ET:13 Given the following paired RDDs

RDD1 = {(1, 2), (3, 4), (3, 6)}
RDD2 = {(3, 9) (3, 6)}

Using PySpark, write code to perform an inner join of these paired RDDs. What is the resulting RDD? Make your Spark available in your notebook:

A: [(3, (4, 9)), (3, (6, 9))]
B: [(3, (4, 9)), (3, (4, 6)), (3, (6, 9)), (3, (6, 6))]
C: [(3, (4, 9)), (3, (4, 6)), (3, (6, 9)), (3, (6, 9))]
D: None of the above

In [1]:

```
%matplotlib inline
import os
import sys #current as of 9/26/2015
spark_home = os.environ['SPARK_HOME'] = \
   '/opt/spark16'

sys.path.insert(0,os.path.join(spark_home,'python'))
sys.path.insert(0,os.path.join(spark_home,'python/lib/py4j-0.8.2.1-src.zip'))
execfile(os.path.join(spark_home,'python/pyspark/shell.py'))
```

```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.6.1
      /_/

Using Python version 2.7.10 (default, Oct 23 2015 19:19:21)
SparkContext available as sc, SQLContext available as sqlContext.
```

In [2]:

```
RDD1 = sc.parallelize([(1, 2), (3, 4), (3, 6)])
RDD2 = sc.parallelize([(3, 9), (3, 6)])
RDD1.join(RDD2).collect()
```

Out[2]:

```
[(3, (4, 9)), (3, (4, 6)), (3, (6, 9)), (3, (6, 6))]
```

Answer: B

ET:15

Use Spark and the following notebook, https://www.dropbox.com/s/6s5ph41h74bggwi/Linear-Regression-on-Beer-Data.ipynb?dl=0 (https://www.dropbox.com/s/6s5ph41h74bggwi/Linear-Regression-on-Beer-Data.ipynb?dl=0) to answer this question.

The mean absolute percentage error (MAPE), also known as mean absolute percentage deviation (MAPD), is a measure of prediction accuracy of a model for say a forecasting method in statistics, for example in trend estimation. It usually expresses accuracy as a percentage, and is defined by the formula:

MAPE = average over all examples (100*Abs(Actual - Predicted) / Actual))

Note when Actual is zero that test row is dropped from the evaluation.

Construct a mean model for target variable CASES18PK. Calculate the MAPE for the mean model over the training set. Select the closest answer.

(a) 200% (b) 250% (c) 20% (d) 180%

In [4]:

```
df=sc.textFile("beerSales.txt")
```

In [24]:

```
nums= df.map(lambda x: x.split("\t")[2]).\
        filter(lambda x: '.' in x).\
        map(lambda x: float(x))
mean=  nums.reduce(lambda x,y: x+y) / nums.count()
```

In [25]:

```
mean
```

Out[25]:

```
16.724615384615387
```

In [28]:

```
MAPE = lambda x, p: 100*abs(x-p)/x
```

In [30]:

```
nums.map(lambda x: MAPE(x, mean)).reduce(lambda x,y: x+y) / nums.count()
```

Out[30]:

```
14.581300722435568
```

Answer: C

ET:16

Use Spark and the following notebook, https://www.dropbox.com/s/6s5ph41h74bggwi/Linear-Regression-on-Beer-Data.ipynb?dl=0 (https://www.dropbox.com/s/6s5ph41h74bggwi/Linear-Regression-on-Beer-Data.ipynb?dl=0) to answer this question.

The target variable CASES18PK is skewed, so take the log of it (and make it more normally distributed) and compute the MAPE of the mean model for CASES18PK Select the closest answer to your calculated MAPE.

(a) 200%
(b) 30%
(c) 20%
(d) 10%

In [33]:

```
from math import log
nums_log = nums.map(lambda x: log(x))
mean=  nums_log.reduce(lambda x,y: x+y) / nums_log.count()
```

In [34]:

```
mean
```

Out[34]:

```
2.80630886928261
```

In [36]:

```
nums_log.map(lambda x: MAPE(x, mean)).reduce(lambda x,y: x+y) / nums_log.count()
```

Out[36]:

```
5.165527753854489
```

Answer: D

ET:17

Use Spark and the following notebook, https://www.dropbox.com/s/6s5ph41h74bggwi/Linear-Regression-on-Beer-Data.ipynb?dl=0 (https://www.dropbox.com/s/6s5ph41h74bggwi/Linear-Regression-on-Beer-Data.ipynb?dl=0) to answer this question.

Build a linear regression model using the following variables:

Log(CASES18PK) ~ log(PRICE12PK), log(PRICE18PK), log(PRICE30PK)

Calculate MAPE over the test set and select the closest answer.

(a) 4.3%
(b) 4.6%
(c) 3.5%
(d) 3.9%

In [125]:

```
from pyspark.mllib.regression import LinearRegressionWithSGD, LabeledPoint
import numpy as np

def mapPoints(x):
    _, PRICE12PK, \
        PRICE18PK, PRICE30PK, \
        CASES12PK, CASES18PK, \
        CASES30PK                = x.split("\t")
    dvars = map(float,[PRICE12PK, PRICE18PK, PRICE30PK])
    return LabeledPoint(log(float(CASES18PK)), map(log,dvars))

df_points= df.filter(lambda x: '.' in x)\
        .map(lambda x: mapPoints(x))
```

In [220]:

```
model = LinearRegressionWithSGD.train(df_points\
                                       , 2000\
                                       , regParam=0.01
                                       , regType='l2'
                                       , step=0.1
                                       , intercept=True
                                       , convergenceTol=0.00001)
```

In [221]:

```
non_labeled = df_points.map(lambda x: x.features)
labels = df_points.map(lambda x: x.label)
preds = model.predict(non_labeled)
```

In [222]:

```
labels.zip(preds).map(lambda x: MAPE(x[0],x[1])).reduce(lambda x,y: x+y) / preds.cou
```

Out[222]:

```
17.706339967881924
```