```
In [1]: import pyspark as spark
```

```
In [2]: sc = spark.SparkContext()
```

```
24/10/14 15:10:43 WARN Utils: Your hostname, HP-Victus resolves to a loopback address: 127.0.1.1;
using 10.255.255.254 instead (on interface lo)
24/10/14 15:10:43 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/10/14 15:10:44 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform...
using builtin-java classes where applicable
```

Friends Test

1. Use the “friends_test” dataset. Col1 is ID, Col2 is name, Col 3 is Age, Col 4 is num of friends. Understand mapvalues function of RDD in spark and find the average number of friends for each unique age present in the dataset.

```
In [3]: friends = sc.textFile('friends_test.csv')
```

```
In [4]: age_grp = friends.map(lambda x: x.split(',')[2:4]) \
                    .map(lambda x: (int(x[0]), int(x[1]))) \
                    .groupByKey() \
                    .mapValues(lambda x : round(sum(x) / len(x), 2)) \
                    .sortByKey()
```

In [5]:
```
for age, avg in age_grp.collect():
    print(f"Age: {age}\t\t Average Friends: {avg}")
```

```
Age: 18          Average Friends: 343.38
Age: 19          Average Friends: 213.27
Age: 20          Average Friends: 165.0
Age: 21          Average Friends: 350.88
Age: 22          Average Friends: 206.43
Age: 23          Average Friends: 246.3
Age: 24          Average Friends: 233.8
Age: 25          Average Friends: 197.45
Age: 26          Average Friends: 242.06
Age: 27          Average Friends: 228.12
Age: 28          Average Friends: 209.1
Age: 29          Average Friends: 215.92
Age: 30          Average Friends: 235.82
Age: 31          Average Friends: 267.25
Age: 32          Average Friends: 207.91
Age: 33          Average Friends: 325.33
Age: 34          Average Friends: 245.5
Age: 35          Average Friends: 211.62
Age: 36          Average Friends: 246.6
Age: 37          Average Friends: 249.33
Age: 38          Average Friends: 193.53
Age: 39          Average Friends: 169.29
Age: 40          Average Friends: 250.82
Age: 41          Average Friends: 268.56
Age: 42          Average Friends: 303.5
Age: 43          Average Friends: 230.57
Age: 44          Average Friends: 282.17
Age: 45          Average Friends: 309.54
Age: 46          Average Friends: 223.69
Age: 47          Average Friends: 233.22
Age: 48          Average Friends: 281.4
Age: 49          Average Friends: 184.67
Age: 50          Average Friends: 254.6
Age: 51          Average Friends: 302.14
Age: 52          Average Friends: 340.64
Age: 53          Average Friends: 222.86
Age: 54          Average Friends: 278.08
Age: 55          Average Friends: 295.54
Age: 56          Average Friends: 306.67
Age: 57          Average Friends: 258.83
Age: 58          Average Friends: 116.55
Age: 59          Average Friends: 220.0
Age: 60          Average Friends: 202.71
Age: 61          Average Friends: 256.22
Age: 62          Average Friends: 220.77
Age: 63          Average Friends: 384.0
Age: 64          Average Friends: 281.33
Age: 65          Average Friends: 298.2
Age: 66          Average Friends: 276.44
Age: 67          Average Friends: 214.62
Age: 68          Average Friends: 269.6
Age: 69          Average Friends: 235.2
```

Temp

2. Use the “temp.csv” dataset. Column headers are present in the dataset. Understand filter operations and filter out only the “TMIN” values from the “desc” column. With the resultant data (RDD) find the following: a. Minimum temperature (overall) b. Minimum temperature for every ItemID c. Minimum temperature for every StationID.
3. Use the same dataset, filter only “TMAX” column and find the maximum temperatures just like the ones mentioned above.

In [6]:
```
lines = sc.textFile('temp.csv')
```

In [7]:
```
header = lines.first()
header
```

Out[7]:
```
'itemID,stationID,desc,temp'
```

In [8]:
```python
lines = lines.filter(lambda line: line != header) \
                .map(lambda line: line.split(',')) \
                .map(lambda line: (line[0], line[1], line[2], int(line[3])))
lines.take(5)
```

Out[8]:
```
[('ITE00100554', '18000101', 'TMAX', -75),
 ('ITE00100554', '18000101', 'TMIN', -148),
 ('GM000010962', '18000101', 'PRCP', 0),
 ('EZE00100082', '18000101', 'TMAX', -86),
 ('EZE00100082', '18000101', 'TMIN', -135)]
```

Min Temp

In [9]:
```python
minTemp = lines.filter(lambda x: "TMIN" in x[2])
minTemp.take(5)
```

Out[9]:
```
[('ITE00100554', '18000101', 'TMIN', -148),
 ('EZE00100082', '18000101', 'TMIN', -135),
 ('ITE00100554', '18000102', 'TMIN', -125),
 ('EZE00100082', '18000102', 'TMIN', -130),
 ('ITE00100554', '18000103', 'TMIN', -46)]
```

In [10]:
```python
minOverall = minTemp.map(lambda x: x[-1]).min()
print(f"Min. Temp Overall: {minOverall}")
```

```
Min. Temp Overall: -148
```

In [11]:
```python
minItemID = minTemp.map(lambda x: (x[0], x[3])).reduceByKey(lambda x,y: min(x,y))
for item, temp in minItemID.collect():
    print(f"ItemID: {item}\tMin.Temp: {temp}")
```

```
ItemID: ITE00100554     Min.Temp: -148
ItemID: EZE00100082     Min.Temp: -135
```

In [12]:
```python
minStationID = minTemp.map(lambda x: (x[1], x[3])).reduceByKey(lambda x,y: min(x,y))
for station, temp in minStationID.collect():
    print(f"StationID: {station}\tMin.Temp: {temp}")
```

```
StationID: 18000102     Min.Temp: -130
StationID: 18000104     Min.Temp: -74
StationID: 18000106     Min.Temp: -57
StationID: 18000110     Min.Temp: -75
StationID: 18000111     Min.Temp: -62
StationID: 18000112     Min.Temp: -60
StationID: 18000114     Min.Temp: -35
StationID: 18000115     Min.Temp: -23
StationID: 18000116     Min.Temp: -37
StationID: 18000117     Min.Temp: -35
StationID: 18000118     Min.Temp: 9
StationID: 18000122     Min.Temp: -16
StationID: 18000124     Min.Temp: -3
StationID: 18000126     Min.Temp: 16
StationID: 18000127     Min.Temp: 15
StationID: 18000128     Min.Temp: 33
StationID: 18000130     Min.Temp: 3
StationID: 18000202     Min.Temp: 19
StationID: 18000205     Min.Temp: 22
```

Max Temp

In [13]:
```python
maxTemp = lines.filter(lambda x: "TMAX" in x[2])
maxTemp.take(5)
```

Out[13]:
```
[('ITE00100554', '18000101', 'TMAX', -75),
 ('EZE00100082', '18000101', 'TMAX', -86),
 ('ITE00100554', '18000102', 'TMAX', -60),
 ('EZE00100082', '18000102', 'TMAX', -44),
 ('ITE00100554', '18000103', 'TMAX', -23)]
```

In [14]:
```python
maxOverall = maxTemp.map(lambda x: x[-1]).max()
print(f"Max. Temp Overall: {maxOverall}")
```

```
Max. Temp Overall: 323
```

In [15]:
```python
maxItemID = maxTemp.map(lambda x: (x[0], x[3])).reduceByKey(lambda x,y: max(x,y))
for item, temp in maxItemID.collect():
    print(f"ItemID: {item}\tMax.Temp: {temp}")
```

```
ItemID: ITE00100554     Max.Temp: 323
ItemID: EZE00100082     Max.Temp: 323
```

In [16]:
```python
maxStationID = maxTemp.map(lambda x: (x[1], x[3])).reduceByKey(lambda x,y: max(x,y))
for station, temp in maxStationID.collect():
    print(f"StationID: {station}\tMax.Temp: {temp}")
```

```
StationID: 18000102     Max.Temp: -44
StationID: 18000104     Max.Temp: 0
StationID: 18000106     Max.Temp: 13
StationID: 18000110     Max.Temp: 46
StationID: 18000111     Max.Temp: 66
StationID: 18000112     Max.Temp: 41
StationID: 18000114     Max.Temp: 41
StationID: 18000115     Max.Temp: 54
StationID: 18000116     Max.Temp: 56
StationID: 18000117     Max.Temp: 84
StationID: 18000118     Max.Temp: 59
StationID: 18000122     Max.Temp: 81
StationID: 18000124     Max.Temp: 85
StationID: 18000126     Max.Temp: 75
StationID: 18000127     Max.Temp: 73
StationID: 18000128     Max.Temp: 79
StationID: 18000130     Max.Temp: 66
StationID: 18000202     Max.Temp: 67
StationID: 18000205     Max.Temp: 79
```