Lab_Mar30

```
%pyspark
from pandas import Series, DataFrame
import pandas as pd
import numpy as np
```

FINISHED

Took 0 sec. Last updated by anonymous at March 30 2017, 6:58:48 PM.

```
%pyspark
data_iwm = pd.read_csv('/Users/datascienceadmin/Downloads/iwm.csv', sep=",", parse_dates=True
data_iwm.info()
df = data_iwm.ix[1:]
print(df.shape)
df1= df.ix[1:1000,]
df2=df.ix[1001:2000,]
print(df1.shape)
print(df2.shape)

data1 = pd.to_numeric(df.Open)
data2 = pd.to_numeric(df.Close)
data3 = pd.to_numeric(df['High'])

type(data1)
#print(df2)
```

FINISHED

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 4205 entries, 2017-02-13 to 2000-05-26
Data columns (total 6 columns):
Open         4205 non-null float64
High         4205 non-null float64
Low          4205 non-null float64
Close        4205 non-null float64
Volume       4205 non-null int64
Adj Close    4205 non-null float64
dtypes: float64(5), int64(1)
memory usage: 230.0 KB
(4204, 6)
(999, 6)
(999, 6)
<class 'pandas.core.series.Series'>
```

Took 0 sec. Last updated by anonymous at March 30 2017, 8:15:44 PM.

```
%pyspark
frame = DataFrame({'Open': data1, 'Close': data2, 'High': data3})
#factor = pd.cut(frame.data1,4)
#factor[:10]
frame
```

FINISHED

```
2000-06-20  105.187500  105.343758  104.656242
2000-06-19  104.500000  104.500000  102.500000
2000-06-16  101.937500  103.000000  102.937500
2000-06-15  102.500000  102.500000  101.750000
2000-06-14  102.375000  103.718758  103.718758
2000-06-13  102.875000  102.875000  101.328117
2000-06-12  101.578117  105.000000  105.000000
2000-06-09  104.687500  104.687500  103.875000
2000-06-08  102.875000  104.250000  104.250000
2000-06-07  103.125000  103.125000  102.375000
2000-06-06  103.000000  104.812500  103.609383
2000-06-05  102.000000  103.125000  102.125000
2000-06-02  102.375000  102.375000  101.718758
2000-06-01   97.312500   97.312500   97.109383
2000-05-31   95.156242   96.375000   95.125000
2000-05-30   94.812500   94.812500   92.750000
2000-05-26   91.437500   91.437500   91.062500
[4204 rows x 3 columns]
```

Took 0 sec. Last updated by anonymous at March 30 2017, 8:15:56 PM.

FINISHED

```
%pyspark
rets = frame.pct_change().dropna()
rets
```

```
2000-06-20 -0.004731 -0.005018  0.003897
2000-06-19 -0.006536 -0.008010 -0.020603
2000-06-16 -0.024522 -0.014354  0.004268
2000-06-15  0.005518 -0.004854 -0.011536
2000-06-14 -0.001220  0.011890  0.019349
2000-06-13  0.004884 -0.008135 -0.023049
2000-06-12 -0.012606  0.020656  0.036238
2000-06-09  0.030611 -0.002976 -0.010714
2000-06-08 -0.017313 -0.004179  0.003610
2000-06-07  0.002430 -0.010791 -0.017986
2000-06-06 -0.001212  0.016364  0.012057
2000-06-05 -0.009709 -0.016100 -0.014327
2000-06-02  0.003676 -0.007273 -0.003978
2000-06-01 -0.049451 -0.049451 -0.045315
2000-05-31 -0.022158 -0.009634 -0.020435
2000-05-30 -0.003612 -0.016213 -0.024967
2000-05-26 -0.035597 -0.035597 -0.018194
[4203 rows x 3 columns]
```

Took 0 sec. Last updated by anonymous at March 30 2017, 8:16:04 PM.

FINISHED

```
%pyspark
open_corr = lambda x: x.corrwith(x['Open'])
open_corr

by_year = rets.groupby(lambda x: x.year)
by_year.apply(open_corrClose)
```

```
2000  0.211016  0.751631   1.0
2001  0.232025  0.664719   1.0
2002  0.133391  0.740484   1.0
2003  0.191029  0.709983   1.0
2004  0.116431  0.680469   1.0
2005  0.978898  0.993194   1.0
2006  0.141991  0.701554   1.0
2007  0.190359  0.691831   1.0
2008  0.104910  0.691846   1.0
2009  0.226277  0.680368   1.0
2010  0.324483  0.763784   1.0
2011  0.338510  0.775950   1.0
2012  0.307069  0.760884   1.0
2013  0.164838  0.701875   1.0
2014  0.164515  0.709029   1.0
2015  0.271998  0.706422   1.0
2016  0.268257  0.695582   1.0
2017  0.135370  0.558339   1.0
```

Took 0 sec. Last updated by anonymous at March 30 2017, 8:16:38 PM.

```
%pyspark
# Annual correlation of Closing prices with High prices for stocks
by_year.apply(lambda g: g['Close'].corr(g['High']))
```

FINISHED

```
2001    0.670741
2002    0.569075
2003    0.601328
2004    0.562687
2005    0.991517
2006    0.563527
2007    0.583641
2008    0.505477
2009    0.682910
2010    0.663096
2011    0.654911
2012    0.688828
2013    0.581989
2014    0.610730
2015    0.630977
2016    0.736297
2017    0.769283
dtype: float64
```

Took 0 sec. Last updated by anonymous at March 30 2017, 8:16:55 PM.

```
%pyspark
import statsmodels.api as sm
```

FINISHED

Took 4 sec. Last updated by anonymous at March 30 2017, 8:26:51 PM.

```
%pyspark
def regression(data, yvar, xvars):
  Y = data[yvar]
  X = data[xvars]
```

FINISHED

```
    X['intercept'] = 1.
    result = sm.OLS(Y,X).fit()
    return result.params
```

Took 0 sec. Last updated by anonymous at March 30 2017, 8:26:54 PM.

```
%pyspark
by_year.apply(regression,'Close',['Open'])
```

FINISHED

```
          Open  intercept
2000  0.210685   0.000124
2001  0.209317  -0.000107
2002  0.124583   0.000882
2003  0.186596  -0.001106
2004  0.121176  -0.000392
2005  0.944552   0.000025
2006  0.137710  -0.000412
2007  0.199065   0.000202
2008  0.094961   0.001804
2009  0.245712  -0.000537
2010  0.341360  -0.000484
2011  0.362229   0.000317
2012  0.280029  -0.000402
2013  0.177912  -0.000848
2014  0.150396  -0.000097
2015  0.271727   0.000262
2016  0.271430  -0.000553
```

Took 0 sec. Last updated by anonymous at March 30 2017, 8:27:39 PM.

```
%pyspark
```

READY