US Stocks Behaviour During Financial Crisis 2008

2008's Financial Crisis Analysis.

We can see real life events ocurring in the Data Frame and Graphics.

This is a EDA - Exploratory Data Analysis (not include a ML - Machine Learning Model)

The Data:

Scraped data from web:

- Stock Info: stock prices (High, Low, Open, Close)
- Volume
- Date: datetime
- Bank Ticker: BAC, C, GS, JPM, MS, WFC

Goal / Target:

Explore to undertand facts and events that ocurrs during the crisis and their impact on stock prices.

Tools:

- Python
- Numpy, Pandas
- Matplotlib, Seaborn
- Plotly, Cufflinks
- Yahoo Finance

Step 1: Imports and Data Reading

We have to install it beforehand:

```
pip install pandas_datareader # to read data direct from source
pip install cufflinks # to see data with interactive graphic
pip install yfinance # to get data direct from web
```

Importing libraries:

```
from pandas_datareader import data as pdr
from pandas_datareader import wb

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly as px
import cufflinks as cf

import datetime

import yfinance as yf
```

Period of analysis:

```
start = '2006-1-1'
end = '2016-1-1'
```

Getting data direct from Yahoo Finance Data Base:

```
#Tickers from NYSE:

BAC = pdr.get_data_yahoo('BAC', start, end)

C = pdr.get_data_yahoo('C', start, end)

GS = pdr.get_data_yahoo('GS', start, end)

JPM = pdr.get_data_yahoo('JPM', start, end)

MS = pdr.get_data_yahoo('MS', start, end)

WFC = pdr.get_data_yahoo('WFC', start, end)
```

A sample looks like that:

```
BAC.head()
```

	High	Low	Open	Close	Volume	Adj Close
Date						
2006-01-03	47.180000	46.150002	46.919998	47.080002	16296700.0	33.170307
2006-01-04	47.240002	46.450001	47.000000	46.580002	17757900.0	32.818043
2006-01-05	46.830002	46.320000	46.580002	46.639999	14970700.0	32.860325
2006-01-06	46.910000	46.349998	46.799999	46.570000	12599800.0	32.811001
2006-01-09	46.970001	46.360001	46.720001	46.599998	15619400.0	32.832130

Creating data frame with all the stocks together:

```
df = pdr.get_data_yahoo(['BAC','C','GS','JPM','MS','WFC'], start, end)

tickers = ['BAC', 'C', 'GS', 'JPM', 'MS', 'WFC']

bank_stocks = pd.concat([BAC, C, GS, JPM, MS, WFC],axis=1,keys=tickers)

bank_stocks.head()
```

	BAC						C				...	MS		
	High	Low	Open	Close	Volume	Adj Close	High	Low	Open	Close	...	Open	Close	Volum
Date														
2006-01-03	47.180000	46.150002	46.919998	47.080002	16296700.0	33.170307	493.799988	481.100006	490.000000	492.899994	...	57.169998	58.310001	537700
2006-01-04	47.240002	46.450001	47.000000	46.580002	17757900.0	32.818043	491.000000	483.500000	488.600006	483.799988	...	58.700001	58.349998	797780
2006-01-05	46.830002	46.320000	46.580002	46.639999	14970700.0	32.860325	487.799988	484.000000	484.399994	486.200012	...	58.549999	58.509998	577800
2006-01-06	46.910000	46.349998	46.799999	46.570000	12599800.0	32.811001	489.000000	482.000000	488.799988	486.200012	...	58.770000	58.570000	688980
2006-01-09	46.970001	46.360001	46.720001	46.599998	15619400.0	32.832130	487.399994	483.000000	486.000000	483.899994	...	58.630001	59.189999	414450

5 rows × 36 columns

Info:

```
bank_stocks.info()

DatetimeIndex: 2517 entries, 2006-01-03 to 2015-12-31
Data columns (total 36 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   (BAC, High)      2517 non-null   float64
 1   (BAC, Low)       2517 non-null   float64
 2   (BAC, Open)      2517 non-null   float64
 3   (BAC, Close)     2517 non-null   float64
 4   (BAC, Volume)    2517 non-null   float64
 5   (BAC, Adj Close) 2517 non-null   float64
 6   (C, High)        2517 non-null   float64
 7   (C, Low)         2517 non-null   float64
 8   (C, Open)        2517 non-null   float64
 9   (C, Close)       2517 non-null   float64
 10  (C, Volume)      2517 non-null   float64
 11  (C, Adj Close)   2517 non-null   float64
 12  (GS, High)       2517 non-null   float64
 13  (GS, Low)        2517 non-null   float64
 14  (GS, Open)       2517 non-null   float64
 15  (GS, Close)      2517 non-null   float64
 16  (GS, Volume)     2517 non-null   float64
 17  (GS, Adj Close)  2517 non-null   float64
 18  (JPM, High)      2517 non-null   float64
 19  (JPM, Low)       2517 non-null   float64
...
 34  (WFC, Volume)    2517 non-null   float64
 35  (WFC, Adj Close) 2517 non-null   float64
dtypes: float64(36)
```

Setting columns level name:

```
bank_stocks.columns.names = ['Bank Ticker','Stock Info']
bank_stocks.head()
```

Bank Ticker	BAC						C	
Stock Info	High	Low	Open	Close	Volume	Adj Close	High	Low
Date								
2006-01-03	47.180000	46.150002	46.919998	47.080002	16296700.0	33.584057	493.799988	481.10
2006-01-04	47.240002	46.450001	47.000000	46.580002	17757900.0	33.227398	491.000000	483.50
2006-	46.830002	46.320000	46.580002	46.639999	14970700.0	33.270214	487.799988	484.00

Step 2: EDA - Exploratory Data Analysis

Historical maximuns at closing:

```
bank_stocks.xs(key='Close',axis=1,level='Stock Info').max() #  using cross
sections (.xs)

Bank Ticker
BAC       54.900002
C        564.099976
GS       247.919998
JPM       70.080002
MS        89.300003
WFC       58.520000
```

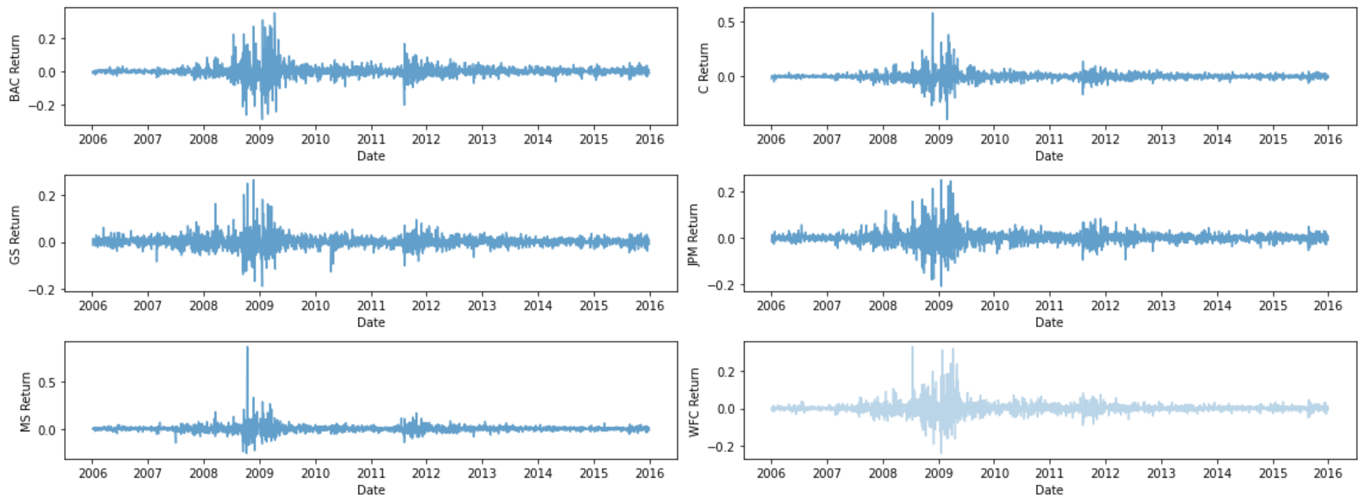
% of appreciation / depreciation:

```
returns = pd.DataFrame()

for tick in tickers:
    returns[tick+' Return'] = bank_stocks[tick]['Close'].pct_change() #
.pct_change() METHOD = percentage change in the time-series

returns.head()
```

	BAC Return	C Return	GS Return	JPM Return	MS Return	WFC Return
Date						
2006-01-03	NaN	NaN	NaN	NaN	NaN	NaN
2006-01-04	-0.010620	-0.018462	-0.013812	-0.014183	0.000686	-0.011599
2006-01-05	0.001288	0.004961	-0.000393	0.003029	0.002742	-0.001110
2006-01-06	-0.001501	0.000000	0.014169	0.007046	0.001025	0.005874
2006-01-09	0.000644	-0.004731	0.012030	0.016242	0.010586	-0.000158

In graph:

```
plt.figure(figsize=(16,6))

plt.subplot(321)
sns.lineplot(returns, x='Date', y='BAC Return', alpha=0.70)

plt.subplot(322)
sns.lineplot(returns, x='Date', y='C Return', alpha=0.70)

plt.subplot(323)
sns.lineplot(returns, x='Date', y='GS Return', alpha=0.70)

plt.subplot(324)
sns.lineplot(returns, x='Date', y='JPM Return', alpha=0.70)

plt.subplot(325)
sns.lineplot(returns, x='Date', y='MS Return', alpha=0.70)

plt.subplot(326)
sns.lineplot(returns, x='Date', y='WFC Return', alpha=0.30)

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

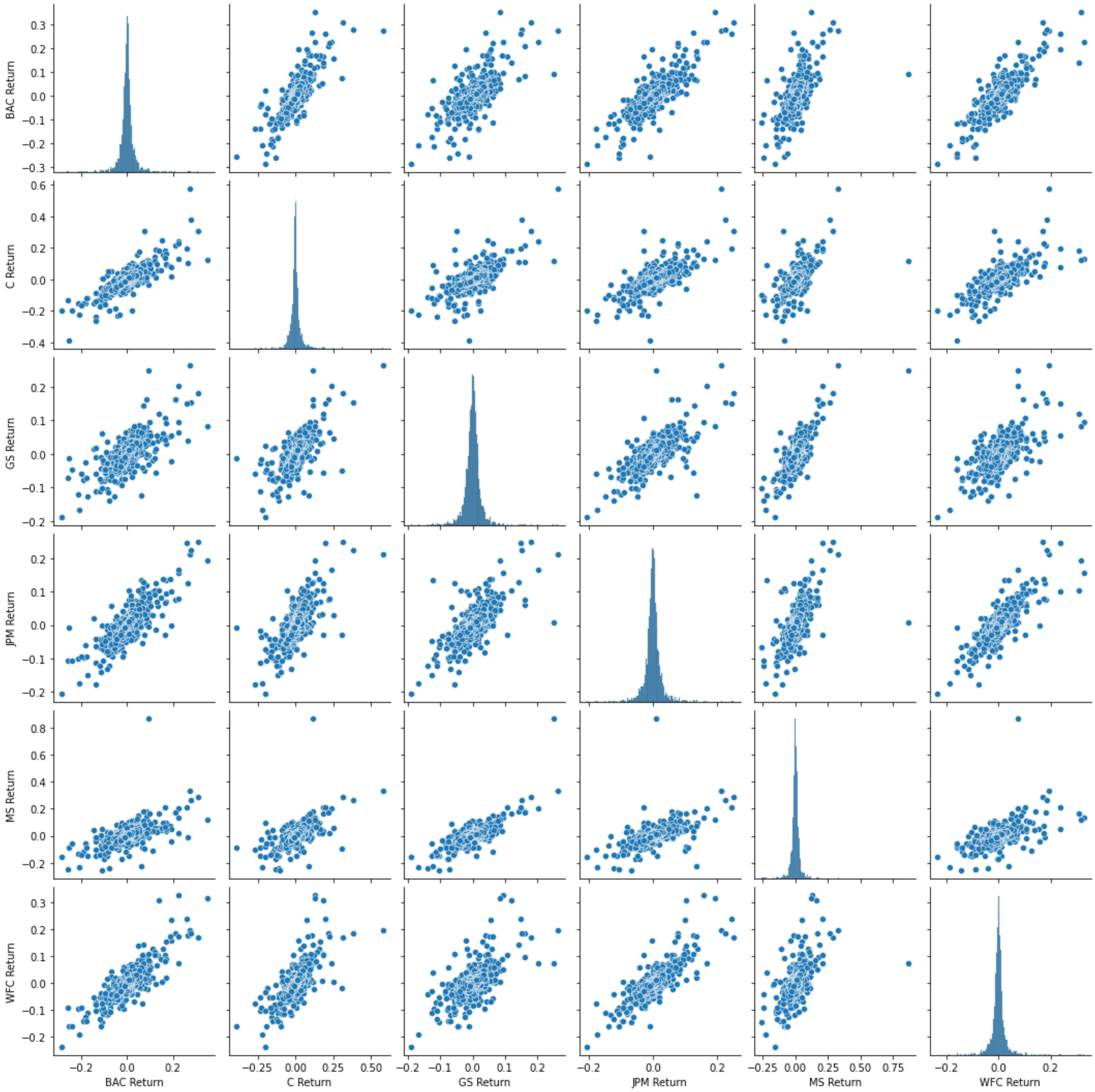


% of depreciation of MS and CITI hit a highest point than the others.

CITI's recovery looks faster than the others. Probably, this ocurrs because CITI was early split in two (restructuring of the company when the crisis started).

Comparing tickers:

```
sns.pairplot(returns[1:])
```



Rock bottom day:

```
returns.idxmin()

BAC Return    2009-01-20
C Return      2009-02-27
GS Return     2009-01-20
JPM Return    2009-01-20
MS Return     2008-10-09
WFC Return    2009-01-20
```

Observatiosn about the minimums:

- 2009-01-20: the market was concerned about the changing of the USA presidency this day.
- 2008-10-09: MS tumbled 25.9%
- 2009-02-27: CITI deal inspires no confidence.

Standard Deviation:

```
returns.std().sort_values(ascending=False)

C Return      0.038672
MS Return     0.037819
BAC Return    0.036647
WFC Return    0.030238
JPM Return    0.027667
GS Return     0.025390
```

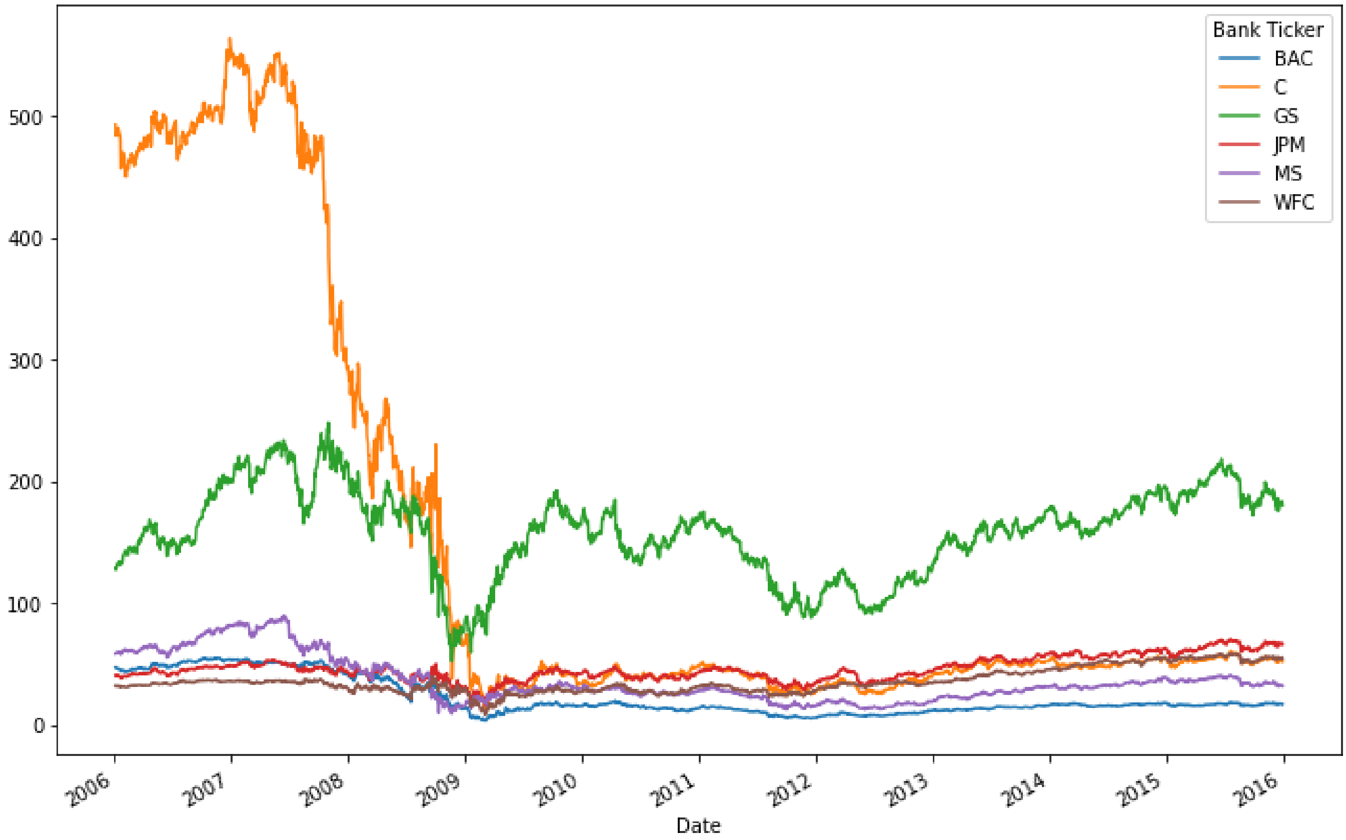
There was a bigger std for MS and CITI too (+ risk/volatility).

Highest risk/volatility in 2015 (after the crisis, last year of this DB):

```
returns['2015-01-01':'2015-12-31'].std().sort_values(ascending=False)

MS Return     0.016249
BAC Return    0.016163
C Return      0.015289
GS Return     0.014046
JPM Return    0.014017
WFC Return    0.012591
```

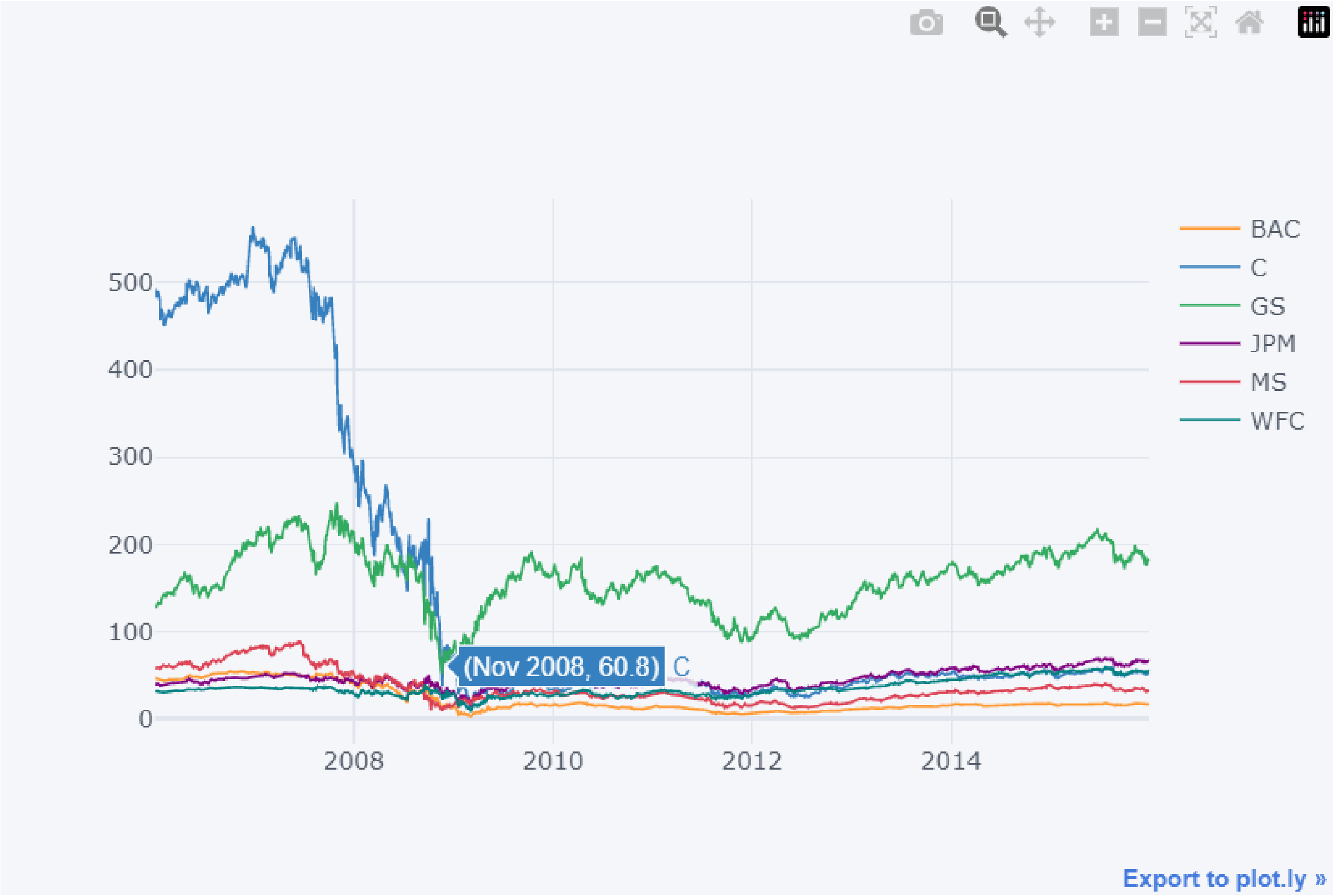
Closing prices for each bank/ticker for the entire period:

```
for tick in tickers:
    bank_stocks[tick]['Close'].plot(label=tick,figsize=(12,8))
```



Look at the same graph with an interactive plotly:

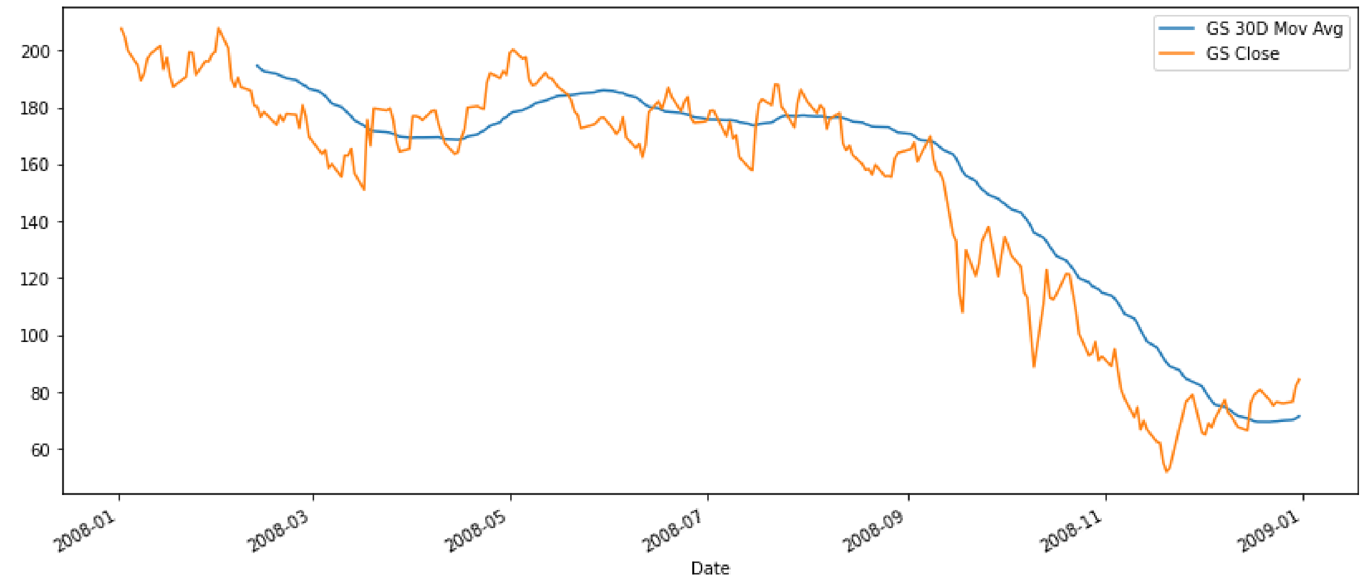
```
cf.go_offline() # needed to run on jupiter

--NotebookApp.iopub_data_rate_limit=10000000 # increasing the memory used by
jupiter

bank_stocks.xs(key='Close',axis=1,level='Stock Info').iplot()
```



GS recovery was due, in part, because of the purchaseof $5 billion in shares by Warren Buffet and the bailout from US goverment.

Looking close to GS. Moving averages vs. stock prices:

```
GS['Close']['2008-01-01':'2008-12-31'].rolling(window=30).mean().plot(label='GS 30D Mov Avg',figsize=(12,8))

GS['Close']['2008-01-01':'2008-12-31'].plot(label='GS Close',figsize=(14,6))
plt.legend()
```

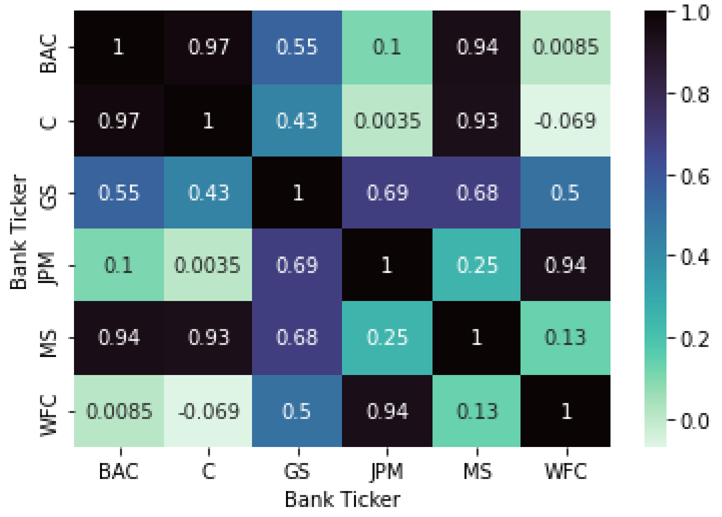


Simple mov. avg with plotly:

```
gs_sma = GS['Close']['2015-01-01':'2015-03-31'].ta_plot(study='sma',periods=
[14,28,60])
```

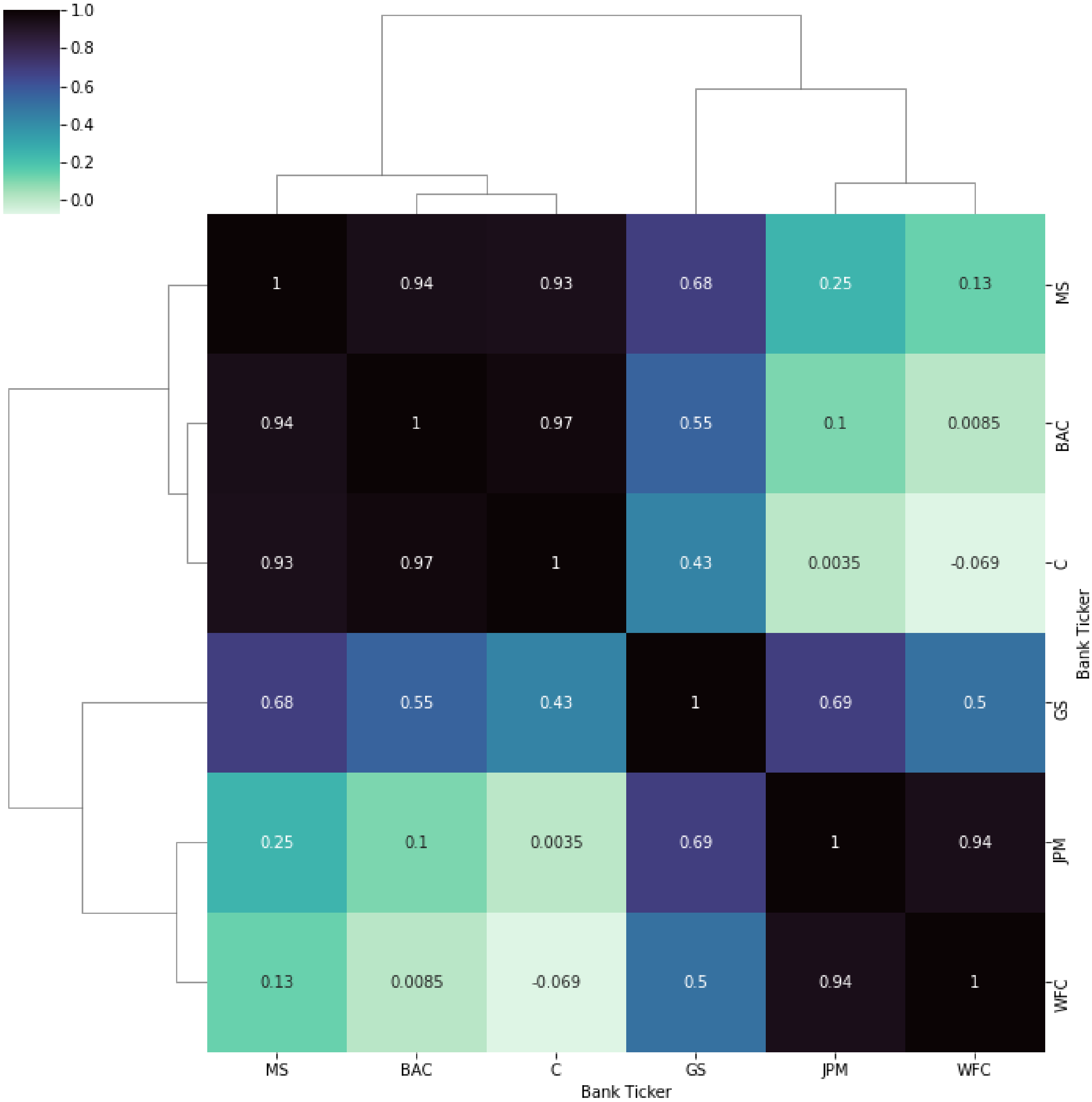
Correlation between stock prices:

```
sns.heatmap(bank_stocks.xs(key='Close', axis=1, level='Stock Info').corr(),
annot=True, cmap='mako_r')
```



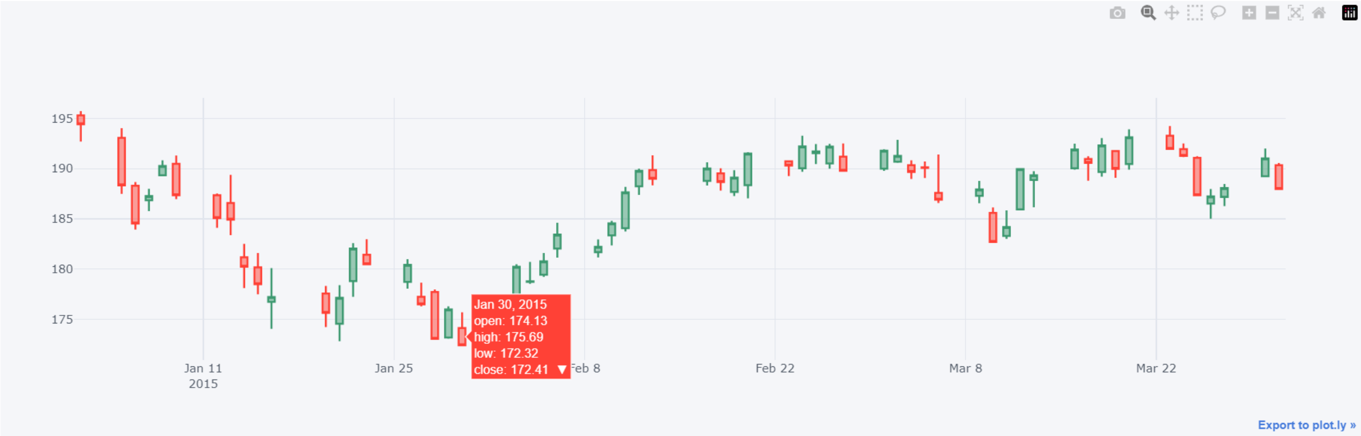
To see more clear:

```
sns.clustermap(bank_stocks.xs(key='Close', axis=1, level='Stock Info').corr(),
annot=True, cmap='mako_r')
```



Tipical interactive candle plot:

```
gs_candle = GS[['Open','High','Low','Close']]['2015-01-01':'2015-03-31']
gs_candle.iplot(kind='candle')
```



Technical Analysis plot:

```
gs_sma = GS['Close']['2015-01-01':'2015-03-31'].ta_plot(study='boll')
```