

SPARK – INSTALLATION

The following steps show how to install Apache Spark.

Step 1: Verifying Java Installation

If Java is already installed on your system, you get to see the following response or some other versions.

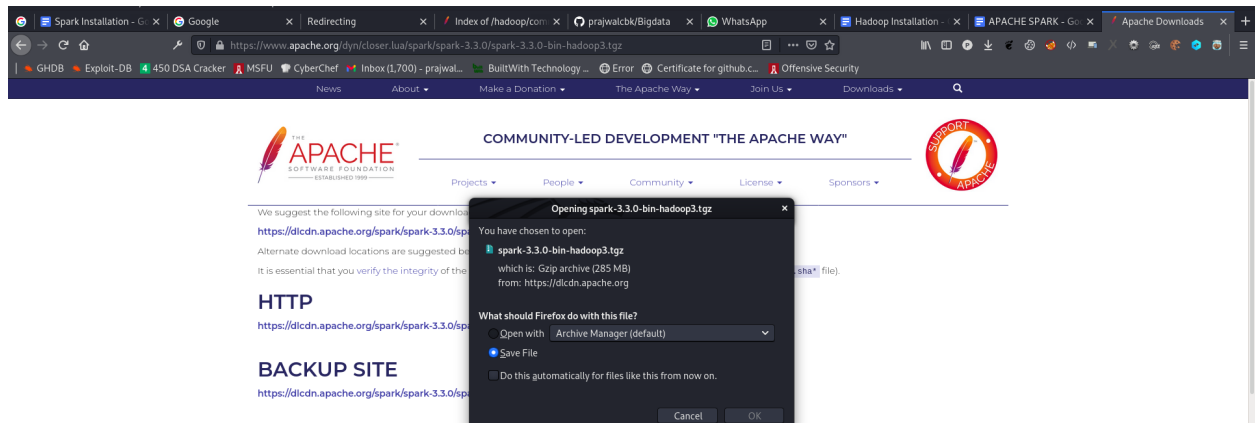
```
root@kali:~# java --version
openjdk 11.0.11-ea 2021-04-20
OpenJDK Runtime Environment (build 11.0.11-ea+4-post-Debian-1)
OpenJDK 64-Bit Server VM (build 11.0.11-ea+4-post-Debian-1, mixed mode, sharing)
```

Step 2: Downloading Apache Spark

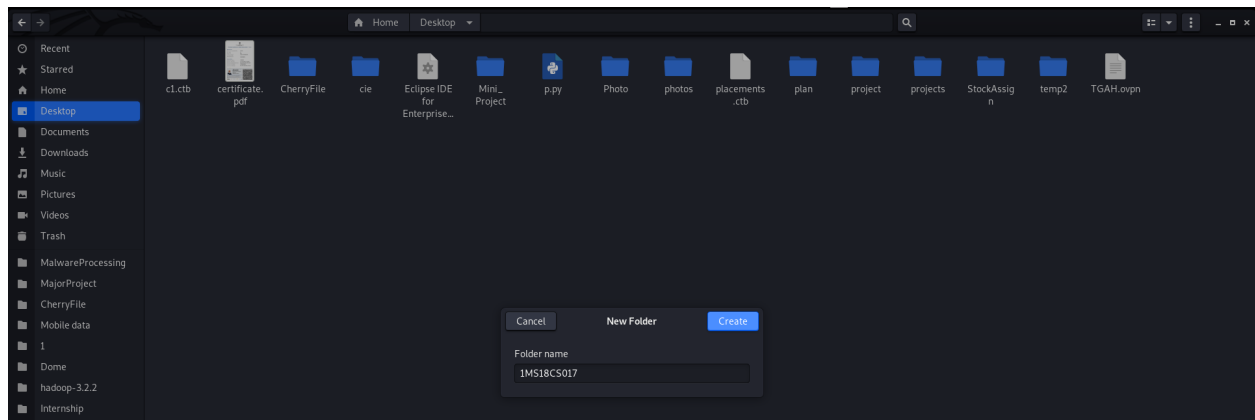
Download the latest version of Spark by visiting the following link [Download Spark](https://spark.apache.org/downloads.html)

<https://spark.apache.org/downloads.html> . Select the latest version in Spark release and select pre-built for Apache Hadoop 3.3 and later . Click on the Download Spark link . It will navigate to one more page , and use HTTP to download the file . After downloading it, you will find the Spark tar file in the download folder.

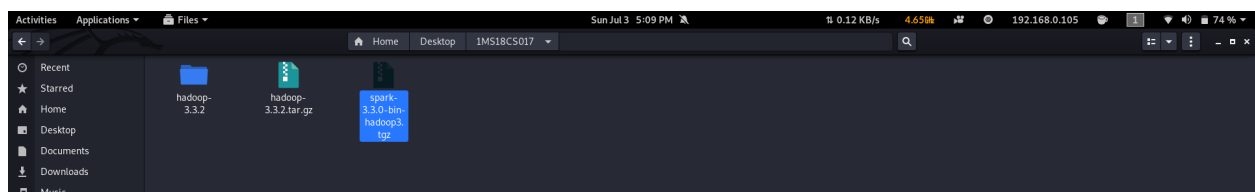
The image shows two screenshots related to downloading Apache Spark. The top screenshot is a screenshot of the Apache Spark download page. It features a navigation bar with links like 'Download', 'Libraries', 'Documentation', 'Examples', 'Community', and 'Developers'. The main content area is titled 'Download Apache Spark™' and contains a list of steps: 1. Choose a Spark release (3.3.0 (Jun 16 2022)), 2. Choose a package type (Pre-built for Apache Hadoop 3.3 and later), 3. Download Spark (spark-3.3.0-bin-hadoop3.tgz), and 4. Verify this release using the 3.3.0 signatures, checksums and project release KEYS by following these procedures. There is also a 'Latest News' sidebar on the right. The bottom screenshot is a browser view of the same page, showing the Apache Software Foundation logo, the 'COMMUNITY-LED DEVELOPMENT "THE APACHE WAY"' banner, and the download instructions. It suggests the download site <https://d1cdn.apache.org/spark/spark-3.3.0/spark-3.3.0-bin-hadoop3.tgz> and provides alternate download locations and verification instructions using PCP signature or hash.



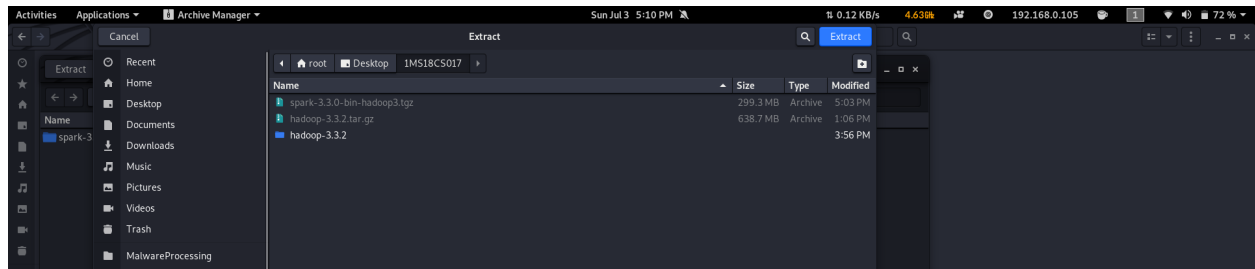
Step 3: Create a new Folder inside Desktop , name the Folder as your USN <1ms18cs017>.



Step 4 . Move the Downloaded Spark File to USN <1ms18cs017> Folder.



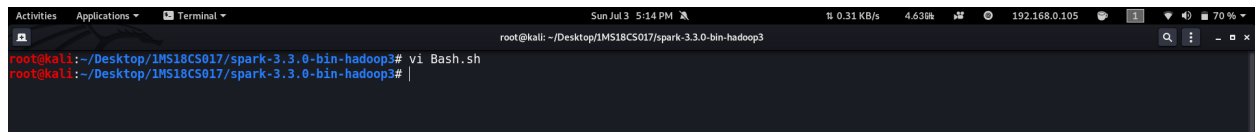
Step 5. Right Click on that File and Extract inside the USN <1ms18cs017> Folder.



Step 6:. Open Terminal

Navigate to Extracted Hadoop Folder `cd ~/Desktop/<1ms18cs017>/spark-3.3.0-bin-hadoop3`

7. Create a New File named Bash.sh



8. Copy the Below code and Paste inside Bash.sh and save that File.

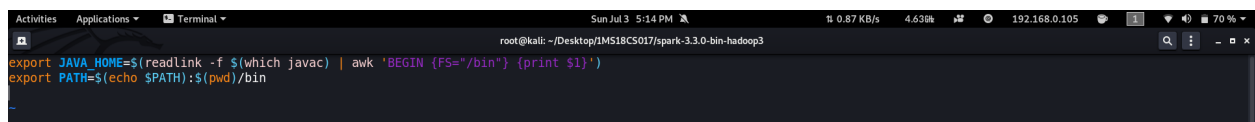
```
export JAVA_HOME=$(readlink -f $(which javac) | awk 'BEGIN {FS="/bin"} {print $1}')
```

```
if ! command -v spark-shell --version &> /dev/null
```

```
then
```

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

```
fi
```



9. Execute the bash.sh File using following command `source Bash.sh`.

NOTE: Make source before compiling or running spark compile this file.

10. Verify JAVA_HOME variable to be set to Java Path and PATH variable has your USN Spark Folder.If any previous PATH set to Spark Folder remove that inside .bashrc file.

```
Activities Applications Terminal Sun Jul 3 5:22 PM 1.05 KB/s 4.63% 192.168.0.105 root@kali: ~/Desktop/IMS18CS017/spark-3.3.0-bin-hadoop3 root@kali:~/Desktop/IMS18CS017/spark-3.3.0-bin-hadoop3# echo $PATH /root/.nmv/versions/node/v14.17.0/bin:/root/anaconda3/condabin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin:/root/Desktop/IMS18CS017/spark-3.3.0-bin-hadoop3/bin root@kali:~/Desktop/IMS18CS017/spark-3.3.0-bin-hadoop3# echo $JAVA_HOME /usr/lib/jvm/java-11-openjdk-amd64 root@kali:~/Desktop/IMS18CS017/spark-3.3.0-bin-hadoop3# |
```

11. Verify Hadoop is Installed or not by executing **spark-shell --version** command.if command gives Information about Hadoop command then Hadoop is Successfully Installed.

```
scala> root@kali:~/Desktop/IMS18CS017/spark-3.3.0-bin-hadoop3# spark-shell --version 22/07/03 17:18:01 WARN Utils: Your hostname, kali resolves to a loopback address: 127.0.1.1; using 192.168.0.105 instead (on interface wlan0) 22/07/03 17:18:01 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address WARNING: An illegal reflective access operation has occurred WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/usr/local/lib/python3.9/dist-packages/pyspark/jars/spark-unsafe_2.12-3.2.0.jar) to constructor java.nio.DirectByteBuffer(long,int) WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations WARNING: All illegal access operations will be denied in a future release Welcome to version 3.2.0 Using Scala version 2.12.15, OpenJDK 64-Bit Server VM, 11.0.11-ea Branch HEAD Compiled by user ubuntu on 2021-10-06T12:46:30Z Revision 5d45a415f3a29898d92380380cfd82bfc7f579ea Url https://github.com/apache/spark Type --help for more information.
```

Execute all spark python files with **spark-submit<python_filename>.py <inputFile> <outputfolder>**

Write a spark to analyze the given weather report data and to generate a report with cities having maximum temperature for a particular year

```
Activities Applications Terminal Sun Jul 3 7:57 PM 0.36 KB/s 4.63% 192.168.0.105 96% root@kali: ~/Desktop/IMS18CS017/spark-3.3.0-bin-hadoop3/programs/data/weather root@kali:~/Desktop/IMS18CS017/spark-3.3.0-bin-hadoop3# source Bash.sh root@kali:~/Desktop/IMS18CS017/spark-3.3.0-bin-hadoop3# echo $PATH /root/.nmv/versions/node/v14.17.0/bin:/root/anaconda3/condabin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin:/root/Desktop/IMS18CS017/spark-3.3.0-bin-hadoop3/bin root@kali:~/Desktop/IMS18CS017/spark-3.3.0-bin-hadoop3# spark-shell --version 22/07/03 19:55:51 WARN Utils: Your hostname, kali resolves to a loopback address: 127.0.1.1; using 192.168.0.105 instead (on interface wlan0) 22/07/03 19:55:51 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address WARNING: An illegal reflective access operation has occurred WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/usr/local/lib/python3.9/dist-packages/pyspark/jars/spark-unsafe_2.12-3.2.0.jar) to constructor java.nio.DirectByteBuffer(long,int) WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations WARNING: All illegal access operations will be denied in a future release Welcome to version 3.2.0 Using Scala version 2.12.15, OpenJDK 64-Bit Server VM, 11.0.11-ea Branch HEAD Compiled by user ubuntu on 2021-10-06T12:46:30Z Revision 5d45a415f3a29898d92380380cfd82bfc7f579ea Url https://github.com/apache/spark Type --help for more information. root@kali:~/Desktop/IMS18CS017/spark-3.3.0-bin-hadoop3# cd programs/data/weather/ root@kali:~/Desktop/IMS18CS017/spark-3.3.0-bin-hadoop3/programs/data/weather# cat 1.py import sys if(len(sys.argv)!=3): print("Provide Input File and Output Directory") sys.exit(0) from pyspark import SparkContext sc = SparkContext() f = sc.textFile(sys.argv[1]) tempf.map(lambda x: (int(x[15:19]),int(x[87:92]))) max1=temp.reduceByKey(lambda a,b:a if a>b else b) max1.saveAsTextFile(sys.argv[2]) root@kali:~/Desktop/IMS18CS017/spark-3.3.0-bin-hadoop3/programs/data/weather# cat 1.py 2.py out/ out2/ tem.txt root@kali:~/Desktop/IMS18CS017/spark-3.3.0-bin-hadoop3/programs/data/weather# cat tem.txt 0067011909999991950051507004+68750+023550FM-12+038299999V0203301N00671220001CN99999999N+00001+99999999999 0043011909999991950051512004+68750+023550FM-12+038299999V0203201N00671220001CN99999999N+00221+99999999999 0043011909999991950051518004+68750+023550FM-12+038299999V0203201N00671220001CN99999999N+00111+99999999999 0043012650999991949032412004+62300+010750FM-12+048599999V0202701N00461220001CN0500001N+00111+99999999999 0043012650999991949032412004+62300+010750FM-12+048599999V0202701N00461220001CN0500001N+00781+99999999999 root@kali:~/Desktop/IMS18CS017/spark-3.3.0-bin-hadoop3/programs/data/weather# spark-submit 1.py tem.txt output 22/07/03 19:56:55 WARN Utils: Your hostname, kali resolves to a loopback address: 127.0.1.1; using 192.168.0.105 instead (on interface wlan0) 22/07/03 19:56:55 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address WARNING: An illegal reflective access operation has occurred WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/usr/local/lib/python3.9/dist-packages/pyspark/jars/spark-unsafe_2.12-3.2.0.jar) to constructor java.nio.DirectByteBuffer(long,int) WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations WARNING: All illegal access operations will be denied in a future release
```

```
import sys
if(len(sys.argv)!=3):
    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])))
maxi=temp.reduceByKey(lambda a,b:a if a>b else b)
maxi.saveAsTextFile(sys.argv[2])
```

```
root@kali: ~/Desktop/IMS18CS017/spark-3.3.0-bin-hadoop3/programs/data/weather
22/07/03 19:56:58 INFO BlockManagerInfo: Added broadcast 2 piece0 in memory on 192.168.0.105:36767 (size: 40.4 KiB, free: 434.3 MiB)
22/07/03 19:56:58 INFO SparkContext: Created broadcast 2 from broadcast at DAGScheduler.scala:1427
22/07/03 19:56:58 INFO DAGScheduler: Submitting 2 missing tasks from ResultStage 1 (MapPartitionsRDD[8] at saveAsTextFile at NativeMethodAccessorImpl.java:0) (first 15 tasks are for partitions Vector(0, 1))
22/07/03 19:56:58 INFO TaskSchedulerImpl: Adding task set 1.0 with 2 tasks resource profile 0
22/07/03 19:56:58 INFO TaskSetManager: Starting task 0.0 in stage 1.0 (TID 2) (192.168.0.105, executor driver, partition 0, NODE_LOCAL, 4271 bytes) taskResourceAssignments Map()
22/07/03 19:56:58 INFO TaskSetManager: Starting task 1.0 in stage 1.0 (TID 3) (192.168.0.105, executor driver, partition 1, NODE_LOCAL, 4271 bytes) taskResourceAssignments Map()
22/07/03 19:56:58 INFO Executor: Running task 1.0 in stage 1.0 (TID 3)
22/07/03 19:56:58 INFO Executor: Running task 0.0 in stage 1.0 (TID 2)
22/07/03 19:56:58 INFO ShuffleBlockFetcherIterator: Getting 1 (72.0 B) non-empty blocks including 1 (72.0 B) local and 0 (0.0 B) host-local and 0 (0.0 B) push-merged-local and 0 (0.0 B) remote blocks
22/07/03 19:56:58 INFO ShuffleBlockFetcherIterator: Getting 1 (72.0 B) non-empty blocks including 1 (72.0 B) local and 0 (0.0 B) host-local and 0 (0.0 B) push-merged-local and 0 (0.0 B) remote blocks
22/07/03 19:56:58 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 8 ms
22/07/03 19:56:58 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 8 ms
22/07/03 19:56:58 INFO HadoopMapRedCommitProtocol: Using output committer class org.apache.hadoop.mapred.FileOutputCommitter
22/07/03 19:56:58 INFO FileOutputCommitter: File Output Committer Algorithm version is 1
22/07/03 19:56:58 INFO FileOutputCommitter: FileOutputCommitter skip cleanup temporary folders under output directory:false, ignore cleanup failures: false
22/07/03 19:56:58 INFO HadoopMapRedCommitProtocol: Using output committer class org.apache.hadoop.mapred.FileOutputCommitter
22/07/03 19:56:58 INFO FileOutputCommitter: File Output Committer Algorithm version is 1
22/07/03 19:56:58 INFO FileOutputCommitter: FileOutputCommitter skip cleanup temporary folders under output directory:false, ignore cleanup failures: false
22/07/03 19:56:58 INFO PythonRunner: Times: total = 10, boot = -319, init = 328, finish = 0
22/07/03 19:56:58 INFO PythonRunner: Times: total = 10, boot = -319, init = 328, finish = 0
22/07/03 19:56:58 INFO FileOutputCommitter: Saved output of task 'attempt-202207031956579019894571941898978_0008_m_000001_0' to file:/root/Desktop/IMS18CS017/spark-3.3.0-bin-hadoop3/programs/data/weather/output/
temporary/0/task_202207031956579019894571941898978_0008_m_000001_0
22/07/03 19:56:58 INFO SparkHadoopMapRedUtil: attempt-202207031956579019894571941898978_0008_m_000001_0: Committed
22/07/03 19:56:58 INFO FileOutputCommitter: Saved output of task 'attempt-202207031956579019894571941898978_0008_m_000000_0' to file:/root/Desktop/IMS18CS017/spark-3.3.0-bin-hadoop3/programs/data/weather/output/
temporary/0/task_202207031956579019894571941898978_0008_m_000000_0
22/07/03 19:56:58 INFO Executor: Finished task 1.0 in stage 1.0 (TID 3). 1909 bytes result sent to driver
22/07/03 19:56:58 INFO SparkHadoopMapRedUtil: attempt-202207031956579019894571941898978_0008_m_000000_0: Committed
22/07/03 19:56:58 INFO Executor: Finished task 0.0 in stage 1.0 (TID 2). 1909 bytes result sent to driver
22/07/03 19:56:58 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 2) in 112 ms on 192.168.0.105 (executor driver) (1/2)
22/07/03 19:56:58 INFO TaskSetManager: Finished task 1.0 in stage 1.0 (TID 3) in 111 ms on 192.168.0.105 (executor driver) (2/2)
22/07/03 19:56:58 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
22/07/03 19:56:58 INFO DAGScheduler: ResultsStage 1 (runJob at SparkHadoopWriter.scala:83) finished in 0.157 s
22/07/03 19:56:58 INFO DAGScheduler: Job 0 is finished. Cancelling potential speculative or zombie tasks for this job
22/07/03 19:56:58 INFO TaskSchedulerImpl: Killing all running tasks in stage 1: Stage finished
22/07/03 19:56:58 INFO DAGScheduler: Job 0 finished: runJob at SparkHadoopWriter.scala:83, took 0.901546 s
22/07/03 19:56:58 INFO SparkHadoopWriter: Start to commit write Job job_202207031956579019894571941898978_0008
22/07/03 19:56:58 INFO SparkHadoopWriter: Write Job job_202207031956579019894571941898978_0008 committed. Elapsed time: 9 ms.
22/07/03 19:56:58 INFO SparkContext: Invoking stop() from shutdown hook
22/07/03 19:56:58 INFO SparkUI: Stopped Spark web UI at http://192.168.0.105:4040
22/07/03 19:56:58 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
22/07/03 19:56:58 INFO MemoryStore: MemoryStore cleared
22/07/03 19:56:58 INFO BlockManager: BlockManager stopped
22/07/03 19:56:58 INFO BlockManagerMaster: BlockManagerMaster stopped
22/07/03 19:56:58 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
22/07/03 19:56:58 INFO SparkContext: Successfully stopped SparkContext
22/07/03 19:56:58 INFO ShutdownHookManager: Shutdown hook called
22/07/03 19:56:58 INFO ShutdownHookManager: Deleting directory /tmp/spark-e585eaf8-a5e6-41d4-82ea-9dc061326242
22/07/03 19:56:58 INFO ShutdownHookManager: Deleting directory /tmp/spark-e585eaf8-a5e6-41d4-82ea-9dc061326242/pyspark-5906c883-5db4-45ec-8642-9795e2dda63d
22/07/03 19:56:58 INFO ShutdownHookManager: Deleting directory /tmp/spark-42b834fe-eb18-4908-9e86-61848ea7abc9
root@kali:~/Desktop/IMS18CS017/spark-3.3.0-bin-hadoop3/programs/data/weather# cat output/*
(1950, 22)
(1949, 11)
root@kali:~/Desktop/IMS18CS017/spark-3.3.0-bin-hadoop3/programs/data/weather#
```

Write a spark to analyze the given weather report data and to generate a report with cities having minimum temperature for a particular year

```
import sys
if(len(sys.argv)!=3):
    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])
```

Write a spark program to analyze the given Earthquake data and generate statistics with region and magnitude

```
import sys
if(len(sys.argv)!=3):
    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: (x.split(',')[11],float(x.split(',')[8])))
maxi=temp.reduceByKey(lambda a,b:a if a>b else b)
maxi.saveAsTextFile(sys.argv[2])
```

Write a spark program to analyze the given Earthquake data and generate statistics with region and depth

```
import sys
if(len(sys.argv)!=3):
    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: (x.split(',')[11],float(x.split(',')[9])))
maxi=temp.reduceByKey(lambda a,b:a if a>b else b)
maxi.saveAsTextFile(sys.argv[2])
```

Write a spark program to analyze the given Earthquake data and generate statistics with region and latitude

```
import sys
if(len(sys.argv)!=3):
    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: (x.split(',')[11],float(x.split(',')[6])))
maxi=temp.reduceByKey(lambda a,b:a if a>b else b)
maxi.saveAsTextFile(sys.argv[2])
```

Write a spark program to analyze the given Earthquake data and generate statistics with region and longitude

```
import sys
if(len(sys.argv)!=3):
    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: (x.split(',')[11],float(x.split(',')[7])))
maxi=temp.reduceByKey(lambda a,b:a if a>b else b)
maxi.saveAsTextFile(sys.argv[2])
```

Write a spark program to analyze the given Insurance data and generate a statistics report with the construction building name and the count of building.

```
import sys
if(len(sys.argv)!=3):
    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: (x.split(',')[16],1))
data=temp.countByKey()
dd=sc.parallelize(data.items())
dd.saveAsTextFile(sys.argv[2])
```

Write a spark program to analyze the given Insurance data and generate a statistics report with the county name and its frequency.

```
import sys
if(len(sys.argv)!=3):
    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: (x.split(',')[2],1))
data=temp.countByKey()
dd=sc.parallelize(data.items())
dd.saveAsTextFile(sys.argv[2])
```

Write a map-reduce program to analyze the given employee record data and generate a statistics report with the total Sales for female and male employees

```
import sys
if(len(sys.argv)!=3):
    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: (x.split("\t")[3],float(x.split("\t")[8])))
total=temp.reduceByKey(lambda a,b : a+b)
total.saveAsTextFile(sys.argv[2])
```

Write a map-reduce program 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

```
import sys
if(len(sys.argv)!=3):
    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: (x.split(',')[7],1))
data=temp.countByKey()
dd=sc.parallelize(data.items())
dd.saveAsTextFile(sys.argv[2])
```

Write a map-reduce program to analyze the given sales records over a period of time and generate data about the country's total sales and the frequency of the payment mode.

```
import sys
if(len(sys.argv)!=3):
    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: (x.split(',')[3],1))
```



```
data=temp.countByKey()  
dd=sc.parallelize(data.items())  
dd.saveAsTextFile(sys.argv[2])
```