 70000.0 Recptionist

Lachu 24 10000.0 Security

Time taken: 0.171 seconds, Fetched: 7 row(s)

hive> SELECT * FROM EMP_CAT2 WHERE AGE > 21;

OK

Magudeshwaren 23		10000.0	Designe r
Dhasush	22	80000.0	HR
Hariharan	23	4000.0	Cleaner
Sanjai	44	70000.0	Recptionist
Lachu	24	10000.0	Security

Time taken: 0.295 seconds, Fetched: 5 row(s)

hive> SELECT * FROM EMP_CAT2 WHERE SALARY >80000;

OK

Elanchezhian 20 100000.0 CEO Time taken: 0.159 seconds, Fetched: 1 row(s)

rr@ubuntu:~$ stop-all.sh

WARNING: Stopping all Apache Hadoop daemons as hadoop in 10 seconds. WARNING: Use CTRL-C to abort.

Stopping namenodes on [0.0.0.0] Stopping

datanodes Stopping secondary namenodes

[ubuntu]

2023-03-05 05:32:11,114 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Stopping nodemanagers Stopping resourcemanager

Lab 9: Implement Simple Queries for database using Mongo

Objective: Explore MongoDB queries for database.

Connecting to MongoDB

Open your terminal or command prompt.

Start the MongoDB CLI by running the following command:

mongo

Once connected, you can switch to a specific database using the use command:

use mydatabase

Note: Replace mydatabase with the name of your database.

1. **Inserting Documents**

To insert documents into a collection, use the insertOne or insertMany methods.

For example, to insert a single document:

db.mycollection.insertOne({"name": "Alice", "age":

30}) To insert multiple documents:

db.mycollection.insertMany([

{"name": "Bob", "age": 25},

{"name": "Charlie", "age": 35}

])

2. Querying Documents

Find All Documents

To retrieve all documents in a collection, use the find method:

db.mycollection.find({})

For example: db.mycollection.find({"name":

"Alice"}) # Find documents where the "age" is

greater than 25 db.mycollection.find({"age": {"$gt":

25}})

3. Limit the Number of Results

To limit the number of results, use the limit method:

db.mycollection.find({}).limit(2)

4. Updating Documents

To update documents, use the updateOne or updateMany methods. For example:

Update a single document

db.mycollection.updateOne({"name": "Alice"}, {"$set": {"age":

31}}) # Update multiple documents

db.mycollection.updateMany({"age": {"$gt": 30}}, {"$set": {"status": "senior"}})

5. Deleting Documents

Use the deleteOne or deleteMany methods to delete

documents: # Delete a single document

db.mycollection.deleteOne({"name": "Alice"})

Delete multiple documents

db.mycollection.deleteMany({"status":

"senior"})

])

6. **Find All Documents**

To retrieve all documents in a collection, use the find method:

db.mycollection.find({})

7. **. Find Documents with a Specific Field**

To find documents with a specific field value, specify the field in the query:

db.mycollection.find({"name": "Alice"})

8. **Find Documents with Complex Conditions**

 # Find documents where the "age" is greater than 25

db.mycollection.find({"age": {"$gt": 25}})

9. **Limit the Number of Results**

To limit the number of results, use the limit method:

db.mycollection.find({}).limit(2)

10. **Update Documents**

To update documents, use the updateOne or updateMany methods.
For example:

The detailed execution of MongoDB commands is given below: test> use BDA

BDA> db.stuInfo.insertOne(

{"_id" : 1, Name:'Elanchezhian', Age:'21', Course:'MCA'}

)

BDA> db.stuInfo.insertMany([

{"_id" : 2,Name : 'Dhanush', Age : '22', Course:'MSC CS'},

{"_id" : 3,Name : 'Hariharan', Age : '20', Course: 'MSC DS'},

{"_id" : 4,Name : 'Lachu', Age : '24', Course: 'MSc IT'}

])

```
BDA> db.stuInfo.find()

BDA> db.stuInfo.updateOne({ "_id":3 }, { $set: { Course: 'MSc IT' } }) BDA> db.stuInfo.updateMany({},{$set:{Batch:'2022'}})

BDA> db.stuInfo.find()

BDA> db.stuInfo.deleteOne({Name:'Lachu'})

BDA> db.stuinfo.deleteMany({Batch:'2022'})

BDA> db.stuInfo.find()
```

OUTPUT:

- **Insert Single Documents:**

```
{ acknowledged: true, insertedId: 1 }
```

- **Insert Many Documents:**

```
{ acknowledged: true, insertedIds: { '0': 2, '1': 3, '2': 4 } }
```

- **Database:**

```
[
  { _id: 1, Name: 'Elanchezhian', Age: '21', Course: 'MCA' },
  { _id: 2, Name: 'Dhanush', Age: '22', Course: 'MSC CS' },
  { _id: 3, Name: 'Hariharan', Age: '20', Course: 'MSC DS' },
  { _id: 4, Name: 'Lachu', Age: '24', Course: 'MSc IT' }
]
```

- **Update Single Document:**

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

- **Update Multiple Document:**

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 4,
  modifiedCount: 4,
  upsertedCount: 0
}
```

➢ **Database:**

```
[
  {
    _id: 1,
    Name: 'Elanchezhian',
    Age: '21',
    Course: 'MCA',
    Batch: '2022'
  },
  {
    _id: 2,
    Name: 'Dhanush',
    Age: '22',
    Course: 'MSC CS',
    Batch: '2022'
  },
  {
    _id: 3,
    Name: 'Hariharan',
    Age: '20',
    Course: 'MSc IT',
    Batch: '2022'
  },
  { _id: 4, Name: 'Lachu', Age: '24', Course: 'MSc IT', Batch: '2022' }
]
```

➢ **Delete Single Documents:**

```
{ acknowledged: true, deletedCount: 1 }
```

➢ **Delete Many Documents:**

```
{ acknowledged: true, deletedCount: 3 }
```

Lab10: Implement Simple Queries for collections using MongoDB

Objective: To use MongoDB to perform simple queries on collections

1. Create a Collection

To create a new collection within the current database:

db.createCollection("mycollection")

Replace mycollection with your preferred collection name.

2. List Collections

To list all collections in the current database:

show collections

The detailed execution of these MongoDB commands is given below:

test> use BDA

BDA > db.stuinfo.insertMany([

{ "_id": 1, Name: 'Divya', Age: '22', Course: 'MSC CS' },

{ "_id": 2, Name: 'Shalini', Age: '20', Course: 'MSC DS' },

{ "_id": 3, Name: 'Hariharan', Age: '24', Course: 'MSc IT' }

])

BDA > db.stuinfo.find()

BDA > db.stuinfo.find({Name:'Shalini'})

BDA > db.stuinfo.find({$or:[{Name:'Divya'},{Age:'20'}]})

BDA >

db.stuinfo.find({Name:{$in:["Hariharan","Divya"]}})

BDA > db.stuinfo.find({ Age: { $gt:'21'}})

BDA > db.stuinfo.find({ Age: { $lt:'21'}})

BDA > db.stuinfo.find({ Name: {

$eq:'Shalini'}}) BDA > db.stuinfo.find({ Name:

{ $nin:['Shalini']}})

OUTPUT:

- **Finding Documents:**

```
[
  { _id: 1, Name: 'Divya', Age: '22', Course: 'MSC CS' },
  { _id: 2, Name: 'Shalini', Age: '20', Course: 'MSC DS' },
  { _id: 3, Name: 'Hariharan', Age: '24', Course: 'MSc IT' }
]
```

- **AND Operator:**

```
[ { _id: 2, Name: 'Shalini', Age: '20', Course: 'MSC DS' } ]
```

- **OR Operator:**

```
[
  { _id: 1, Name: 'Divya', Age: '22', Course: 'MSC CS' },
  { _id: 2, Name: 'Shalini', Age: '20', Course: 'MSC DS' }
]
```

- **IN Operator:**

```
[
  { _id: 1, Name: 'Divya', Age: '22', Course: 'MSC CS' },
  { _id: 3, Name: 'Hariharan', Age: '24', Course: 'MSc IT' }
]
```

- **Greater than Operator:**

```
[
  { _id: 1, Name: 'Divya', Age: '22', Course: 'MSC CS' },
  { _id: 3, Name: 'Hariharan', Age: '24', Course: 'MSc IT' }
]
```

- **Less than Operator:**

- **Equal to Operator:**

`

- **NIN Operator:**