In [1]:
```
from pyspark.sql import SparkSession

spark = SparkSession \
        .builder \
        .master("local[*]") \
        .appName("aggregrate Transformation") \
        .enableHiveSupport() \
        .getOrCreate()

#path of the data file on the local machine
data_file = '/Users/vaishaliyasala/Desktop/Github/Spark/Exercise_Dependencies/sales_data.csv'

#Read the csv into a dataframe
df = spark.read.csv(data_file, header = True, )

df1 = df.select(df["InvoiceNo"],df["Quantity"]).repartition(4)

print(df1.printSchema())

#Creating view of the dataframe of with 3 required columns and sample of 3% of data
sample_df = df1.sample(0.01,25)

sample_df.show()
```

```
22/10/13 17:50:05 WARN Utils: Your hostname, Vaishalis-MacBook-Pro.local resolves to a loopback address: 127.0.
0.1; using 192.168.0.105 instead (on interface en0)
22/10/13 17:50:05 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).
22/10/13 17:50:06 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builti
n-java classes where applicable

root
 |-- InvoiceNo: string (nullable = true)
 |-- Quantity: string (nullable = true)

None
+---------+--------+
|InvoiceNo|Quantity|
+---------+--------+
|   536399|       6|
|   536488|       1|
|   536520|       3|
|   536520|      12|
|   536384|      12|
|   536396|       2|
|   536488|       5|
|   536425|      12|
+---------+--------+
```

In [2]:
```
# apply a map() transformation to rdd to create (K, V) pairs

#In this key-value pair, key is the InvoiceN and Quantity is the value

rdd1 = df1.rdd.map(lambda x : (x[0],int(x[1])))
print("Number of elements =",len(rdd1.collect()))
print("Number of Partitions =",rdd1.getNumPartitions())
```

```
Number of elements = 999
Number of Partitions = 4
```

In [3]:
```
# apply a reduceByKey() transformation on rdd1 to create a (key, value) pair

#  where key is the InvoiceNo and value is sum of prices for each key

#we can create more partitions than its parent RDD.

rdd2 = rdd1.reduceByKey(lambda a, b: a+b)

print("Number of elements =",len(rdd2.collect()))
print("Number of Partitions =",rdd2.getNumPartitions())
print(rdd2.take(10))
```

```
Number of elements = 66
Number of Partitions = 4
[('536401', 124), ('536412', 220), ('536460', 158), ('C536391', -132), ('536416', 110), ('536420', 111), ('5363
85', 53), ('536508', 216), ('536369', 3), ('536405', 128)]
```

In [4]:
```
#Sort by key ascending

sort_key = rdd2.sortByKey(ascending = True)
print("Number of elements =",sort_key.take(10))
```

```
Number of elements = [('536365', 40), ('536366', 12), ('536367', 83), ('536368', 15), ('536369', 3), ('536370',
449), ('536371', 80), ('536372', 12), ('536373', 88), ('536374', 32)]
```

In [5]:
```
#Sort by key descending

sort_key = rdd2.sortByKey(ascending = False)
print("Number of elements =",sort_key.take(10))
```

```
Number of elements = [('C536506', -6), ('C536391', -132), ('C536383', -1), ('C536379', -1), ('536520', 154),
('536514', 118), ('536508', 216), ('536502', 39), ('536500', 102), ('536488', 72)]
```