Zaman_Shazia_MSDS_7333_Case_Study_4

February 12, 2018

Name: Shazia Zaman, Case Study: 2, Class: MSDS 7333-402

In [233]:
```
import pandas as pd
import numpy as np
import pandas.io.data as web
from datetime import datetime, date
import matplotlib.pylab as plt
from scipy.stats import percentileofscore
from collections import defaultdict
%matplotlib inline
```

0.1 Initial Analysis

I have decided to compare the stocks for some of popular retail stores for last 10 years. List of stocks used in this report are following: Dillard's (DDS), Kohl's (KSS), Macys (M), Nordstorm (JWN), JCPenny (JCP)

In [308]:
```
#Initialize start and end date to capture stocks
start = datetime(2006,1,1)
end = date.today()

# Stock symbol list to include in the report
retail_symbol_list = ['DDS','KSS','M','JWN','JCP']
# S&P 500 stocks will be used later in the report for benchmark analysis
S_and_P = ['SPY']
stock_symbol_list = retail_symbol_list + S_and_P
#get stock data from yahoo finance
stocks = web.DataReader(stock_symbol_list, "yahoo", start, end)
type(stocks)
```

Out[308]:
```
pandas.core.panel.Panel
```

In [309]:
```
# check types of reports and types
stocks.dtypes
```

Out[309]:
```
Open          float64
High          float64
Low           float64
```

```
Close          float64
Volume         float64
Adj Close      float64
dtype: object
```

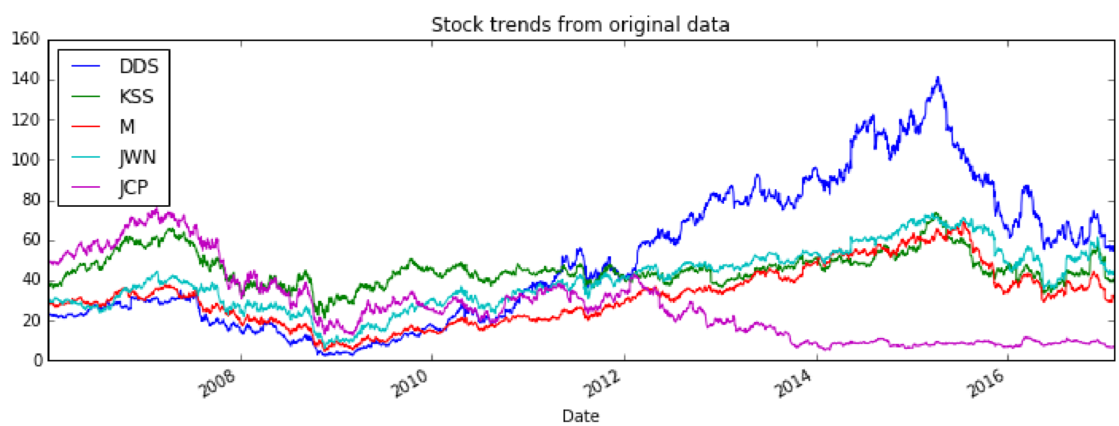
In [310]:
```
#As the data return from yahoo finance is two dimensional by report type and stocks
#Choose to evaluate report for Adj Close

adj_close_px=stocks['Adj Close'][retail_symbol_list]
adj_close_px.head()
```

Out[310]:

Date	DDS	KSS	M	JWN	JCP
2006-01-03	22.181773	39.644398	27.178371	27.825640	49.139058
2006-01-04	22.443970	39.318004	27.517005	27.788753	49.818037
2006-01-05	22.400270	36.773798	27.596682	29.264131	48.233749
2006-01-06	22.269172	37.242469	27.871574	29.138726	48.848834
2006-01-09	22.854743	37.142041	28.592665	29.581338	48.944801

In [311]:
```
#Evaluate the stock trends from original data
adj_close_px.plot(figsize=(12,4),title='Stock trends from original data')
```

Out[311]: <matplotlib.axes._subplots.AxesSubplot at 0x26a8877cd68>



As seen from the chart for 'Stock trends from original data' , all five of the retail businesses have suffered from 2007-2009 recesion. Then stocks have picked up from in rising trends from most of the retailers from 2009 till 2011 and then stabilize for another year

In [344]:
```
#Resample the data for business day frequency with resample rule 'B'
adj_close_px = adj_close_px.resample('B').ffill()
adj_close_px.head()
```

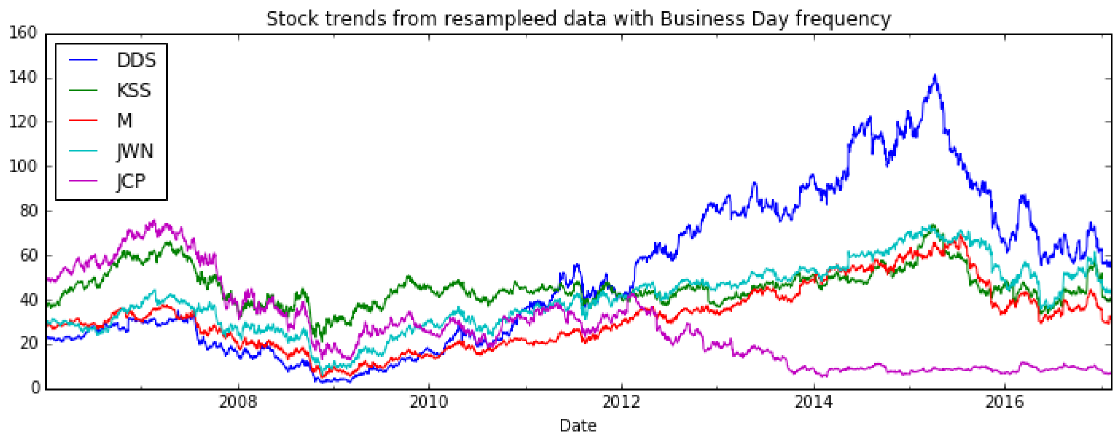
Out[344]:

Date	DDS	KSS	M	JWN	JCP

```
2006-01-03  22.181773  39.644398  27.178371  27.825640  49.139058
2006-01-04  22.443970  39.318004  27.517005  27.788753  49.818037
2006-01-05  22.400270  36.773798  27.596682  29.264131  48.233749
2006-01-06  22.269172  37.242469  27.871574  29.138726  48.848834
2006-01-09  22.854743  37.142041  28.592665  29.581338  48.944801
```

In [345]:
```
#Evaluate the stock trends from resample data
adj_close_px.plot(figsize=(12,4),title='Stock trends from resampleed data with Busin
```

Out[345]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x26a8bddf320>
```



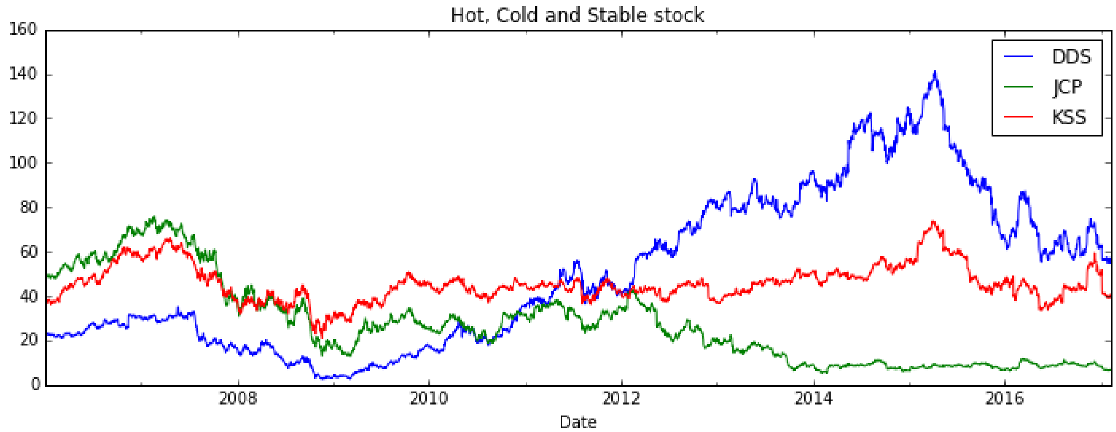
From the chart 'Stock trends from resampled data with Business Day Frequency', it is still evident that JCP stock for JCPenny has suffered from recesion of 2007-2009 and then rised again for next couple of year until 2012. Then it has started declining again. However, on the other side DDS for Dillard's has been rising up after recovery from recesion period until around 2015, and then it has joined the group of retailers who have sustained themselves after recession, for example Macys (M), Kohl's (KSS) and Nordstorm (JWN). One more interesting fact is the Kohl's (KSS) was minimally damages during the recession period of 2007-2009, and it is able to maintained its stock prices least variations than other retailers involved in this report.

0.2 Momentum Strategy to eliminate hot vesus cold stocks

For next couple of steps I will focus on rolling means and standard deviation for three stocks to compare hot, cold and maintained stocks that is Dillard's (DDS), JCPenny (JCP) and Kohl's (KSS) respectively.

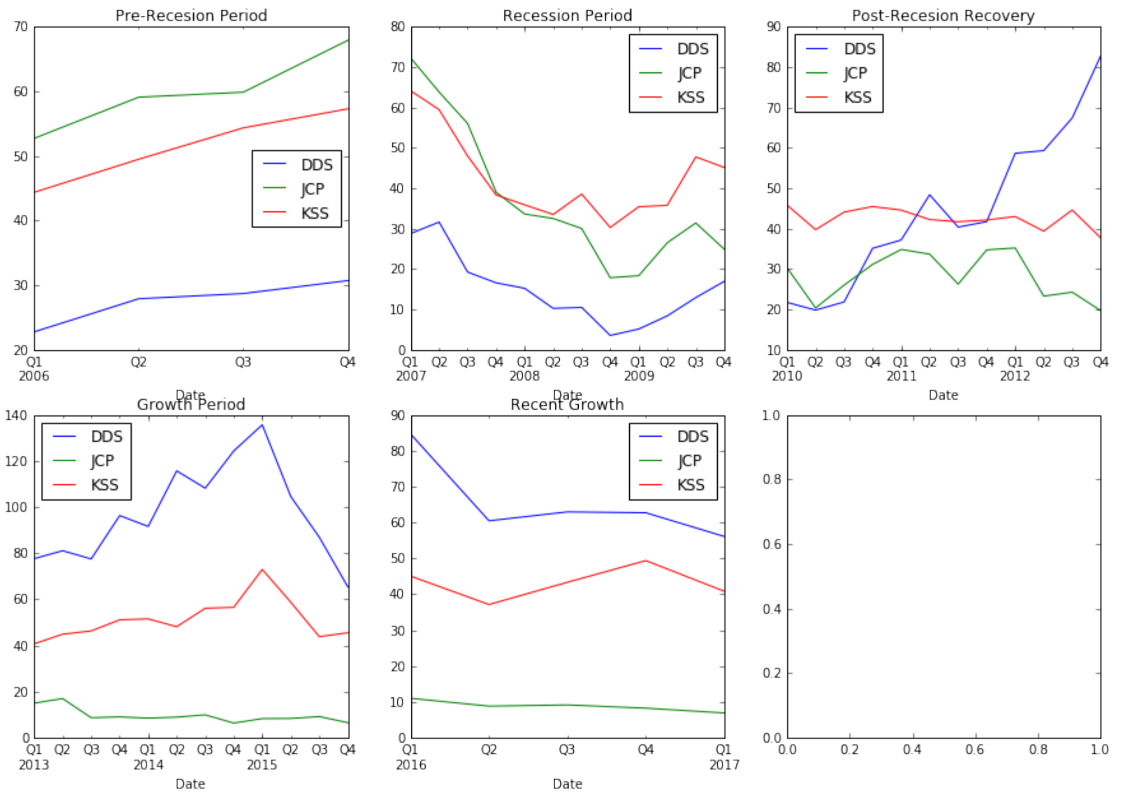
In [363]:
```
#Evaluate the stock for DDS, JCP and KSS
hot_cold_stable_px = adj_close_px[['DDS','JCP','KSS']]
hot_cold_stable_px.plot(figsize=(12,4), title='Hot, Cold and Stable stock')
```

Out[363]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x26a90beb9b0>
```



From the chart 'Hot, Cold and Stable stock' it is evident that Kohl's has been able to stabilize the stock before and after recession and even during recession, it was the least impacted stock. Next I will evaluate the stocks based on quarterly summary.

```
In [365]: # Check the trends for Quarterly end of frequency sample
          hot_cold_stable_q = hot_cold_stable_px.resample('Q-DEC').ffill()
          fig, axes = plt.subplots(nrows=2,ncols=3,sharex=False,sharey=False,figsize=(15,10))
          hot_cold_stable_q.ix[:'2006'].plot(ax=axes[0][0],title='Pre-Recesion Period')
          hot_cold_stable_q.ix['2007':'2009'].plot(ax=axes[0][1],title='Recession Period')
          hot_cold_stable_q.ix['2010':'2012'].plot(ax=axes[0][2],title='Post-Recesion Recovery
          hot_cold_stable_q.ix['2013':'2015'].plot(ax=axes[1][0],title='Growth Period')
          hot_cold_stable_q.ix['2016':].plot(ax=axes[1][1],title='Recent Growth')

Out[365]: <matplotlib.axes._subplots.AxesSubplot at 0x26a92600198>
```



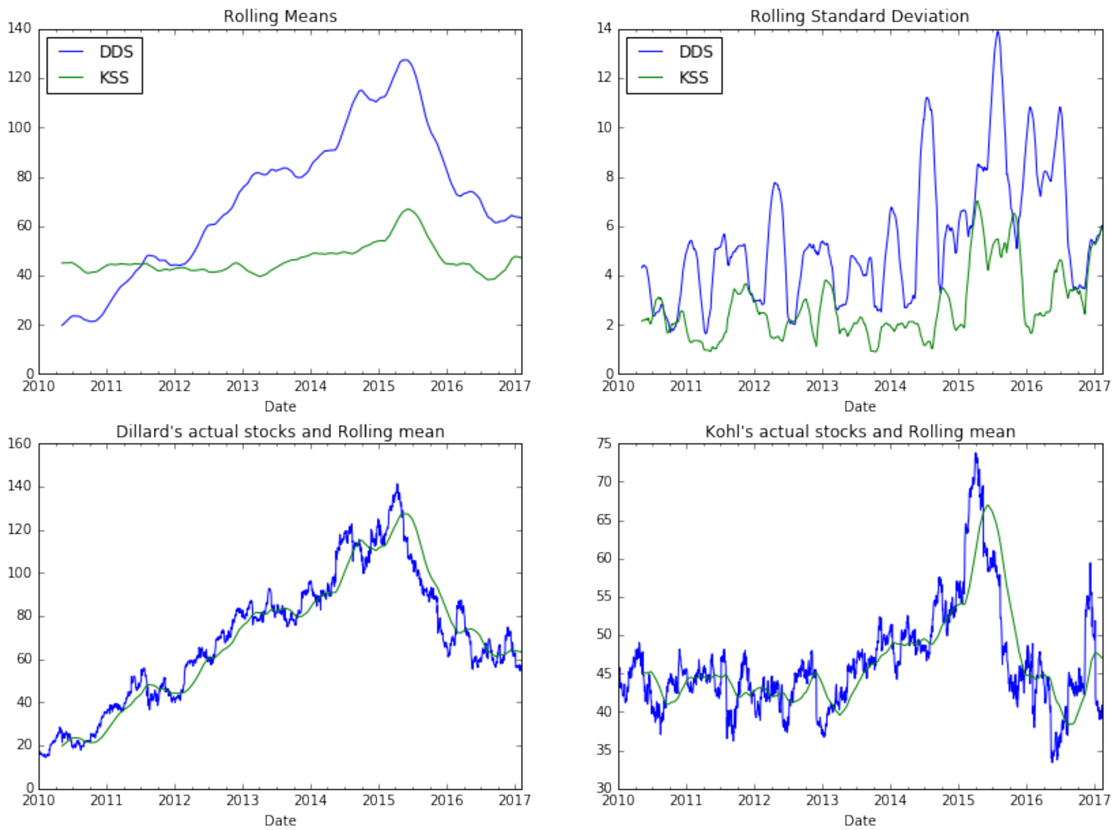
From the quarterly stock chart as shown above, it is evident that JCPenny (JCP) has some tough time to recover back to the original stock price reported between USD 50 and USD 60 before recession in 2016. So I can consider it as a cold stock. On the other hand, Dillards (DDS) has been growing tremendously recovering from recession period and moving forward. DDS was between USD 20 and USD 30 before recession and it has been closed between USD 60 and USD 70 in the last quarter of 2016 with the hisghest stock reaching between USD 80 and USD 90 in first quarter of 2016. Last but not least, Kohl's (KSS) has been stable through out the decase. It was the least impacted stock during the recession period. It has also closed between USD 40 and USD 50 in last quarter of 2016 and this was the range of stock price for Kohl's in fist quarter of 2006.

Next, I will compare between Kohl's(KSS) and Dillard's(DDS) to evaluate which one would be the actual hot riding stock by analyzing rolling mean and standard deviation trends for post recession period.

```
In [379]: # Computing and plotting rolling means and standard deviations over 90 day window
          period = 90
          fig, axes = plt.subplots(nrows=2,ncols=2,sharex=False,sharey=False,figsize=(14,10))
          post_recession_hot_px_mean = adj_close_px['2010':][['DDS','KSS']].rolling(window=per
          post_recession_hot_px_sd = adj_close_px['2010':][['DDS','KSS']].rolling(window=perio
          post_recession_hot_px_mean.plot(ax=axes[0][0], title = 'Rolling Means')
          post_recession_hot_px_sd.plot(ax=axes[0][1], title = 'Rolling Standard Deviation')
          adj_close_px['2010':].DDS.plot(ax=axes[1][0])
          post_recession_hot_px_mean.DDS.plot(ax=axes[1][0])
          adj_close_px['2010':].KSS.plot(ax=axes[1][1])
```

```
post_recession_hot_px_mean.KSS.plot(ax=axes[1][1])
axes[1][0].set_title('Dillard\'s actual stocks and Rolling mean')
axes[1][1].set_title('Kohl\'s actual stocks and Rolling mean')
```

Out[379]: <matplotlib.text.Text at 0x26a96214c18>



From the charts listed above for Dillard's (DDS) and Kohl's(KSS), it is evident that Dillard's DDS is the hot rising stock until 2016. Starting 2016, both Dillard's and Kohl's have been struggling to maintain the stock, along with other retailers Macy's (M) and Nordstorm (JWN) that has been covered in the report ealier. So, there might be other factors involved starting 2015.

0.3 Further Analysis of Rising Hot stock

For the continuation of this report I will take Dillard's DDS as a hot rising stock and cover the report for time period from 2010 to 2014.

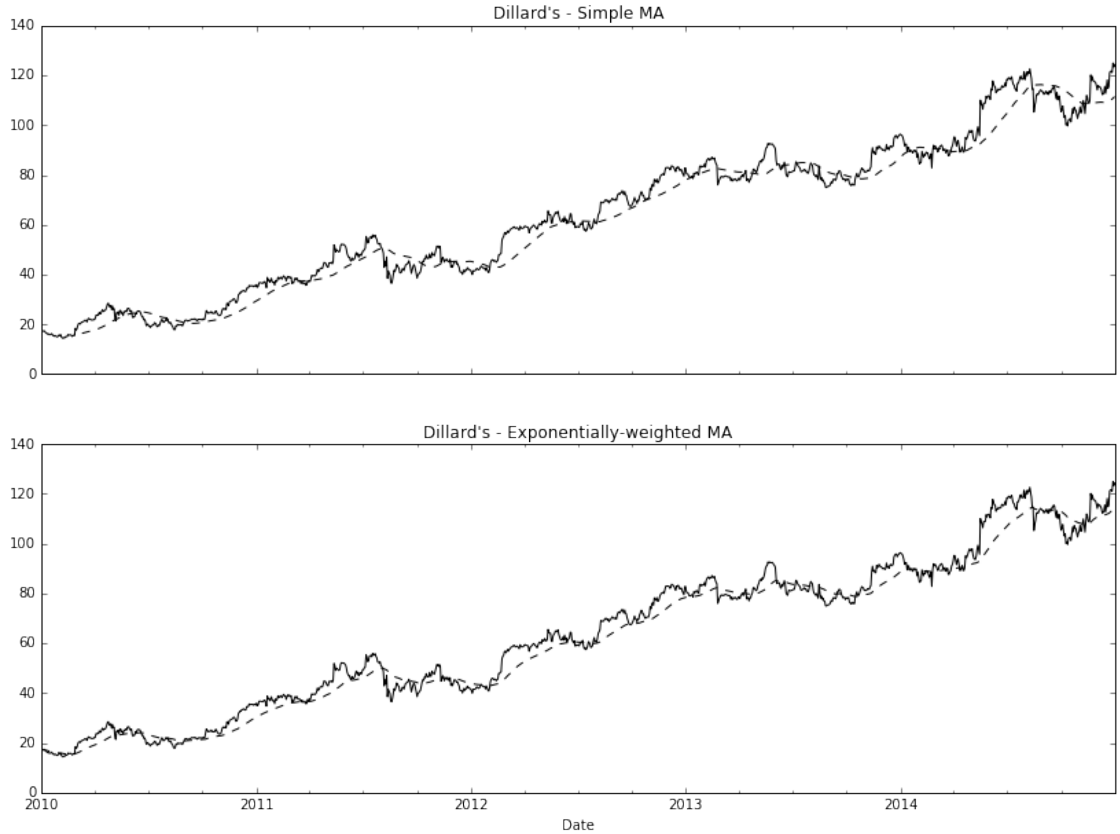
In [380]:
```
expanding_mean=lambda x : rolling_mean(x,len(x),min_periods=1)
```

In [386]:
```
# calucating and plotting rolling mean average and exponentially weighted mean avera
fig, axes = plt.subplots(nrows=2,ncols=1,sharex=True,sharey=True,figsize=(14,10))
dds_px = adj_close_px.DDS['2010':'2014']
ma60=pd.rolling_mean(dds_px,60,min_periods=50)
```

```
ewma60=pd.ewma(dds_px,span=60)
dds_px.plot(style='k-', ax=axes[0])
ma60.plot(style='k--',ax=axes[0])
dds_px.plot(style='k-',ax=axes[1])
ewma60.plot(style='k--',ax=axes[1])
axes[0].set_title('Dillard\'s - Simple MA')
axes[1].set_title('Dillard\'s - Exponentially-weighted MA')
```

```
C:\Anaconda3\lib\site-packages\ipykernel\__main__.py:4: FutureWarning: pd.rolling_mean is depre
        Series.rolling(min_periods=50,window=60,center=False).mean()
C:\Anaconda3\lib\site-packages\ipykernel\__main__.py:5: FutureWarning: pd.ewm_mean is deprecate
        Series.ewm(span=60,ignore_na=False,min_periods=0,adjust=True).mean()
```

```
Out[386]: <matplotlib.text.Text at 0x26a8d3a8f60>
```



From the charts list above, it is evident that Dillard's stock has an steady growth for the time period starting at 2010 and ending in the last quarter of 2014.

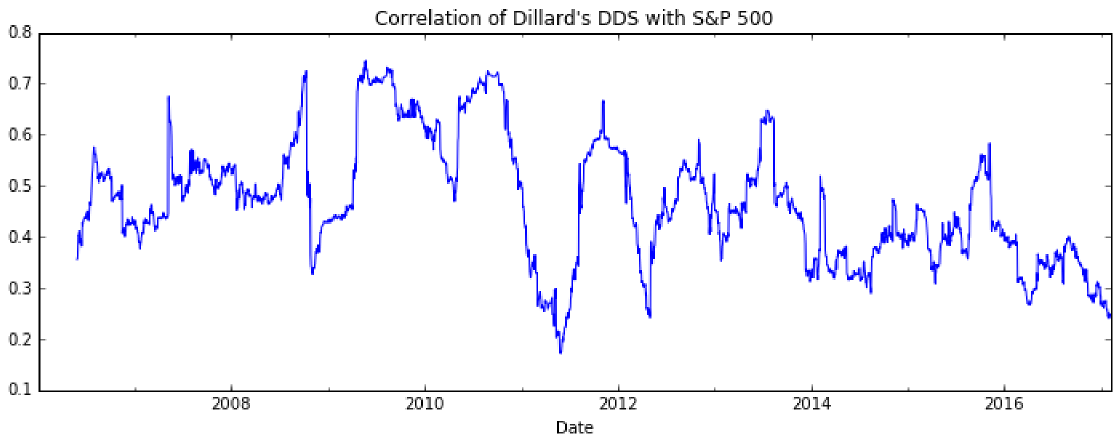
Analyze the coorelation of Retail stocks for Dillards with S&P 500.

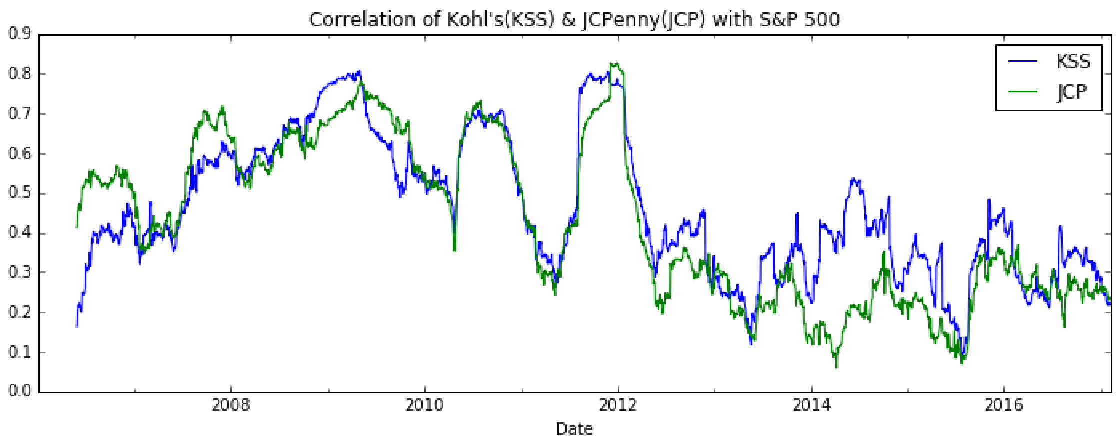
```
In [487]: #Computing and plotting correlation matrix with S&P 500 for selected retail stocks a
          spy_px=stocks['Adj Close'][S_and_P].SPY
```

```
spy_rets=spx_px/spx_px.shift(1)-1
returns=adj_close_px.pct_change()
corr=pd.rolling_corr(returns[['DDS','KSS','JCP']],spx_rets,125,min_periods=100)
corr.DDS.plot(figsize=(12,4), title='Correlation of Dillard\'s DDS with S&P 500')
corr[['KSS','JCP']].plot(figsize=(12,4), title='Correlation of Kohl\'s(KSS) & JCPenny
```

```
C:\Anaconda3\lib\site-packages\ipykernel\__main__.py:5: FutureWarning: pd.rolling_corr is depre
        DataFrame.rolling(min_periods=100,window=125).corr(other=<Series>)
```

```
Out[487]: <matplotlib.axes._subplots.AxesSubplot at 0x26aa5226eb8>
```



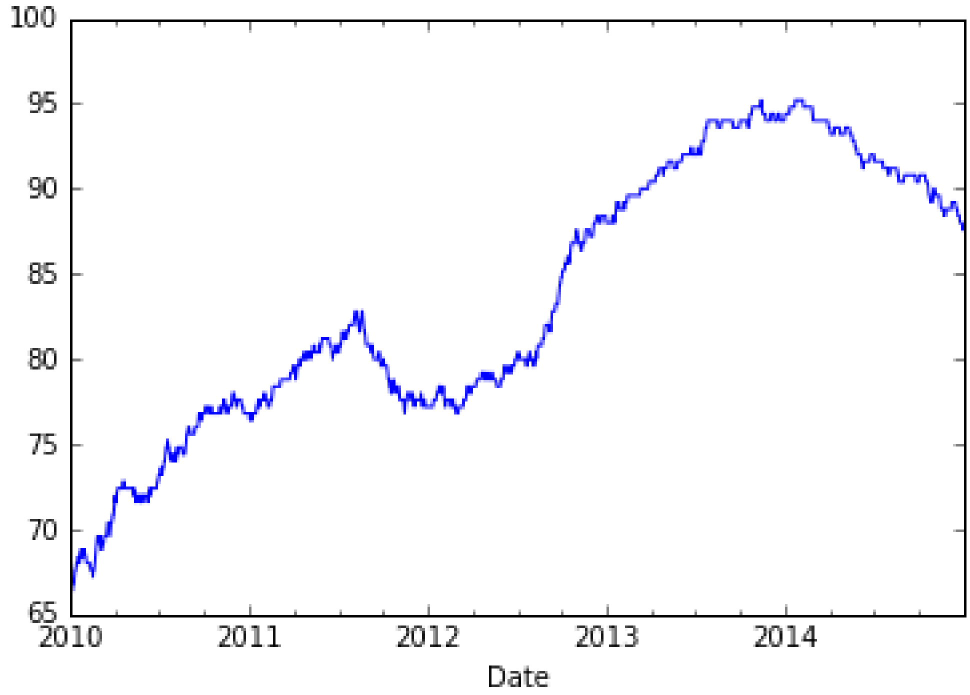


Analyzing Dillard's stock with custom defined function for rolling value of percentile over 90 day period

```
In [488]: #Define a custom function for calculating rolling value of percentile over 250 day p
          score_at_2percent= lambda x: percentileofscore(x,0.02)
          result=pd.rolling_apply(returns.DDS,250,score_at_2percent)
          result['2010':'2014'].plot()
```

```
C:\Anaconda3\lib\site-packages\ipykernel\__main__.py:3: FutureWarning: pd.rolling_apply is dep
        Series.rolling(center=False,window=250).apply(args=<tuple>,func=<function>,kwargs=<dic
  app.launch_new_instance()
```

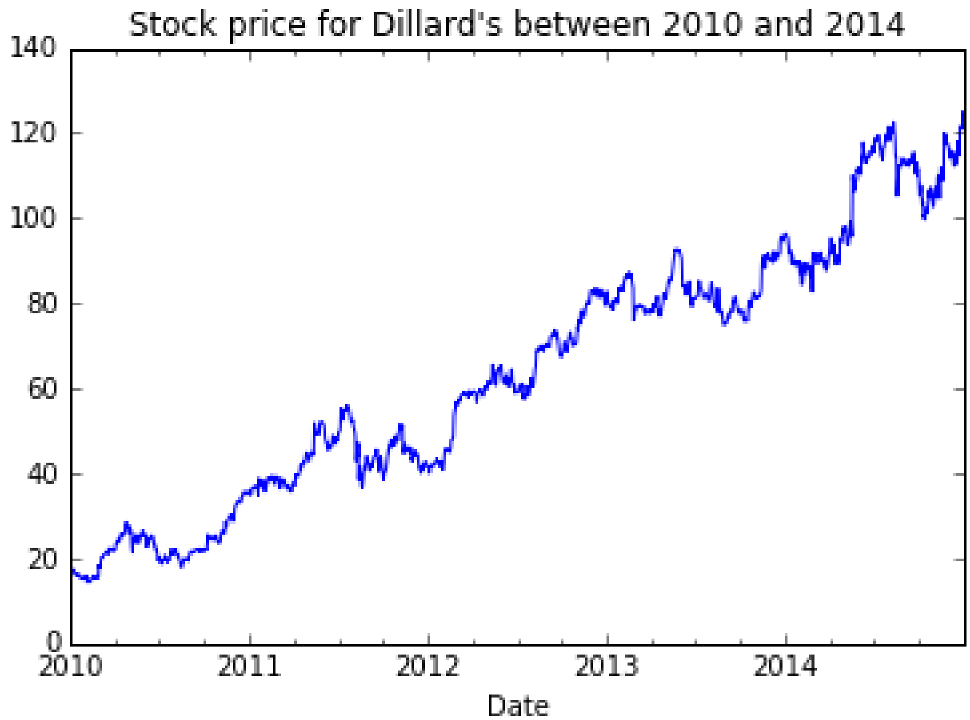
Out[488]: <matplotlib.axes._subplots.AxesSubplot at 0x26aa544c860>



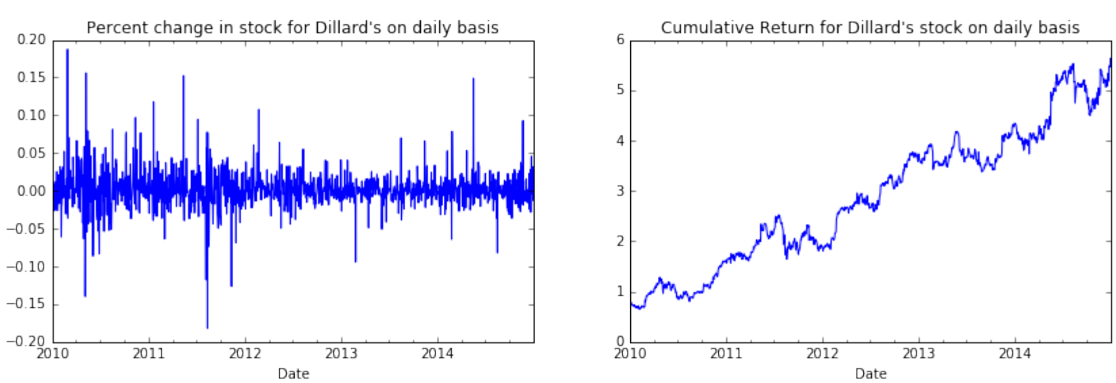
1 Sharpe Ratio Calculation

In [489]:
```
#peparing data for Dillard's (DDS) for sharpe ratio calculation
dds_price=adj_close_px.DDS
#capture data trend where it was most stable
dds_price['2010':'2014'].plot(title='Stock price for Dillard\'s between 2010 and 201
```

Out[489]: <matplotlib.axes._subplots.AxesSubplot at 0x26aa55094e0>



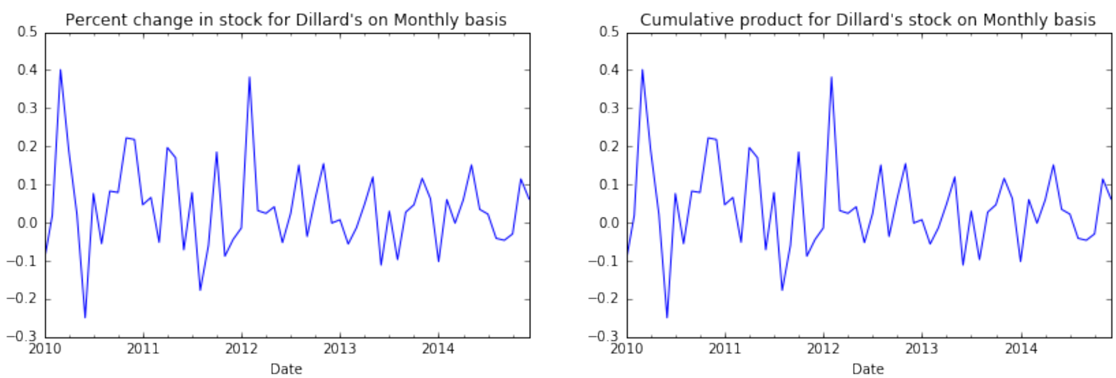

```
In [449]: dds_price['2014-10-01']/dds_price['2014-06-04']-1

Out[449]: -0.079865705598271663

In [500]: #Acquiring percent change per day and cumulative returns on compound product
          returns = dds_price.pct_change()
          ret_index = (1+returns).cumprod()
          ret_index[0]=1
          fig, axes = plt.subplots(nrows=1,ncols=2,sharex=False,sharey=False,figsize=(14,4))
          returns['2010':'2014'].plot(ax=axes[0],title='Percent change in stock for Dillard\'s
          ret_index['2010':'2014'].plot(ax=axes[1], title='Cumulative Return for Dillard\'s sto

Out[500]: <matplotlib.axes._subplots.AxesSubplot at 0x26aa6b6b860>
```



From the chart list above for percent change in stock and cumulative product for Dillard's on daily basis, as the percante change between 2012 and 2014 was mostly between 0.05, complementing with the steady growth in stock pricess during the same time period.

```
In [491]: #Acquiring percent change per month and product
          m_returns = ret_index.resample('M').last().pct_change()
          m_rets = (1 + returns).resample('M',kind='period').prod()-1
          fig, axes = plt.subplots(nrows=1,ncols=2,sharex=False,sharey=False,figsize=(14,4))
          m_returns['2010':'2014'].plot(ax=axes[0],title='Percent change in stock for Dillard\
          m_rets['2010':'2014'].plot(ax=axes[1], title='Cumulative product for Dillard\'s stocl
```

```
Out[491]: <matplotlib.axes._subplots.AxesSubplot at 0x26aa56c56d8>
```



From above charts, the monthly percent change and product being stable mostly between 0.1 and 0.2

First define couple of methods to compute daily returns and lagged moving sum as reference from the book 'Python for Data Analysis' by Wes Mckinney

```
In [495]: # method to calculate daily returns
          def to_index(rets):
              index = (1+rets).cumprod()
              first_loc = max(index.index.get_loc(index.idxmax())-1,0)
              index.values[first_loc]=1
              return index
```

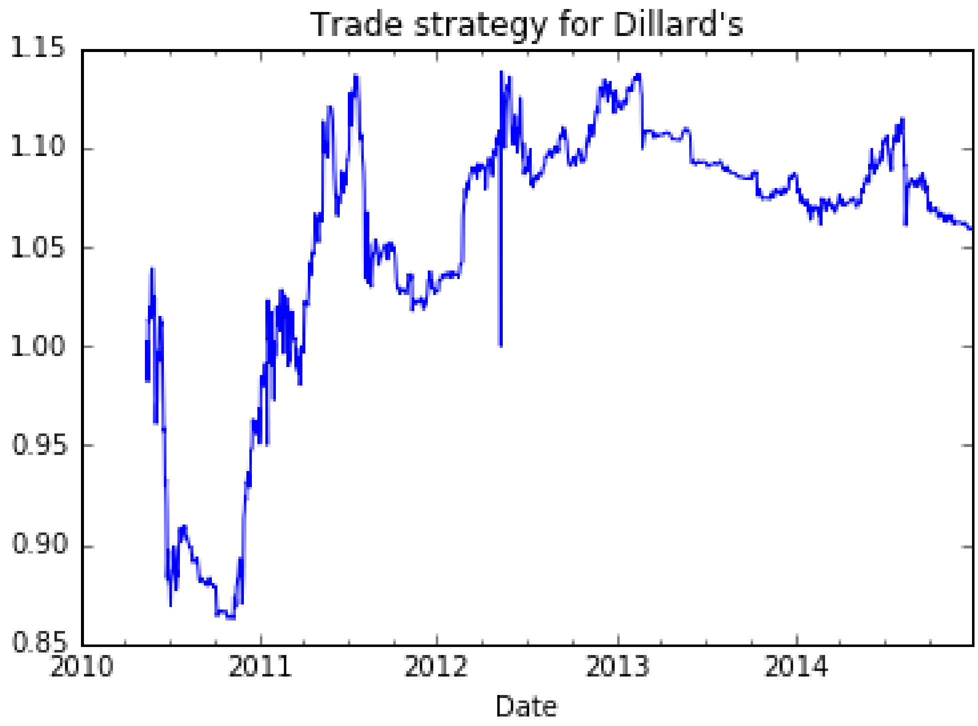
```
In [496]: # method to compute lagged moving sum
          def trend_signal(rets, lookback, lag):
              signal = pd.rolling_sum(rets, lookback, min_periods=lookback - 5)
              return signal.shift(lag)
```

Next, I have calculated the Trade strategy for Dillard's for the time period of 2010 to 2014. The reference has been taken form book 'Python for Data Analysis' by Wes Mckinney

```
In [529]: #calculate and plot trade strategy for business day
          signal = trend_signal(returns['2010':'2014'], 100, 3)
          trade = signal.resample('B').ffill()
          trade_rets =  trade.shift(1)*returns['2010':'2014']
          trade_rets = trade_rets[:len(returns['2010':'2014'])]
          to_index(trade_rets).plot(title='Trade strategy for Dillard\'s')
```

```
C:\Anaconda3\lib\site-packages\ipykernel\__main__.py:2: FutureWarning: pd.rolling_sum is depre
        Series.rolling(min_periods=95,window=100,center=False).sum()
  from ipykernel import kernelapp as app
```

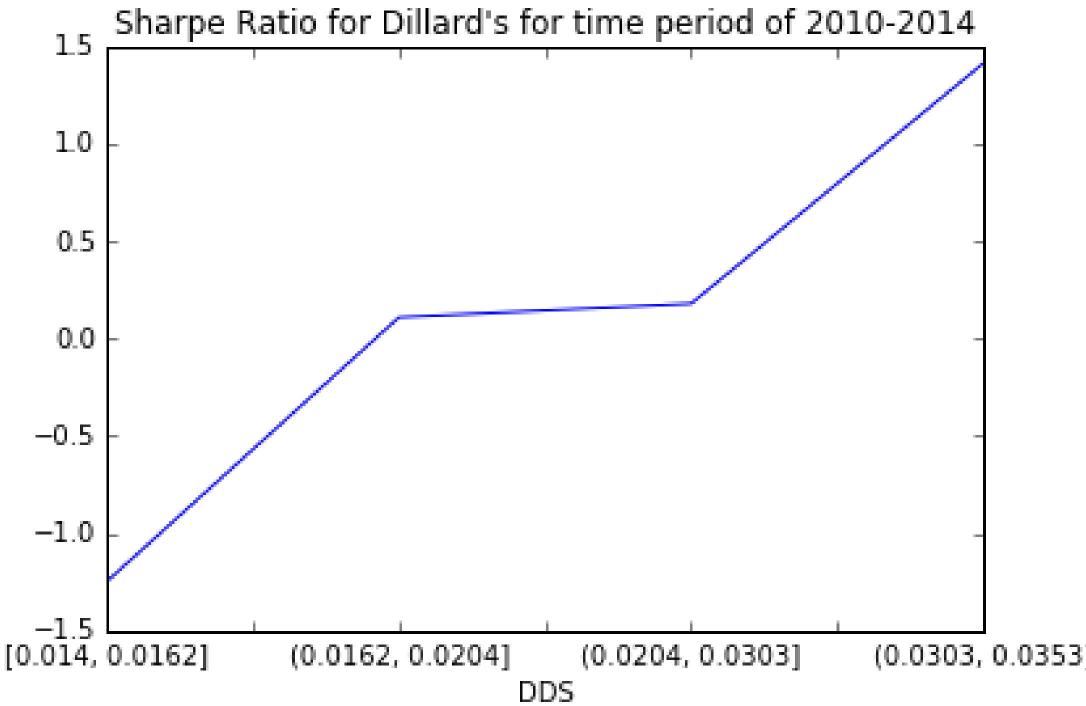
```
Out[529]: <matplotlib.axes._subplots.AxesSubplot at 0x26aa99007b8>
```



Following from trade strategy, now Sharpe ratio can be calculated as following. The reference has been taken form book 'Python for Data Analysis' by Wes Mckinney

```
In [532]: # Calculating Shape ratio
          vol = returns['2010':'2014'].rolling(250, min_periods=200).std()

          def sharpe(rets, ann=250):
              return rets.mean()/rets.std()*np.sqrt(ann)

          trade_rets.groupby(pd.qcut(vol,4)).agg(sharpe).plot(title='Sharpe Ratio for Dillard\
          for time period of 2010-2014')
```

Out[532]: <matplotlib.axes._subplots.AxesSubplot at 0x26aa99c10f0>



1.1 Signal Frontier Analysis

Next I have used Signal Frontier Analysis. The reference has been taken from The reference has been taken form book 'Python for Data Analysis' by Wes Mckinney

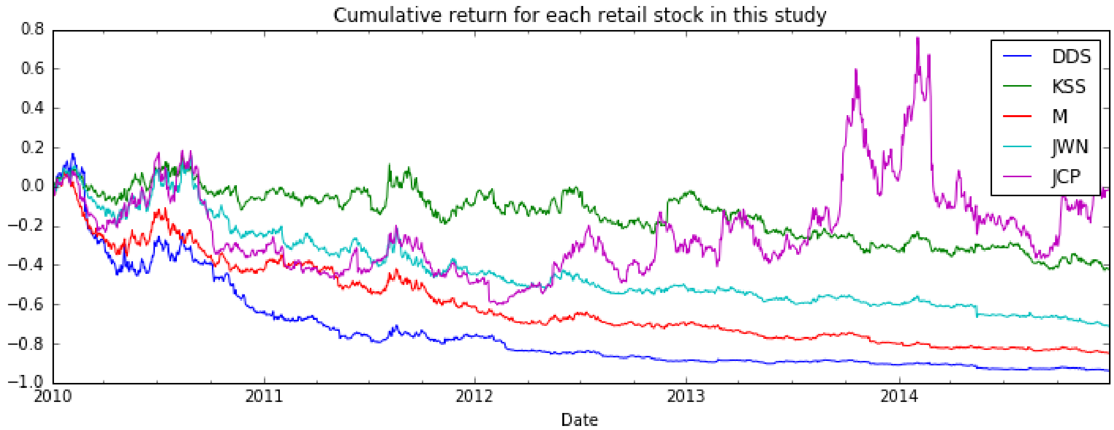
In [598]:
```
# prepare data for business day frequency for the years from 2010 to 2014
px = adj_close_px['2010':'2014'].asfreq('B').fillna(method='pad')
px.head()
```

Out[598]:

	DDS	KSS	M	JWN	JCP
Date					
2010-01-01	16.980893	45.134525	14.445024	30.121338	24.908686
2010-01-04	17.201782	45.176370	14.703586	30.393854	25.423521
2010-01-05	17.303022	45.209847	14.531212	29.920956	25.442243
2010-01-06	17.459486	45.795684	14.738062	30.001108	26.045765
2010-01-07	17.026911	44.741176	15.074192	31.203396	25.281931

In [599]:
```
#calculate return and cumulative return
rets = px.pct_change()
ret_index = (1-rets).cumprod()-1
ret_index.plot(figsize=(12,4), title='Cumulative return for each retail stock in thi
```

Out[599]: <matplotlib.axes._subplots.AxesSubplot at 0x26aaaf7cd30>



First I am going to setup the helper functions and calculation for calculating Sharpe Ratio using Frontier Signal Analysis.

In [646]:
```
# method for computing mean reversion over a given lookback
def calc_mom(price, lookback, lag):
    # case when prices  has sent as series
    if type(price) is pd.core.series.Series:
        mom_ret = price.shift(lag).pct_change(lookback)
        ranks = mom_ret.rank(ascending=False)
        demeanded = ranks.subtract(ranks.mean(),axis=0)
        return demeanded.divide(demeanded.std(),axis=0)
    else: # case when prices has sent as dataframe
        mom_ret = price.shift(lag).pct_change(lookback)
        ranks = mom_ret.rank(axis=1, ascending=False)
        demeanded = ranks.subtract(ranks.mean(axis=1),axis=0)
        return demeanded.divide(demeanded.std(axis=1),axis=0)
```

In [647]:
```
#Custom function for returning compound product
compound = lambda x:(1+x).prod()-1
#Customer function for daily sharpe ratio
daily_sr = lambda x: x.mean()/x.std()
```

In [648]:
```
# method for setting strategy for Sharpe Ratio
def strat_sr(prices, lb, hold):
    #Compute portfolio weights
    freq = '%dB' % hold
    port = calc_mom(prices, lb, lag=1)
    daily_rets = prices.pct_change()

    #compute portfolio returns
    port = port.shift(1).resample(freq).first()
    returns = daily_rets.resample(freq, how=compound)
    # case when prices  has sent as series
```

```
        if type(prices) is pd.core.series.Series:
            port_rets = (port*returns).sum()
        else: # case when prices has sent as dataframe
            port_rets = (port*returns).sum(axis=1)
        # case when prices  has sent as series
        if type(prices) is pd.core.series.Series:
            return port_rets*np.sqrt(252/hold)
        else: # case when prices has sent as dataframe
            return daily_sr(port_rets)*np.sqrt(252/hold)
```

```
In [649]: #testing the custom functions
          strat_sr(px, 70, 30)
```

```
C:\Anaconda3\lib\site-packages\ipykernel\__main__.py:10: FutureWarning: how in .resample() is d
the new syntax is .resample(...)..apply(<func>)
```

```
Out[649]: 0.027758055059689798
```

```
In [650]: def calculateAndStoreSharpeRatio(prices, lookbacks,holdings):
              dd = defaultdict(dict)
              for lb in lookbacks:
                  for hold in holdings:
                      dd[lb][hold]=strat_sr(prices,lb,hold)
              return dd
```

```
In [662]: # Generate a heat map for Shape ratio being calcualted over different lookback and ho
          def heatmap(df, title_str, cmap=plt.cm.Spectral_r):
              fig = plt.figure()
              ax=fig.add_subplot(111)
              axim=ax.imshow(df.values,aspect='auto', cmap=cmap,interpolation='nearest')
              ax.set_xlabel(df.columns.name)
              ax.set_xticks(np.arange(len(df.columns)))
              ax.set_xticklabels(list(df.columns))
              ax.set_yticks(np.arange(len(df.index)))
              ax.set_yticklabels(list(df.index))
              plt.colorbar(axim)
              plt.title(title_str)
```

```
In [660]: # Now calculate sharpe ratio for given lookback and holdings period between 2010 and
          lookbacks = range(20, 90, 5)
          holdings = range(20, 90, 5)
          #calcuate Sharpe Ratio for all the stocks
          dd_all = calculateAndStoreSharpeRatio(px, lookbacks,holdings)
          ddf_all = pd.DataFrame(dd_all)
          ddf_all.index.name = 'Holding Period'
          ddf_all.columns.name = 'Lookback Period'

          #Calculate Sharpe Ratio for Dillard's stock only
```

```
dd_dds = calculateAndStoreSharpeRatio(px.DDS, lookbacks,holdings)
ddf_dds = pd.DataFrame(dd_dds)
ddf_dds.index.name = 'Holding Period'
ddf_dds.columns.name = 'Lookback Period'
```

```
C:\Anaconda3\lib\site-packages\ipykernel\__main__.py:10: FutureWarning: how in .resample() is c
the new syntax is .resample(...)..apply(<func>)
```

```
In [668]: title_str_all='Heatmap of mean reversion Sharpe Ratio for All Retailers\n invoved in
           for time period 2010 - 2014'
          heatmap(ddf_all, title_str_all)
          title_str_dds='Heatmap of mean reversion Sharpe Ratio\n for Dillard\'s invoved in thi
           for time period 2010 - 2014'
          heatmap(ddf_dds, title_str_dds)
```