Advanced Hadoop Exercises

Exercise 11: Word Length Count

Goal: Count the number of words of each length in a text file.

Step 1: Create input file

nano wordlength.txt

Paste the content:

Hadoop is a framework
for distributed storage
and processing of big data

Upload to HDFS:

hdfs dfs -mkdir -p /user/yourname/wordlength/input
hdfs dfs -put wordlength.txt /user/yourname/wordlength/input

Step 2: Run MapReduce Job

hadoop jar $EXAMPLES_JAR wordcount \
/user/yourname/wordlength/input \
/user/yourname/wordlength/output

Task for trainees: Write a **custom mapper script** to count words by length instead of by word.

Step 3: Check output

hdfs dfs -cat /user/yourname/wordlength/output/part-r-00000

Exercise 12: Sorting Numbers

Goal: Use MapReduce to sort numbers stored in a file.

Step 1: Create numbers file

nano numbers.txt

Paste:

25
3
42
7
19
12

Upload to HDFS:

```
hdfs dfs -mkdir -p /user/yourname/numbers/input
hdfs dfs -put numbers.txt /user/yourname/numbers/input
```

Step 2: Run Hadoop sorting example

```
hadoop jar $EXAMPLES_JAR sort \
/user/yourname/numbers/input \
/user/yourname/numbers/output
```

Step 3: Check sorted output

```
hdfs dfs -cat /user/yourname/numbers/output/part-r-00000
```

Exercise 13: Filtering Logs by Status Code

Goal: Extract all 404 HTTP errors from a log file using MapReduce.

Step 1: Create log file

```
nano logs.txt
```

Paste:

```
200 GET /index.html
404 GET /missing.html
500 GET /error
404 GET /page-not-found.html
200 POST /submit
```

Upload:

```
hdfs dfs -mkdir -p /user/yourname/filterlogs/input
hdfs dfs -put logs.txt /user/yourname/filterlogs/input
```

Step 2: Run MapReduce Job

Use grep in a streaming MapReduce example:

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-*.jar \
-input /user/yourname/filterlogs/input \
-output /user/yourname/filterlogs/output \
-mapper "grep 404" \
-reducer "cat"
```

Step 3: View output

```
hdfs dfs -cat /user/yourname/filterlogs/output/part-00000
```

Task for trainees: Modify the mapper to filter multiple status codes (e.g., 404 and 500).

Exercise 14: Count Lines per File in a Directory

Goal: Count the number of lines in each file in an HDFS directory.

Step 1: Create multiple small files

```
echo -e "line1\nline2\nline3" > file1.txt
echo -e "line1\nline2" > file2.txt
hdfs dfs -mkdir -p /user/yourname/multifiles/input
hdfs dfs -put file1.txt file2.txt /user/yourname/multifiles/input
```

Step 2: Run Hadoop MapReduce

Use the WordCount example with a **custom mapper** to count lines per file. For example, a mapper that outputs <filename, 1> for each line.

Step 3: Verify output

```
hdfs dfs -cat /user/yourname/multifiles/output/part-r-00000
```

Task for trainees: Implement the mapper in Python streaming.

Exercise 15: Compute Average Number from a File

Goal: Compute the average of numbers in a file using MapReduce.

Step 1: Create file

```
nano avgnumbers.txt
```

Paste:

```
10
20
```

30
40
50

Upload:

```
hdfs dfs -mkdir -p /user/yourname/avg/input
hdfs dfs -put avgnumbers.txt /user/yourname/avg/input
```

Step 2: Run Streaming MapReduce

Mapper (Python) example:

```
# mapper.py
import sys
for line in sys.stdin:
    print(f"sum\t{line.strip()}")
    print("count\t1")
```

Reducer (Python) example:

```
# reducer.py
import sys
total_sum = 0
count = 0
for line in sys.stdin:
    key, value = line.strip().split("\t")
    if key == "sum":
        total_sum += int(value)
    elif key == "count":
        count += int(value)
print(total_sum / count)
```

Run Hadoop streaming:

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-*.jar \
-input /user/yourname/avg/input \
-output /user/yourname/avg/output \
-mapper mapper.py \
-reducer reducer.py \
-file mapper.py \
-file reducer.py
```

Step 3: Check output

```
hdfs dfs -cat /user/yourname/avg/output/part-00000
```

Exercise 16: Merge Multiple Files in HDFS

Goal: Combine all small files in an HDFS directory into a single file.

Step 1: Upload multiple files

```
echo "apple" > f1.txt
echo "banana" > f2.txt
echo "cherry" > f3.txt
hdfs dfs -mkdir -p /user/yourname/merge/input
hdfs dfs -put f1.txt f2.txt f3.txt /user/yourname/merge/input
```

Step 2: Merge files

```
hdfs dfs -getmerge /user/yourname/merge/input merged.txt
cat merged.txt
```

Task for trainees: Use MapReduce to do the merge as well.

Exercise 17: Word Frequency Across Multiple Files

Goal: Count the total occurrences of words across multiple files in HDFS.

Step 1: Upload files

```
echo "hadoop spark hive" > fileA.txt
echo "hadoop hdfs spark" > fileB.txt
hdfs dfs -mkdir -p /user/yourname/multitext/input
hdfs dfs -put fileA.txt fileB.txt /user/yourname/multitext/input
```

Step 2: Run WordCount

```
hadoop jar $EXAMPLES_JAR wordcount \
/user/yourname/multitext/input \
/user/yourname/multitext/output
hdfs dfs -cat /user/yourname/multitext/output/part-r-00000
```