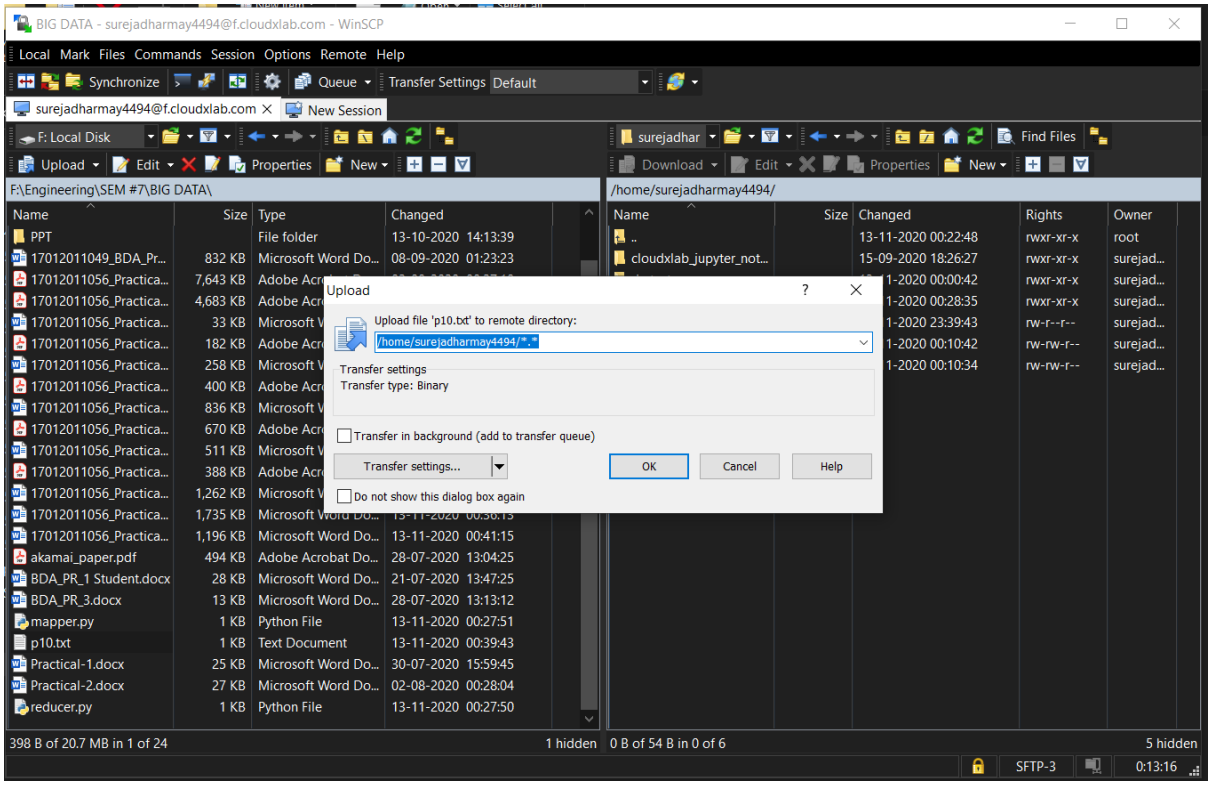
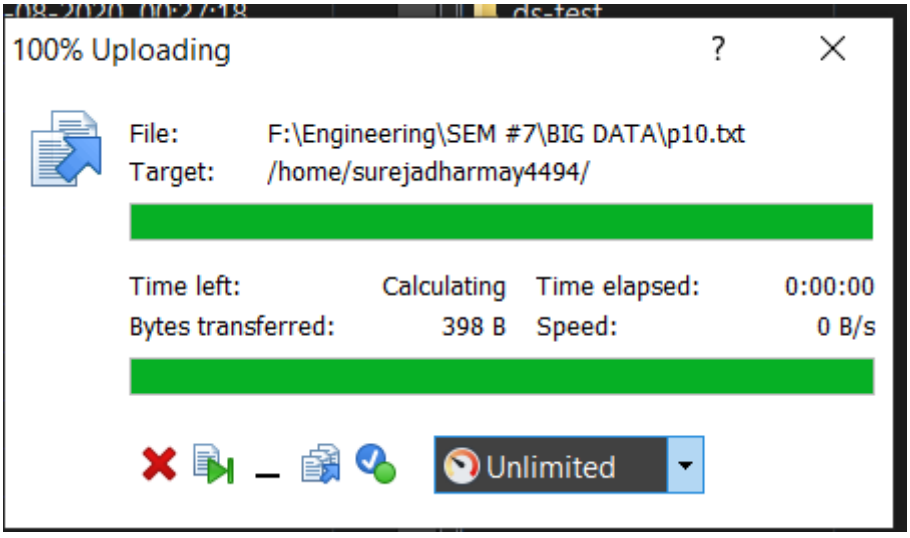
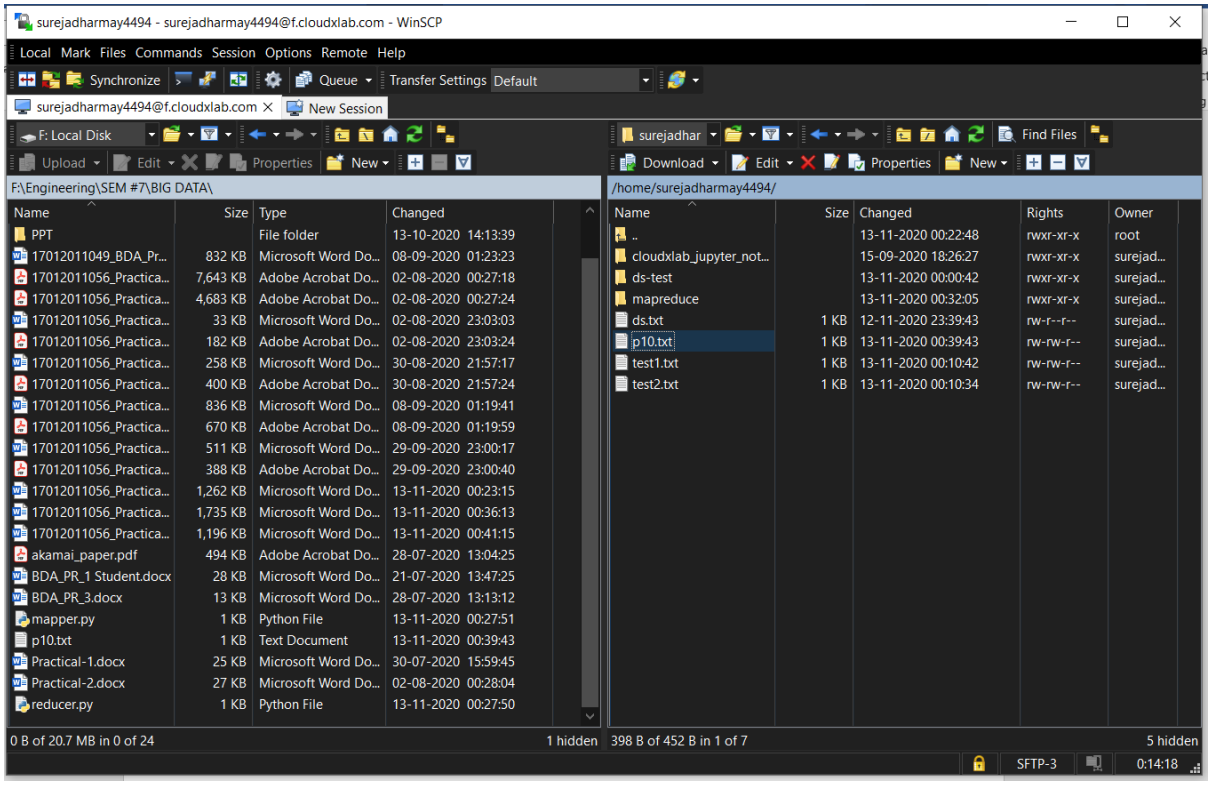
Practical-10

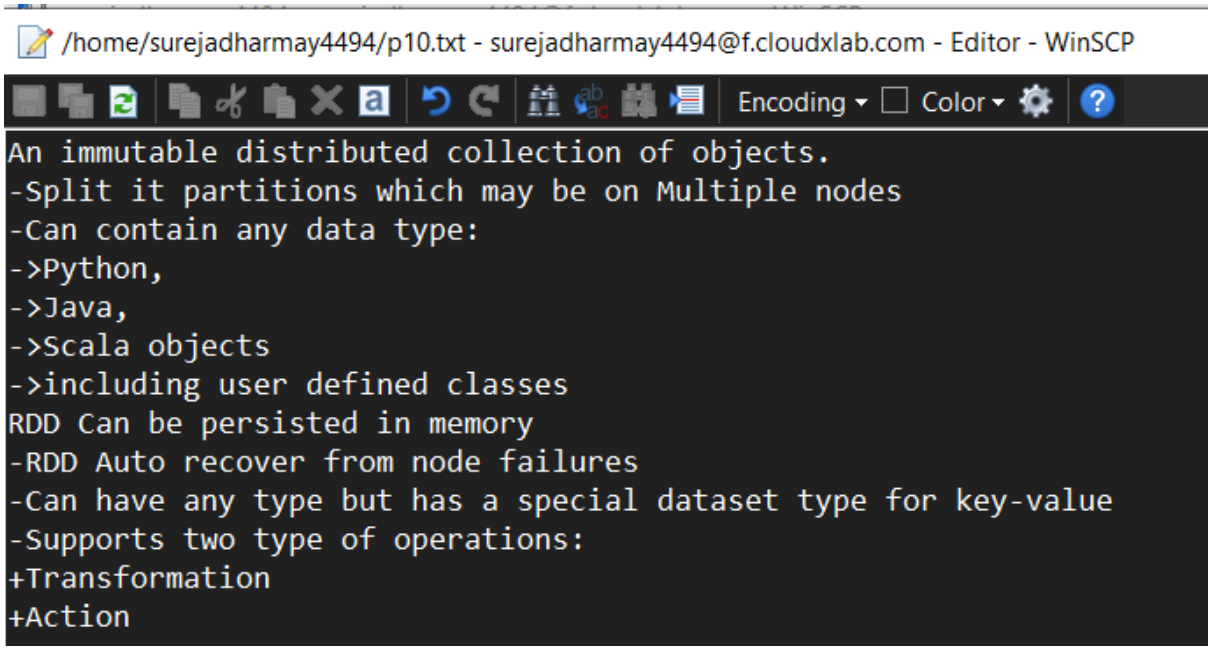
TASK : Create and upload one text file with some data in it. Remember the number of lines in it must be a minimum of 10. Now read this file using the Scala and Python interface in cloudxlab and exit both the interface.







P10-sample.txt



Scala :

```
[surejadharmay4494@cxln5 mapreduce]$ scala
Welcome to Scala 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_212).
Type in expressions for evaluation. Or try :help.

scala>
```

```
scala> val source = scala.io.Source.fromFile("p10.txt").mkString
source: String =
"An immutable distributed collection of objects.
-Split it partitions which may be on Multiple nodes
-Can contain any data type:
->Python,
->Java,
->Scala objects
->including user defined classes
RDD Can be persisted in memory
-RDD Auto recover from node failures
-Can have any type but has a special dataset type for key-value
-Supports two type of operations:
+Transformation
+Action
"
```

Python :

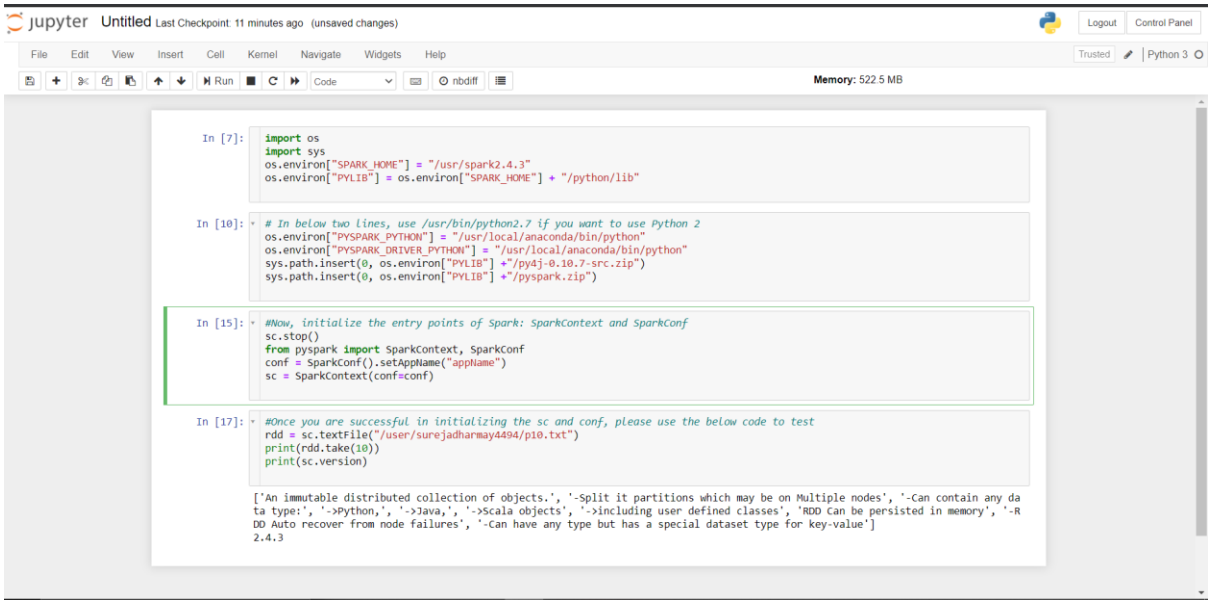
```
[surejadharmay4494@cxln5 ~]$ python
Python 2.7.5 (default, Apr  9 2019, 14:30:50)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-36)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

```
[surejadharmay4494@cxln5 ~]$ python
Python 2.7.5 (default, Apr  9 2019, 14:30:50)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-36)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> f=open('p10.txt','r')
>>> print(f.read())
An immutable distributed collection of objects.
-Split it partitions which may be on Multiple nodes
-Can contain any data type:
->Python,
->Java,
->Scala objects
->including user defined classes
RDD Can be persisted in memory
-RDD Auto recover from node failures
-Can have any type but has a special dataset type for key-value
-Supports two type of operations:
+Transformation
+Action

>>> exit()
[surejadharmay4494@cxln5 ~]$
```

TASK : Open Jupyter Notebook in CloudXLab and initialize Python interface in it and Perform Apache Spark reading file and check the version of the spark.

Code :

```
import os

import sys

os.environ["SPARK_HOME"] = "/usr/spark2.4.3"

os.environ["PYLIB"] = os.environ["SPARK_HOME"] + "/python/lib"

# In below two lines, use /usr/bin/python2.7 if you want to
use Python 2

os.environ["PYSPARK_PYTHON"] =
"/usr/local/anaconda/bin/python"

os.environ["PYSPARK_DRIVER_PYTHON"] =
"/usr/local/anaconda/bin/python"

sys.path.insert(0, os.environ["PYLIB"] +"/py4j-0.10.7-
src.zip")

sys.path.insert(0, os.environ["PYLIB"] +"/pyspark.zip")


#Now, initialize the entry points of Spark: SparkContext and
SparkConf

sc.stop()

from pyspark import SparkContext, SparkConf

conf = SparkConf().setAppName("appName")

sc = SparkContext.setAppName(conf=conf)

#Once you are successful in initializing the sc and conf,
please use the below code to test

rdd = sc.textFile("/user/surejadharmay4494/p10.txt")

print(rdd.take(10))

print(sc.version)
```




```
[surejadharmay4494@cxln5 ~]$ hadoop fs -copyFromLocal p10.txt
[surejadharmay4494@cxln5 ~]$
```

<u>TASK</u> : Create RDD using by distributing existing object (Using parallelized collection) in

SCALA Interface .

- Create an array of number 1 to 10000 and parellize it and take it first 100 numbers.

```
scala> var arr = 1 to 10000
arr: scala.collection.immutable.Range.Inclusive = Range(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 3
6, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 8
3, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 10
5, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 1
24, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161,
 162, 163, 164, 165, 166, 167, 168, 169, 170,...
scala> val nums = sc.parallelize(arr)
nums: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[1] at parallelize at <console>:26

scala> nums.take(100)
res3: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67
, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90,
91, 92, 93, 94, 95, 96, 97, 98, 99, 100)

scala>
```

- Create an array of weekdays and take the first 3 days of the week.

```
scala> var lines = sc.parallelize(Array("Sunday","Monday","Tuesday","wednesday","Thursday","Fr
iday","Saturday"))
lines: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[4] at parallelize at <console>
:24

scala> lines.take(3)
res5: Array[String] = Array(Sunday, Monday, Tuesday)

scala>
```

TASK : Use the Code and run in Python Interface (Using Jupyter Notebook) to find Average

Number of friends using the data given in and map the code understanding explained in the

lab and lectures

Code :

```
import os
import sys

os.environ["SPARK_HOME"] = "/usr/spark2.4.3"
os.environ["PYLIB"] = os.environ["SPARK_HOME"] +
"/python/lib"
# In below two lines, use /usr/bin/python2.7 if you want
to use Python 2
os.environ["PYSPARK_PYTHON"] =
"/usr/local/anaconda/bin/python"
os.environ["PYSPARK_DRIVER_PYTHON"] =
"/usr/local/anaconda/bin/python"
sys.path.insert(0, os.environ["PYLIB"] +"/py4j-0.10.7-
src.zip")
sys.path.insert(0, os.environ["PYLIB"] +"/pyspark.zip")

#Now, initialize the entry points of Spark: SparkContext
and SparkConf

from pyspark import SparkContext, SparkConf
conf =
SparkConf().setMaster("local").setAppName("FriendsByAge")
sc = SparkContext(conf = conf)

def parseLine(line):
    fields = line.split(',')
    age = int(fields[2])
    numFriends = int(fields[3])
    return (age, numFriends)


lines = sc.textFile("/user/
surejadharmay4494/4lecref_friends.csv")
```

```
rdd = lines.map(parseLine)
totalsByAge = rdd.mapValues(lambda x: (x,
1)).reduceByKey(lambda x, y: (x[0] + y[0], x[1] + y[1]))
averagesByAge = totalsByAge.mapValues(lambda x: x[0] /
x[1])
results = averagesByAge.collect();

for result in results:
     print(result)
```

Output :

In [1]:
```
import os
import sys

os.environ["SPARK_HOME"] = "/usr/spark2.4.3"
os.environ["PYLIB"] = os.environ["SPARK_HOME"] + "/python/lib"
```

In [2]:
```
# In below two lines, use /usr/bin/python2.7 if you want to use Python 2
os.environ["PYSPARK_PYTHON"] = "/usr/local/anaconda/bin/python"
os.environ["PYSPARK_DRIVER_PYTHON"] = "/usr/local/anaconda/bin/python"
sys.path.insert(0, os.environ["PYLIB"] +"/py4j-0.10.7-src.zip")
sys.path.insert(0, os.environ["PYLIB"] +"/pyspark.zip")
```

In [3]:
```
#Now, initialize the entry points of Spark: SparkContext and SparkConf

from pyspark import SparkContext, SparkConf
conf = SparkConf().setMaster("local").setAppName("FriendsByAge")
sc = SparkContext(conf = conf)

def parseLine(line):
    fields = line.split(',')
    age = int(fields[2])
    numFriends = int(fields[3])
    return (age, numFriends)


lines = sc.textFile("/user/surejadharmay4494/4lecref_friends.csv")
rdd = lines.map(parseLine)
totalsByAge = rdd.mapValues(lambda x: (x, 1)).reduceByKey(lambda x, y: (x[0] + y[0], x[1] + y[1]))
averagesByAge = totalsByAge.mapValues(lambda x: x[0] / x[1])
results = averagesByAge.collect();

for result in results:
     print(result)
```

```
(33, 325.3333333333333)
(26, 242.05882352941177)
(55, 295.53846153846155)
(40, 250.8235294117647)
(68, 269.6)
(59, 220.0)
(37, 249.33333333333334)
(54, 278.0769230769231)
```

```
(33, 325.3333333333333)
(26, 242.05882352941177)
(55, 295.53846153846155)
(40, 250.8235294117647)
(68, 269.6)
(59, 220.0)
(37, 249.33333333333334)
(54, 278.0769230769231)
(38, 193.53333333333333)
(27, 228.125)
(53, 222.85714285714286)
(57, 258.8333333333333)
(56, 306.6666666666667)
(43, 230.57142857142858)
(36, 246.6)
(22, 206.42857142857142)
(35, 211.625)
(45, 309.53846153846155)
(60, 202.71428571428572)
(67, 214.625)
(19, 213.27272727272728)
(30, 235.8181818181818)
(51, 302.14285714285717)
(25, 197.45454545454547)
(21, 350.875)
(42, 303.5)
(49, 184.66666666666666)
(48, 281.4)
(50, 254.6)
(39, 169.28571428571428)
(32, 207.9090909090909)
(58, 116.54545454545455)
(64, 281.3333333333333)
(31, 267.25)
(52, 340.6363636363636)
(24, 233.8)
(20, 165.0)
(62, 220.76923076923077)
(41, 268.55555555555554)
(44, 282.1666666666667)
(69, 235.2)
(65, 298.2)
(61, 256.22222222222223)
(28, 209.1)
(66, 276.44444444444446)
(46, 223.69230769230768)
(29, 215.91666666666666)
(18, 343.375)
(47, 233.22222222222223)
(34, 245.5)
(63, 384.0)
(23, 246.3)
```

TASK : Use the dataset given and check the code to find the minimum temperature by the

Location and understand it and modify it for to find the maximum temperature by the

Locations .

Code :

```
import os
import sys

os.environ["SPARK_HOME"] = "/usr/spark2.4.3"
os.environ["PYLIB"] = os.environ["SPARK_HOME"] +
"/python/lib"
# In below two lines, use /usr/bin/python2.7 if you want
to use Python 2
os.environ["PYSPARK_PYTHON"] =
"/usr/local/anaconda/bin/python"
os.environ["PYSPARK_DRIVER_PYTHON"] =
"/usr/local/anaconda/bin/python"
sys.path.insert(0, os.environ["PYLIB"] +"/py4j-0.10.7-
src.zip")
sys.path.insert(0, os.environ["PYLIB"] +"/pyspark.zip")

#Now, initialize the entry points of Spark: SparkContext
and SparkConf

from pyspark import SparkContext, SparkConf
sc.stop()
conf =
SparkConf().setMaster("local").setAppName("MinTemperature
s")
sc = SparkContext(conf = conf)

def parseLine(line):
    fields = line.split(',')
    stationID = fields[0]
    entryType = fields[2]
    temperature = float(fields[3]) * 0.1 * (9.0 / 5.0) +
32.0
    return (stationID, entryType, temperature)

lines = sc.textFile("/user/
surejadharmay44945temperatures.csv")
```

```
parsedLines = lines.map(parseLine)
minTemps = parsedLines.filter(lambda x: "TMIN" in x[1])
stationTemps = minTemps.map(lambda x: (x[0], x[2]))
minTemps = stationTemps.reduceByKey(lambda x, y:
min(x,y))
results = minTemps.collect();

for result in results:
    print(result[0] + "\t{:.2f}F".format(result[1]))
```

Output :

In [1]:
```
import os
import sys

os.environ["SPARK_HOME"] = "/usr/spark2.4.3"
os.environ["PYLIB"] = os.environ["SPARK_HOME"] + "/python/lib"
```

In [2]:
```
# In below two lines, use /usr/bin/python2.7 if you want to use Python 2
os.environ["PYSPARK_PYTHON"] = "/usr/local/anaconda/bin/python"
os.environ["PYSPARK_DRIVER_PYTHON"] = "/usr/local/anaconda/bin/python"
sys.path.insert(0, os.environ["PYLIB"] +"/py4j-0.10.7-src.zip")
sys.path.insert(0, os.environ["PYLIB"] +"/pyspark.zip")
```

In [6]:
```
#Now, initialize the entry points of Spark: SparkContext and SparkConf

from pyspark import SparkContext, SparkConf
sc.stop()
conf = SparkConf().setMaster("local").setAppName("MinTemperatures")
sc = SparkContext(conf = conf)
```

In [7]:
```
def parseLine(line):
    fields = line.split(',')
    stationID = fields[0]
    entryType = fields[2]
    temperature = float(fields[3]) * 0.1 * (9.0 / 5.0) + 32.0
    return (stationID, entryType, temperature)

lines = sc.textFile("/user/surejadharmay4494/5temperatures.csv")
parsedLines = lines.map(parseLine)
minTemps = parsedLines.filter(lambda x: "TMIN" in x[1])
stationTemps = minTemps.map(lambda x: (x[0], x[2]))
minTemps = stationTemps.reduceByKey(lambda x, y: min(x,y))
results = minTemps.collect();

for result in results:
    print(result[0] + "\t{:.2f}F".format(result[1]))
```

```
ITE00100554     5.36F
EZE00100082     7.70F
```

TASK : Use the given dataset of customers and their spend and find how much amount is spent

By the individual customer total creating proper RDD in the spark using Python .

Code :

```
import os

import sys

os.environ["SPARK_HOME"] = "/usr/spark2.4.3"

os.environ["PYLIB"] = os.environ["SPARK_HOME"] + "/python/lib"

# In below two lines, use /usr/bin/python2.7 if you want to
use Python 2

os.environ["PYSPARK_PYTHON"] =
"/usr/local/anaconda/bin/python"

os.environ["PYSPARK_DRIVER_PYTHON"] =
"/usr/local/anaconda/bin/python"

sys.path.insert(0, os.environ["PYLIB"] +"/py4j-0.10.7-
src.zip")

sys.path.insert(0, os.environ["PYLIB"] +"/pyspark.zip")

#Now, initialize the entry points of Spark: SparkContext and
SparkConf

sc.stop()

from pyspark import SparkContext, SparkConf

conf = SparkConf().setAppName("appName")

sc = SparkContext(conf=conf)

#Once you are successful in initializing the sc and conf,
please use the below code to test

def extractCustomerPricePairs(line):

    fields = line.split(',')

    return (int(fields[0]), float(fields[2]))

rdd = sc.textFile("hdfs:///user/surejadharmay4494/6customer-
orders.csv")
```

```
mappedInput = rdd.map(extractCustomerPricePairs)

totalByCustomer = mappedInput.reduceByKey(lambda x, y: x + y)

results = totalByCustomer.collect()


for result in results:

    print(result)
```

Output:-

In [1]:
```
import os
import sys
os.environ["SPARK_HOME"] = "/usr/spark2.4.3"
os.environ["PYLIB"] = os.environ["SPARK_HOME"] + "/python/lib"
```

In [2]:
```
# In below two lines, use /usr/bin/python2.7 if you want to use Python 2
os.environ["PYSPARK_PYTHON"] = "/usr/local/anaconda/bin/python"
os.environ["PYSPARK_DRIVER_PYTHON"] = "/usr/local/anaconda/bin/python"
sys.path.insert(0, os.environ["PYLIB"] +"/py4j-0.10.7-src.zip")
sys.path.insert(0, os.environ["PYLIB"] +"/pyspark.zip")
```

In [4]:
```
#Now, initialize the entry points of Spark: SparkContext and SparkConf
#sc.stop()
from pyspark import SparkContext, SparkConf
conf = SparkConf().setAppName("appName")
sc = SparkContext(conf=conf)
```

In [5]:
```
#Once you are successful in initializing the sc and conf, please use the below
def extractCustomerPricePairs(line):
    fields = line.split(',')
    return (int(fields[0]), float(fields[2]))
rdd = sc.textFile("hdfs:///user/surejadharmay4494/6customer-orders.csv")
mappedInput = rdd.map(extractCustomerPricePairs)
totalByCustomer = mappedInput.reduceByKey(lambda x, y: x + y)
results = totalByCustomer.collect()

for result in results:
    print(result)
```

```
(44, 4756.890000000001)
(2, 5994.59)
(70, 5368.249999999999)
(14, 4735.030000000001)
(42, 5696.840000000002)
(50, 4517.2699999999995)
(20, 4836.860000000001)
(48, 4384.33)
(4, 4815.050000000001)
(36, 4278.049999999999)
```

```
(51, 4975.220000000001)
(79, 3790.5699999999997)
(15, 5413.51)
(5, 4561.07)
(31, 4765.049999999999)
(57, 4628.4)
(13, 4367.619999999999)
(55, 5298.089999999999)
(95, 4876.839999999998)
(61, 5497.4800000000005)
(27, 4915.89)
(83, 4635.8)
(75, 4178.5)
(25, 5057.610000000001)
(71, 5995.660000000002)
(39, 6193.110000000001)
(97, 5977.1900000000005)
(7, 4755.069999999999)
(21, 4707.41)
(69, 5123.01)
(37, 4735.200000000001)
(1, 4958.600000000001)
(99, 4172.29)
(73, 6206.200000000001)
(49, 4394.6)
(23, 4042.65)
(19, 5059.43)
(65, 5140.35)
(9, 5322.65)
(59, 5642.889999999999)
(11, 5152.289999999999)
(41, 5637.620000000001)
(87, 5206.4)
(17, 5032.68)
(33, 5254.660000000002)
(43, 5368.83)
(77, 4327.73)
(81, 5112.709999999999)
(3, 4659.63)
(93, 5265.75)
(89, 4851.48)
(45, 3309.3799999999997)
(67, 4505.79)
(63, 5415.1500000000015)
```