**JAVA**

**$ javac --version**

**$ sudo apt install default-jdk**

**$ sudo apt install default-jre**

**$ javac --version**

**$ cd <1MS23SCS/SCN\*\*>**

**Hadoop**

**$ tar xvzf hadoop-3.4.0.tar.gz**

**$ cd hadoop-3.4.0/**

**$ gedit bash.sh**

**export JAVA\_HOME=$(readlink -f $(which javac) | awk 'BEGIN {FS="/bin"} {print $1}')**

**export PATH=$(echo $PATH):$(pwd)/bin**

**export CLASSPATH=$(hadoop classpath)**

**$ source bash.sh**

**$ hadoop**

**SPARK:**

**$ hadoop**

**$ tar xvzf spark-3.3.0-bin-hadoop3.tar.gz**

**$ cd spark-3.3.0-bin-hadoop3/**

**$ gedit bash.sh**

**export PATH=$(echo $PATH):$(pwd)/bin**

**$ source bash.sh**

**$ spark-shell –version**

**Write a spark program using Python, to analyze the given Weather Report Data and to generate a report with cities having maximum and minimum temperature for a particular year.**

**mkdir weather**

**cd weather**

**gedit weather.py**

**gedit input.txt**

import sys

if len(sys.argv) != 4:

print("Provide Input File and Output Directory")

sys.exit(0)

from pyspark import SparkContext

sc = SparkContext()

f = sc.textFile(sys.argv[1])

temp = f.map(lambda x: (int(x[15:19]), int(x[87:92])))

mini = temp.reduceByKey(lambda a, b: a if a < b else b)

mini.saveAsTextFile(sys.argv[2])

maxi = temp.reduceByKey(lambda a, b: a if a > b else b)

maxi.saveAsTextFile(sys.argv[3])

0067011990999991950051507004+68750+023550FM-12+038299999V0203301N00671220001CN9999999N9+00001+99999999999

0043011990999991950051512004+68750+023550FM-12+038299999V0203201N00671220001CN9999999N9+00221+99999999999

0043011990999991950051518004+68750+023550FM-12+038299999V0203201N00261220001CN9999999N9-00111+99999999999

0043012650999991949032412004+62300+010750FM-12+048599999V0202701N00461220001CN0500001N9+01111+99999999999

0043012650999991949032418004+62300+010750FM-12+048599999V0202701N00461220001CN0500001N9+00781+99999999999

spark-submit weather.py input.txt minimum maximum

$ cat minimum/\*

$ cat maximum/\*

**Write a spark program using Python, to analyze the given Earthquake Data and generate statistics with region and magnitude/ region and depth/ region and latitude/ region and longitude**

**mkdir earthquake**

**cd earthquake**

**gedit earthquake.py**

import sys

from pyspark import SparkContext

if len(sys.argv) != 6:

print("Provide Input File and Output Directory")

sys.exit(0)

sc = SparkContext()

f = sc.textFile(sys.argv[1])

temp = f.map(lambda x: (x.split(',')[11], float(x.split(',')[8])))

maxi = temp.reduceByKey(lambda a, b: a if a > b else b)

maxi.saveAsTextFile(sys.argv[2])

temp = f.map(lambda x: (x.split(',')[11], float(x.split(',')[9])))

maxi = temp.reduceByKey(lambda a, b: a if a > b else b)

maxi.saveAsTextFile(sys.argv[3])

temp = f.map(lambda x: (x.split(',')[11], float(x.split(',')[6])))

maxi = temp.reduceByKey(lambda a, b: a if a > b else b)

maxi.saveAsTextFile(sys.argv[4])

temp = f.map(lambda x: (x.split(',')[11], float(x.split(',')[7])))

maxi = temp.reduceByKey(lambda a, b: a if a > b else b)

maxi.saveAsTextFile(sys.argv[5])

spark-submit earthquake.py earthquake-input.csv magnitude depth latitude longitude

$ cat magnitude/\*

$ cat depth/\*

$ cat latitude/\*

$ cat longitude/\*

**Write a spark program using Python, to analyze the given Insurance Data and generate a statistics report with the construction building name and the count of building/ county name and its frequency**

**mkdir insurance**

**cd insurance**

**gedit insurance.py**

import sys

from pyspark import SparkContext

if len(sys.argv) != 4:

print("Provide Input File and Output Directory")

sys.exit(0)

sc = SparkContext()

f = sc.textFile(sys.argv[1])

temp = f.map(lambda x: (x.split(',')[16], 1))

data = temp.countByKey()

dd = sc.parallelize(data.items())

dd.saveAsTextFile(sys.argv[2])

temp = f.map(lambda x: (x.split(',')[2], 1))

data = temp.countByKey()

dd = sc.parallelize(data.items())

dd.saveAsTextFile(sys.argv[3])

spark-submit insurance.py input-insurance.csv construction county

$ cat construction/\*

$ cat county/\*

**Write a spark program using Python, to analyze the given Sales Records over a period of time and generate data about the country’s total sales, and the total number of the products. / Country’s total sales and the frequency of the payment mode.**

**mkdir sales**

**cd sales**

**gedit sales.py**

import sys

from pyspark import SparkContext

if len(sys.argv) != 4:

print("Provide Input File and Output Directory")

sys.exit(0)

sc = SparkContext()

f = sc.textFile(sys.argv[1])

temp = f.map(lambda x: (x.split(',')[7], 1))

data = temp.countByKey()

dd = sc.parallelize(data.items())

dd.saveAsTextFile(sys.argv[2])

temp = f.map(lambda x: (x.split(',')[3], 1))

data = temp.countByKey()

dd = sc.parallelize(data.items())

dd.saveAsTextFile(sys.argv[3])

spark-submit sales.py input-sales.csv products frequency

$ cat products/\*

$ cat frequency/\*