Hadoop - wordcount LAB - 6 (30/5/25)

start_all.sh

nano map.py →
```
#!/usr/bin/env python3
import sys
def mapper():
    for line in sys.stdin:
        line = line.strip()
        if not line:
            continue
        words = line.split()
        for word in words:
            print(f"{word}\t1")

if __name__ == "__main__":
    mapper()
```

nano red.py →
```
#!/usr/bin/env python3
c_word = None
c_count = 0
for line in sys.stdin:
    line = line.strip()
    if not line:
        continue
    word, count = line.split('\t', 1)
    try:
        count = int(count)
    except ValueError:
        continue
    if c_word == word:
        c_count += count
    else:
        if c_word:
            print(f"{c_word}\t{c_count}")
        c_word = word
        c_count = count

if c_word:
    print(f"{c_word}\t{c_count}")
```

(some commands or next top ten words)

Hadoop – Average Temperature

```
start-dfs.sh
start-yarn.sh
jps
hdfs --daemon start namenode
hdfs namenode -format
start-dfs.sh

hdfs dfs -mkdir /weatherdata
```

~~hdfs dfs~~

```
git clone https://github...
hdfs dfs -put hadoopbook/input/ncdc/all/* /weatherdata
```

```
nano mapper.py
chmod +x mapper.py

nano reducer.py
chmod +x reducer.py
```

mapper.py

```
#!/usr/bin/env python3
import sys
for line in sys.stdin:
    year = line[15:19]
    temp = line[87:92]
    if temp != "+9999":
        print(f"{year}\t{temp}")
```

```
hadoop jar jar-path \
input /weatherdata \
output /weather_output \
-mapper mapper.py
-reducer reducer.py
```

reducer.py

```
import sys
c_year = None
temp_sum = 0
count = 0
for line in sys.stdin:
    year, temp = line.strip().
                 split('\t')
    temp = int(temp)
    if c_year == year:
        temp_sum += temp
        count += 1
    else:
        if c_year:
            print(f"{c_year}\t
            {temp_sum/count}")
        c_year = year
        temp_sum = temp
        count = 1
if c_year:
    print(f"{c_year}\t{temp_sum/count}")
```

```
hdfs dfs -cat /weather_output/
part-0000

1901    46.70
1902    21.66
```

~~Hadoop : → Map reduce program~~ Hadoop - Mean Max Temp

(Same commands as previous program)

mapper.py :

```
#!/usr/bin/env python3
import sys
for line in sys.stdin:
    year = line[15:19]
    month = line[19:21]
    temp = line[87:92]
    if temp != "+9999":
        print(f"{year}-{month}\t{temp}")
```

reducer.py :

```
import sys
c_year = None
temp_sum = 0
count = 0

for line in sys.stdin:
    year_month, temp = line.strip().split('\t')
    temp = int(temp)
    if c_year == year_month:
        temp_sum += temp
        count += 1
    else:
        if c_year:
            print(f"{c_year}\t{temp_sum/count:.2f}")
        c_year = year_month
        temp_sum = temp
        count = 1

if c_year:
    print(f"{c_year}\t{temp_sum/count:.2f}")
```

Output :

```
1901-01    -25.88
1901-02    -91.72
1901-03     22.54
    :
```

Hadoop → Map-reduce program to list top 10 max words

```
start-all.sh
nano map.py
    #!/usr/bin/env python3
    import sys
    import re

    for line in sys.stdin:
        words = re.findall(r'\w+', line.lower())
        for word in words:
            print(f"{word}\t1")
```

```
nano red.py
    #!/usr/bin/env python3
    import sys
    from collections import defaultdict
    count_map = defaultdict(int)

    for line in sys.stdin:
        word, count = line.strip().split('\t')
        count_map[word] += int(count)

    sorted_words = sorted(count_map.items(), key=lambda
                          x: (-x[1], x[0]))

    for word, count in sorted_words[:10]:
        print(f"{word}\t{count}")
```

```
chmod +x map.py
chmod +x red.py
hdfs dfs -mkdir /bda11
hdfs dfs -ls /
(make a input file (inp.txt))
hdfs dfs -put inp.txt /bda11
```

```
hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/
hadoop-streaming-3.3.6.jar -input /bdall/inp.txt
-output /bdall/output -mapper /home/hadoop/map.py
-reducer /home/hadoop/red.py -file /home/hadoop/map.py
-file /home/hadoop/red.py

hadoop fs -ls /bdall
hadoop fs -rm /bdall/output
hadoop fs -cat /bdall/output/part-0000
```

Scala : → To print numbers from 1 to 100 using for loop

In ubuntu user.

```
$ sudo apt install scala
$ nano prints.scala
```

```
object prints {
    def main(args: Array[String]): Unit = {
        for (i <- 1 to 100) {
            print(i)
        }
    }
}
```

ctrl+o, enter, ctrl+x

```
$ scalac prints.scala
$ scala prints
1 2 3 4 ... 100
```

Spark : count words > 4

sudo apt update

wget https://archive.apache.org/dist/spark/spark-3.5.1/spark-3.5.1-bin-hadoop3.tgz

tar -xvzf spark-3.5.1-bin-hadoop3.tgz

ls /opt | grep spark

cd /opt

ls /opt | grep spark

nano ~/.bashrc → add these

export SPARK_HOME=/opt/spark

export PATH=$SPARK_HOME/bin:$PATH

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64

source ~/.bashrc

spark-shell

New window

mkdir ~/sparks

nano wordcount.py

```
from pyspark import SparkContext
sc = SparkContext("local", "wordCountApp")
text_rdd = sc.textFile("input.txt")
words_rdd = text_rdd.flatMap(lambda line: line.split())
pairs_rdd = words_rdd.map(lambda word: (word, 1))
words_count_rdd = pairs_rdd.reduceByKey(lambda a, b: a+b)
filt_rdd = words_count_rdd.filter(lambda x: x[1] > 4)
for word, count in filt_rdd.collect():
    print(f"{word}: {count}")
filt_rdd.saveAsTextFile("output_directory")
```

nano input.txt (your text)

spark-submit wordcount.py input.txt

Output:

the: 6

lorem: 6

ipsum: 7

Open-Ended Question : Text cleaning & Print cleaned text (Spark)

mapper.py:

~~#!/usr/bin/env python~~

```
import
{
import org.apache.spark.sql.functions._
import org.apache.spark.sql.types._

val stopwords = Set("the", "is", "in", "and", "to", "a", "of", "on", "that")

def lemmatize(word: String): String = word match {
  case w if w.endsWith("ing") => w.stripSuffix("ing")
  case w if w.endsWith("ed") => w.stripSuffix("s")
  case _ => word
}

val cleanText = udf((line: String) => {
  line.toLowerCase
  .replaceAll("[^a-zA-Z\\s]", "")
  .split("\\s+")
  .filter(w => w.nonEmpty && !stopwords.contains(w))
  .map(lemmatize)
  .mkString(" ")
})

val lines = spark.readStream
  .format("socket")
  .option("host", "localhost")
  .option("port", 9999)
  .load()

val cleaned =
  lines.withColumn("cleaned",
  cleanText(col("value")))

val query = cleaned.writeStream
```

.outputMode("append")
.format("console")
.start()

query.awaitTermination()

}

22/5/25