Strategy Results:

Final pairs obtained from the K—Means Clustering Technique

	Ticker	Nearest_Neighbor
0	RELIANCE	TCS
1	TCS	HDFCBANK
2	HDFCBANK	TCS
3	INFY	ITC
4	HINDUNILVR	NESTLEIND
...	...	...
491	FCONSUMER	MRPL
492	TVTODAY	BALRAMCHIN
493	CHENNPETRO	MRPL
494	OMAXE	ABFRL
495	HERITGFOOD	QUESS

Each pair is tested for Cointegration using ADF test on the spread for data from Jan-2018 to Dec-2018 and is tested for portfolio value Jan 2019-Dec2019

Only pairs showing a Cointegration of 1% will be considered for backtesting.

```
from statsmodels.tsa.api import adfuller
for i in range(496): #496
    stock1=final_pairs_df.loc[i]['Ticker']
    stock2=final_pairs_df.loc[i]['Nearest_Neighbor']
    print(stock1)
    print(stock2)
    stock1=price.loc[stock1]
    stock2=price.loc[stock2]
    stock1=stock1.fillna(0)
    stock2=stock2.fillna(0)
    stock1=stock1[:246]
    stock2=stock2[:246]
    #adding the spread column to the nemDF dataframe
    spread=stock1-stock2
    #instantiating the adfuller test
    adf=adfuller(spread)
    #Logic that states if our test statistic is less than
    #a specific critical value, then the pair is cointegrated at that
    #level, else the pair is not cointegrated
    if adf[0] < adf[4]['1%']:
        print('Spread is Cointegrated at 1% Significance Level')
    elif adf[0] < adf[4]['5%']:
        print('Spread is Cointegrated at 5% Significance Level')
    elif adf[0] < adf[4]['10%']:
        print('Spread is Cointegrated at 10% Significance Level')
    else:
        print('Spread is not Cointegrated')
```

```
RELIANCE
TCS
Spread is not Cointegrated
TCS
HDFCBANK
Spread is not Cointegrated
HDFCBANK
TCS
Spread is not Cointegrated
INFY
ITC
Spread is not Cointegrated
HINDUNILVR
NESTLEIND
Spread is not Cointegrated
HDFC
PFC
Spread is Cointegrated at 5% Significance Level
ICICIBANK
```

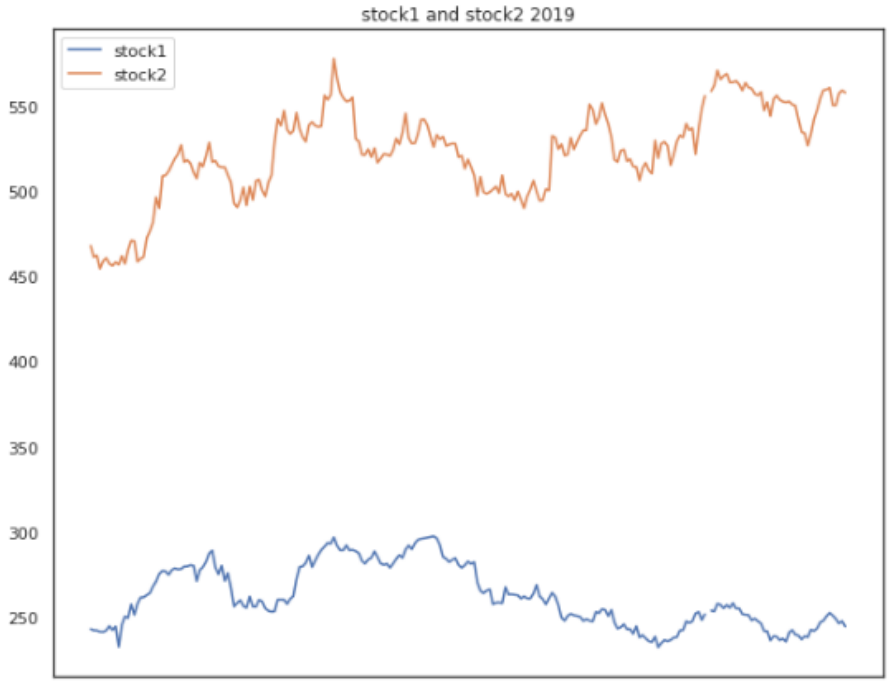
HCLTECH and WIPRO used for backtesting

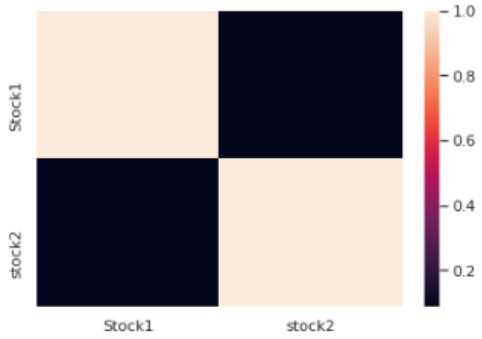
```
HCLTECH
WIPRO
Spread is Cointegrated at 1% Significance Level
```

Plotting price data and checking if future data is also statistically cointegrated

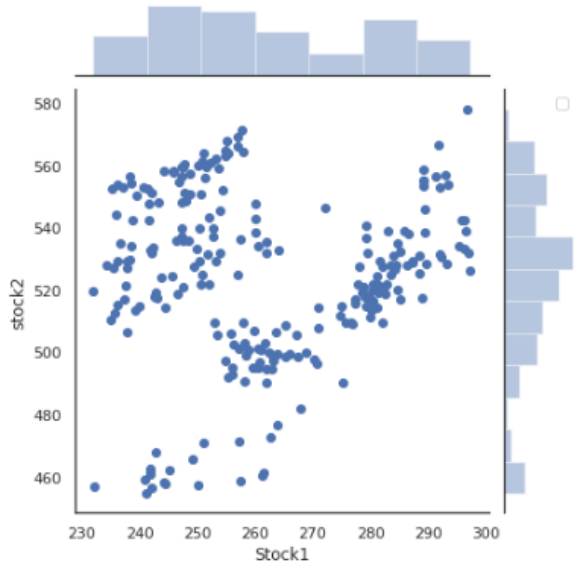
```
stock1=price.loc["WIPRO"]
stock2=price.loc["HCLTECH"]
stock1=stock1[-243:]
stock2=stock2[-243:]
#plotting
plt.figure(figsize=(10,8))
plt.plot(stock1, label='stock1')
plt.plot(stock2, label='stock2')
plt.legend(loc=0)
plt.title('stock1 and stock2 2019')
plt.show()
#Statistical Test
#importing necessary libraries
#data analysis/manipulation
import numpy as np
import pandas as pd
#importing pandas datareader to get our data
import pandas_datareader as pdr
#importing the Augmented Dickey Fuller Test to check for cointegration
from statsmodels.tsa.api import adfuller
#initializing newDF as a pandas dataframe
newDF=pd.DataFrame()
#adding WMT closing prices as a column to the newDF
newDF['Stock1']=stock1
#adding TGT closing prices as a column to the newDF
newDF['stock2']=stock2
newDF.head()
#using seaborn as sns to create a correlation heatmap of WMT and TGT
sns.heatmap(newDF.corr())
#Creating a scatter plot using Seaborn
plt.figure(figsize=(15,10))
sns.jointplot(newDF['Stock1'],newDF['stock2'])
plt.legend(loc=0)
plt.show()
newDF['Stock1']=newDF['Stock1'].fillna(0)
newDF['stock2']=newDF['stock2'].fillna(0)
#adding the spread column to the nemDF dataframe
newDF['Spread']=newDF['Stock1']-newDF['stock2']
#instantiating the adfuller test
adf=adfuller(newDF['Spread'])
#Logic that states if our test statistic is less than
#a specific critical value, then the pair is cointegrated at that
#level, else the pair is not cointegrated
if adf[0] < adf[4]['1%']:
    print('Spread is Cointegrated at 1% Significance Level')
elif adf[0] < adf[4]['5%']:
    print('Spread is Cointegrated at 5% Significance Level')
elif adf[0] < adf[4]['10%']:
    print('Spread is Cointegrated at 10% Significance Level')
else:
    print('Spread is not Cointegrated')
```

Price Data:






```
<Figure size 1080x720 with 0 Axes>
```




```
Spread is not Cointegrated
```

Creating and Executing Trade Signals:

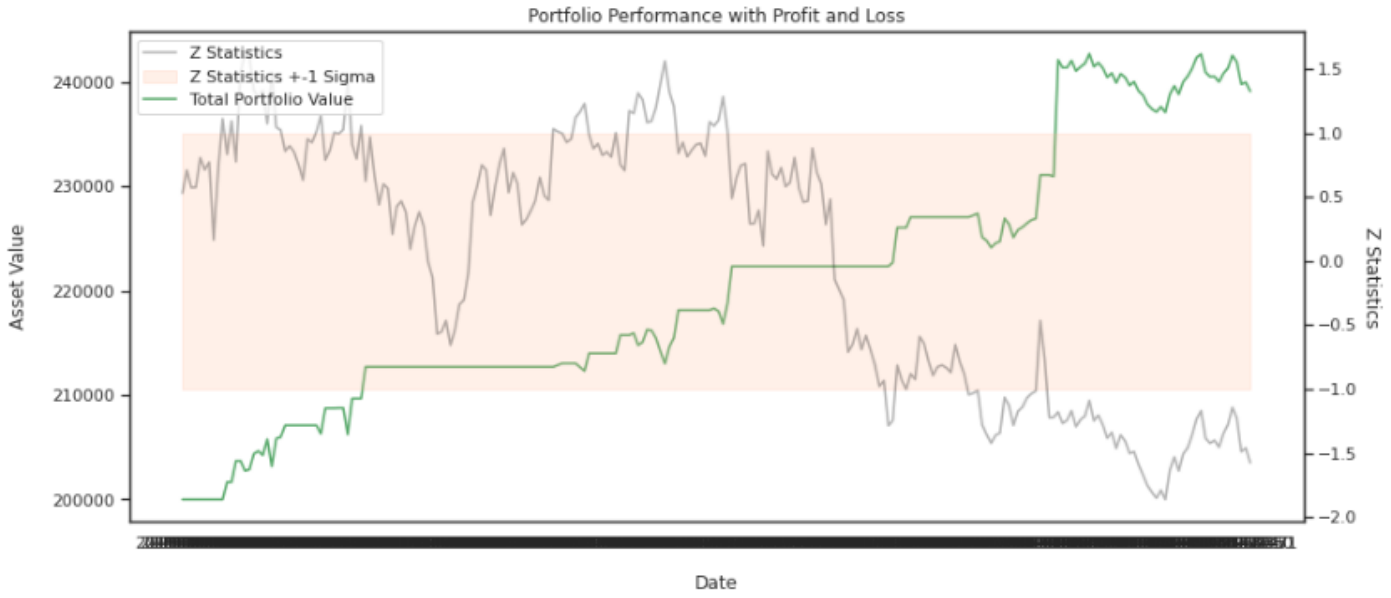
```
asset1="WIPRO"
asset2="HCLTECH"
# calculate z-score
def zscore(series):
    return (series.apply(lambda x: x - series.mean()))/ np.std(series)
# create a dataframe for trading signals
signals = pd.DataFrame()
signals['asset1'] = stock1
signals['asset2'] = stock2
ratios = signals.asset1 / signals.asset2
# calculate z-score and define upper and lower thresholds
signals['z'] = zscore(ratios)
signals['z upper limit'] = np.mean(signals['z']) + np.std(signals['z'])
signals['z lower limit'] = np.mean(signals['z']) - np.std(signals['z'])
# create signal - short if z-score is greater than upper limit else long
signals['signals1'] = 0
signals['signals1'] = np.select([signals['z'] > \
                                 signals['z upper limit'], signals['z'] < signals['z lower limit']], [-1, 1], defaul
# we take the first order difference to obtain portfolio position in that stock
signals['positions1'] = signals['signals1'].diff()
signals['signals2'] = -signals['signals1']
signals['positions2'] = signals['signals2'].diff()
# verify datafame head and tail
signals.head(3).append(signals.tail(3))
# visualize trading signals and position
fig=plt.figure(figsize=(14,6))
bx = fig.add_subplot(111)
bx2 = bx.twinx()
#plot two different assets
l1, = bx.plot(signals['asset1'], c='#4abdac')
l2, = bx2.plot(signals['asset2'], c='#907163')
u1, = bx.plot(signals['asset1'][signals['positions1'] == 1], lw=0, marker='^', markersize=8, c='g',alpha=0.7)
u2, = bx2.plot(signals['asset2'][signals['positions2'] == 1], lw=0, marker='^', markersize=8, c='r',alpha=0.7)
d1, = bx.plot(signals['asset1'][signals['positions1'] == -1], lw=0,marker='v',markersize=8, c='r',alpha=0.7)
u2, = bx2.plot(signals['asset2'][signals['positions2'] == 1], lw=0,marker=2,markersize=9, c='g',alpha=0.9, markeredg
d2, = bx2.plot(signals['asset2'][signals['positions2'] == -1], lw=0,marker=3,markersize=9, c='r',alpha=0.9,markeredg
bx.set_ylabel(asset1,)
bx2.set_ylabel(asset2, rotation=270)
bx.yaxis.labelpad=15
bx2.yaxis.labelpad=15
bx.set_xlabel('Date')
bx.xaxis.labelpad=15
plt.legend([l1,l2,u1,d1,u2,d2], [asset1, asset1,'LONG {}'.format(asset1),
           'SHORT {}'.format(asset1),
           'LONG {}'.format(asset2),
           'SHORT {}'.format(asset2)], loc ='best')
plt.title('Pair Trading')
plt.grid(True)

plt.tight_layout()
plt.show()
```

Executing Trades:

Strategy Analysis

```
initial_capital = 100000

# shares to buy for each position
positions1 = initial_capital// max(signals['asset1'])
positions2 = initial_capital// max(signals['asset2'])

# since there are two assets, we calculate each asset Pnl
# separately and in the end we aggregate them into one portfolio
portfolio = pd.DataFrame()
portfolio['asset1'] = signals['asset1']
portfolio['holdings1'] = signals['positions1'].cumsum() * signals['asset1'] * positions1
portfolio['cash1'] = initial_capital - (signals['positions1'] * signals['asset1'] * positions1).cumsum()
portfolio['total asset1'] = portfolio['holdings1'] + portfolio['cash1']
portfolio['return1'] = portfolio['total asset1'].pct_change()
portfolio['positions1'] = signals['positions1']
# pnl for the 2nd asset
portfolio['asset2'] = signals['asset2']
portfolio['holdings2'] = signals['positions2'].cumsum() * signals['asset2'] * positions2
portfolio['cash2'] = initial_capital - (signals['positions2'] * signals['asset2'] * positions2).cumsum()
portfolio['total asset2'] = portfolio['holdings2'] + portfolio['cash2']
portfolio['return2'] = portfolio['total asset2'].pct_change()
portfolio['positions2'] = signals['positions2']

# total pnl and z-score
portfolio['z'] = signals['z']
portfolio['total asset'] = portfolio['total asset1'] + portfolio['total asset2']
portfolio['z upper limit'] = signals['z upper limit']
portfolio['z lower limit'] = signals['z lower limit']
portfolio = portfolio.dropna()

# plot the asset value change of the portfolio and pnl along with z-score
fig = plt.figure(figsize=(14,6),)
ax = fig.add_subplot(111)
ax2 = ax.twinx()
l1, = ax.plot(portfolio['total asset'], c='g')
l2, = ax2.plot(portfolio['z'], c='black', alpha=0.3)
b = ax2.fill_between(portfolio.index,portfolio['z upper limit'],\
                portfolio['z lower limit'], \
                alpha=0.2,color='#ffb48f')
ax.set_ylabel('Asset Value')
ax2.set_ylabel('Z Statistics',rotation=270)
ax.yaxis.labelpad=15
ax2.yaxis.labelpad=15
ax.set_xlabel('Date')
ax.xaxis.labelpad=15
plt.title('Portfolio Performance with Profit and Loss')
plt.legend([l2,b,l1],['Z Statistics',
                      'Z Statistics +-1 Sigma',
                      'Total Portfolio Value'],loc='upper left');
plt.show()
# calculate CAGR
final_portfolio = portfolio['total asset'].iloc[-1]
delta = len(portfolio.index)-1
print('Number of days = ', delta)
YEAR_DAYS = 365
returns = (final_portfolio/initial_capital) ** (YEAR_DAYS/delta) - 1
print('CAGR = {:.3f}%' .format(returns * 100))
```

Portfolio Value over Period of Execution:




```
Number of days =  239
CAGR = 31.360%
```