In [1]:
```python
import requests
from bs4 import BeautifulSoup
import pandas as pd
import yfinance as yf
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import warnings

# Ignore FutureWarnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

Defining Functions

In [3]:
```python
# Function to get stock data using yfinance
def get_stock_data(symbol):
    stock = yf.Ticker(symbol)
    data = stock.history(period="max")
    data.reset_index(inplace=True)
    return data
```

In [4]:
```python
# Function to convert revenue values to floats
def parse_revenue(revenue_str):
    revenue_str = revenue_str.replace('₹', '').strip()
    if 'B' in revenue_str:
        return float(revenue_str.replace('B', '')) * 1e9  # Convert from billions
    elif 'T' in revenue_str:
        return float(revenue_str.replace('T', '')) * 1e12  # Convert from trillions
    else:
        return float(revenue_str.replace(',', ''))  # Convert normally if no B or T
```

In [5]:
```python
# Function to scrape revenue data from a given URL
def get_revenue_data(url):
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')

    # Find the table and all rows
    table = soup.find('table', {'class': "table"})
    if table is None:
        print(f"Table not found for {url}!")
        return pd.DataFrame()

    rows = table.find_all('tr')

    revenue_data = []
    for row in rows:
        cols = row.find_all('td')

        # Check if there are at least 2 columns (Date and Revenue) ignoring the Change column
        if len(cols) >= 2:
            # Extract the year from the <span class="year"> tag
            year = cols[0].find('span', {'class': 'year'}).text if cols[0].find('span', {'class': 'year'}) else cols[0].text.strip()

            # Extract the revenue value and clean it up
            revenue = parse_revenue(cols[1].text)

            revenue_data.append((year, revenue))
```

```
    # Create DataFrame with columns Date and Revenue
    df = pd.DataFrame(revenue_data, columns=['Date', 'Revenue'])
    return df
```

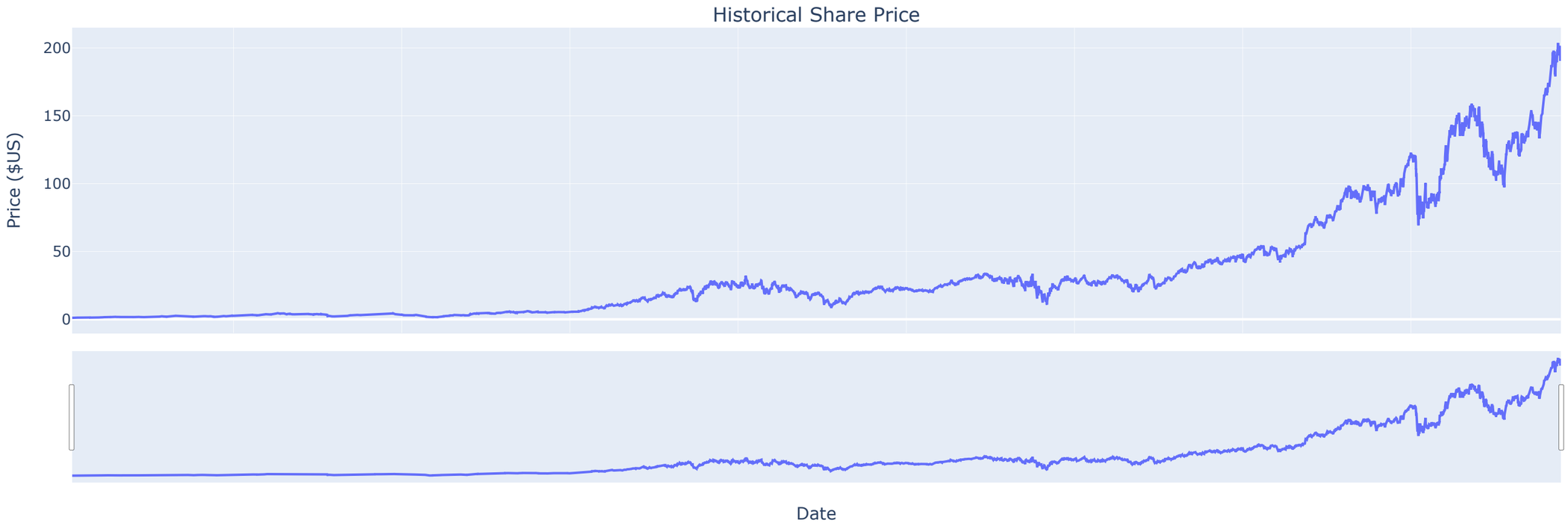
In [6]:
```
# Function to generate graphs for stock and revenue data
def make_graph(stock_data, revenue_data, stock):
    fig = make_subplots(rows=2, cols=1, shared_xaxes=True, subplot_titles=("Historical Share Price", "Historical Revenue"), vertical_spacing=0.3)

    stock_data_specific = stock_data[stock_data['Date'] <= '2024-06-14']
    revenue_data_specific = revenue_data[revenue_data['Date'] <= '2024-04-30']

    # Plotting share price
    fig.add_trace(go.Scatter(x=pd.to_datetime(stock_data_specific['Date']), y=stock_data_specific['Close'].astype("float"), name="Share Price"), row=1, col=1)

    # Plotting revenue
    fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific['Date']), y=revenue_data_specific['Revenue'].astype("float"), name="Revenue"), row=2, col=1)

    # Update layout
    fig.update_xaxes(title_text="Date", row=1, col=1)
    fig.update_xaxes(title_text="Date", row=2, col=1)
    fig.update_yaxes(title_text="Price ($US)", row=1, col=1)
    fig.update_yaxes(title_text="Revenue ($US)", row=2, col=1)
    fig.update_layout(showlegend=False, height=900, title=stock, xaxis_rangeslider_visible=True)

    # Show and save the graph
    fig.show()
    fig.write_html(f'{stock}.html')
```

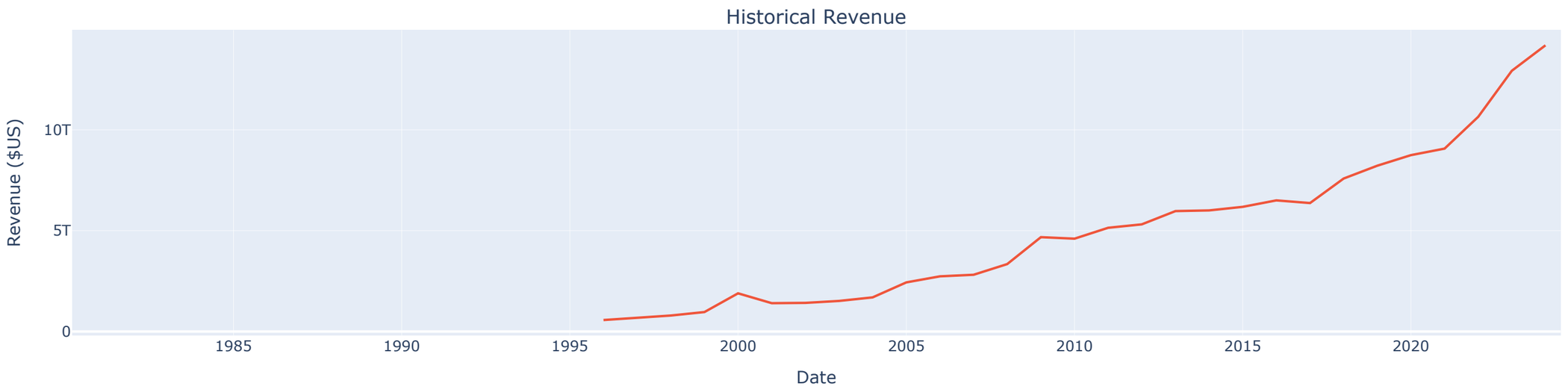
In [7]:
```
# URLs for bank revenue data
urls = {
    "JPMorgan Chase": "https://companiesmarketcap.com/inr/jp-morgan-chase/revenue/",
    "Wells Fargo": "https://companiesmarketcap.com/inr/wells-fargo/revenue/",
    "Bank of America": "https://companiesmarketcap.com/inr/bank-of-america/revenue/",
    "Goldman Sachs": "https://companiesmarketcap.com/inr/goldman-sachs/revenue/",
    "Morgan Stanley": "https://companiesmarketcap.com/inr/morgan-stanley/revenue/",
    "Capital One": "https://companiesmarketcap.com/inr/capital-one/revenue/",
    "American Express": "https://companiesmarketcap.com/inr/american-express/revenue/"
}
```

Share Price and Revenue Plots

In [9]:
```
#JPMorgan Chase
stock_symbol = 'JPM'
jpmc_stock_data = get_stock_data(stock_symbol)
jpmc_revenue_data = get_revenue_data(urls["JPMorgan Chase"])

make_graph(jpmc_stock_data, jpmc_revenue_data, "JPMorgan Chase")
```

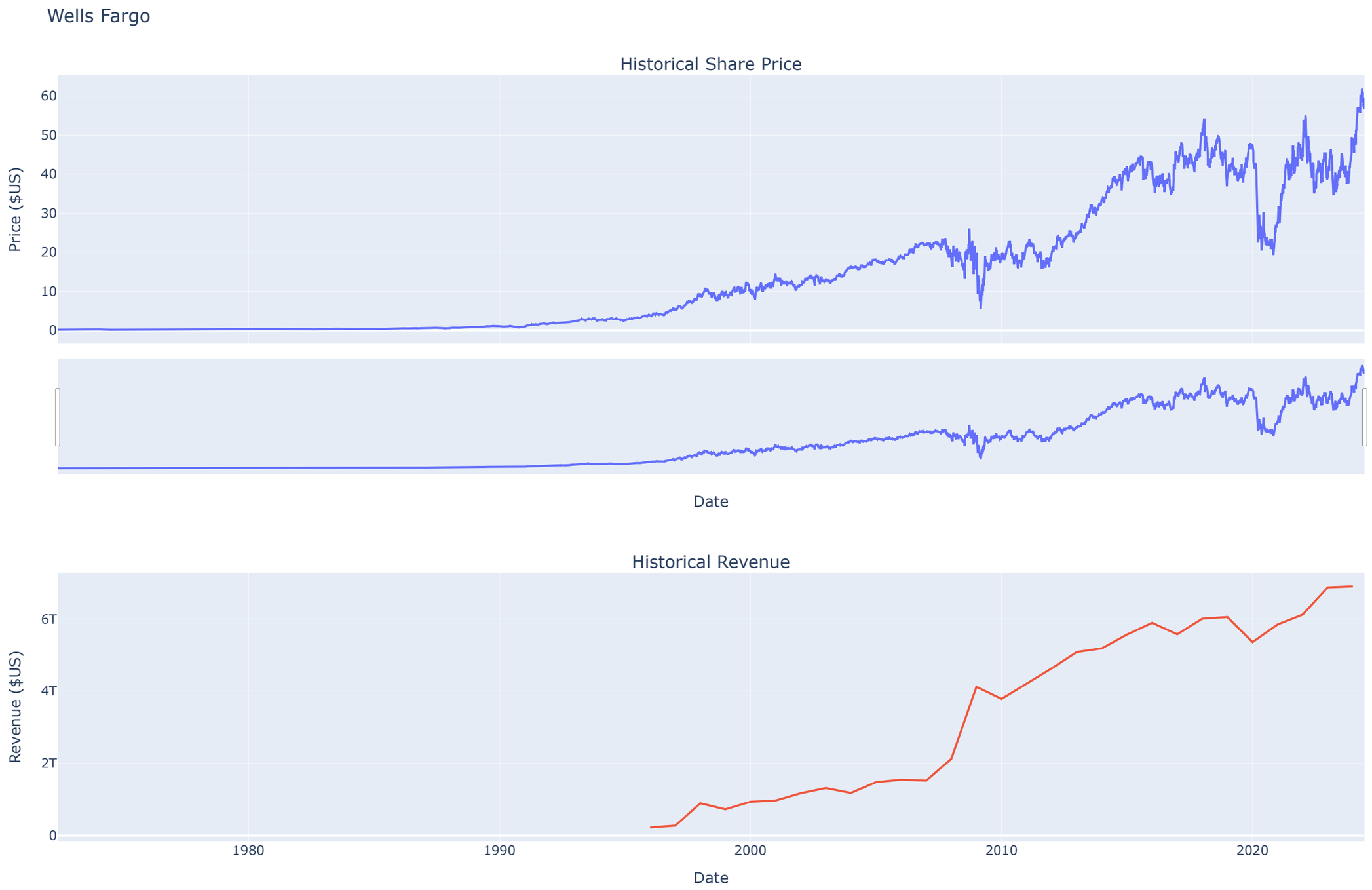
JPMorgan Chase





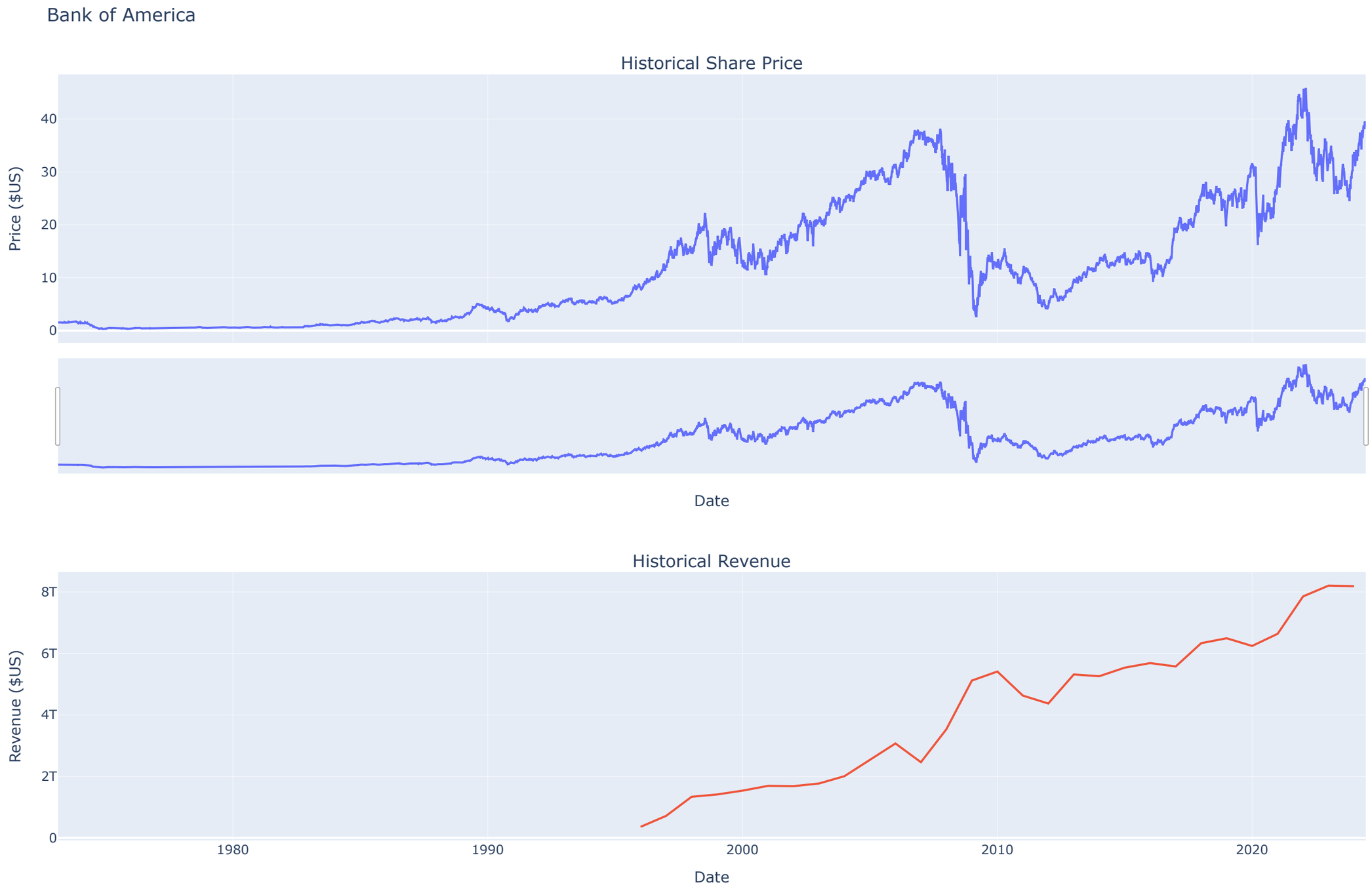
In [10]:
```
#Wells Fargo
stock_symbol = 'WFC'
wellsf_stock_data = get_stock_data(stock_symbol)
wellsf_revenue_data = get_revenue_data(urls["Wells Fargo"])
```

```
make_graph(wellsf_stock_data, wellsf_revenue_data, "Wells Fargo")
```



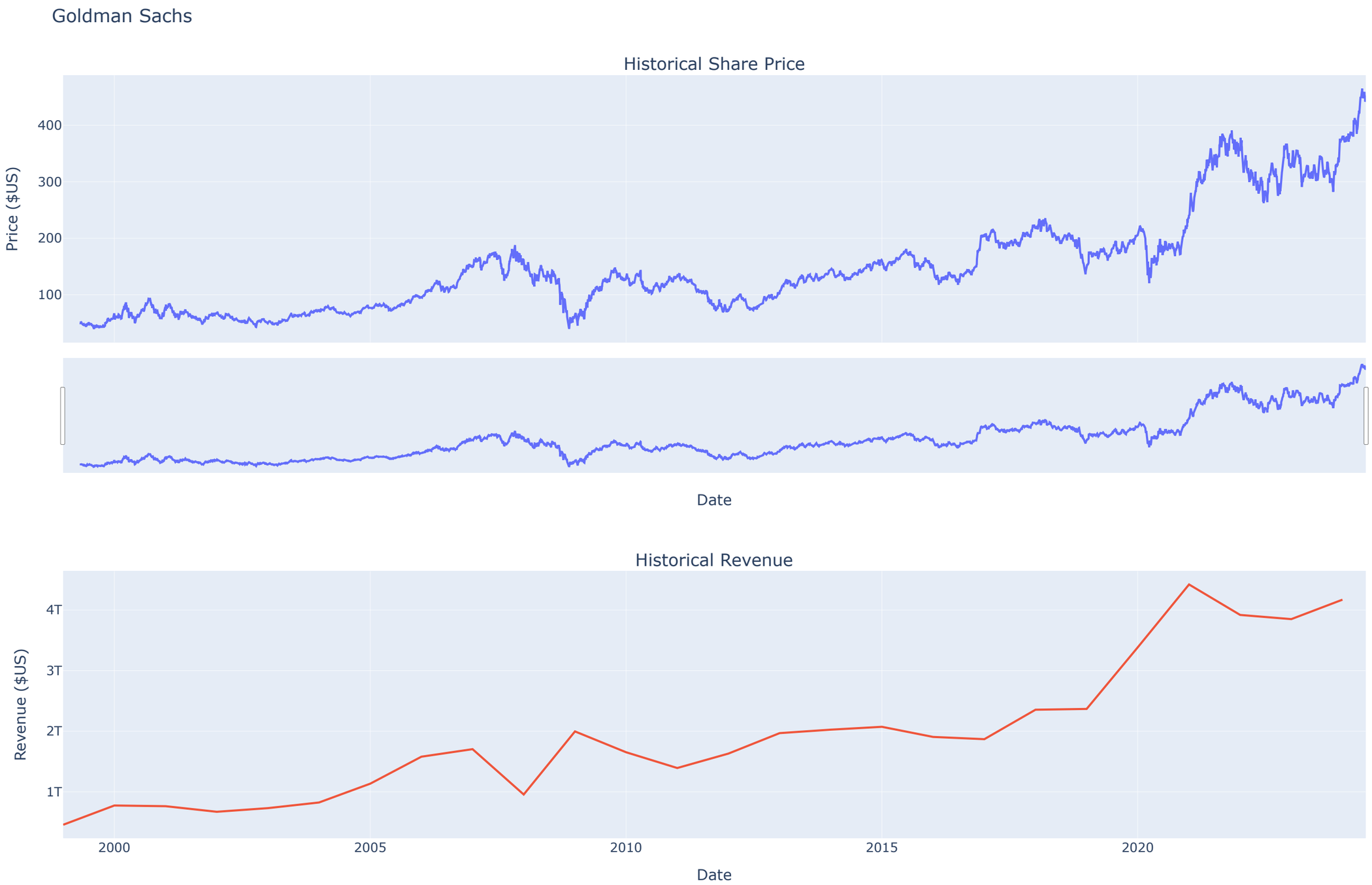
In [11]:
```
#Bank of America
stock_symbol = 'BAC'
boa_stock_data = get_stock_data(stock_symbol)
```

```
boa_revenue_data = get_revenue_data(urls["Bank of America"])

make_graph(boa_stock_data, boa_revenue_data, "Bank of America")
```

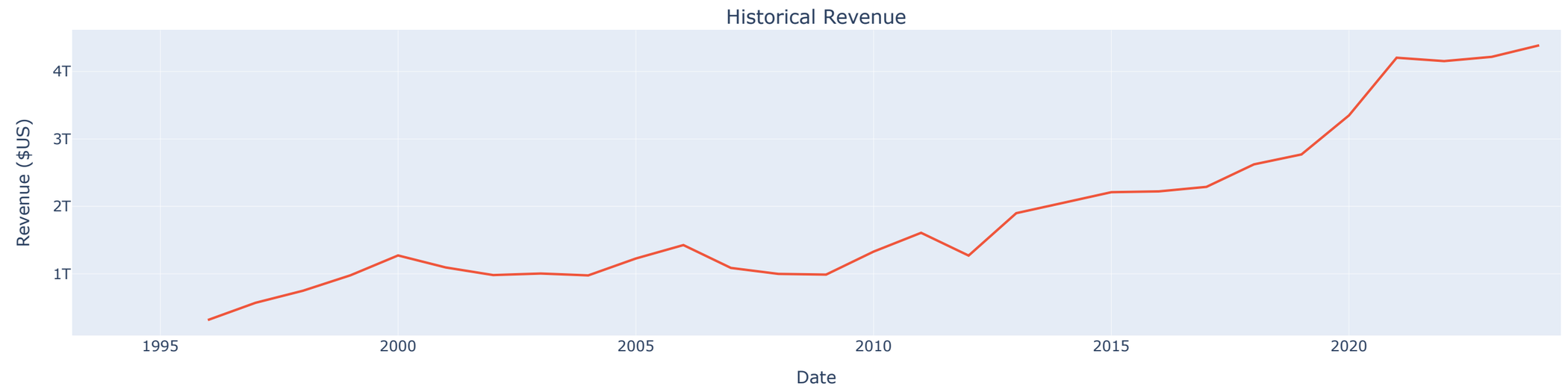


In [12]:
```
#Goldman Sachs
stock_symbol = 'GS'
```

```
goldms_stock_data = get_stock_data(stock_symbol)
goldms_revenue_data = get_revenue_data(urls["Goldman Sachs"])

make_graph(goldms_stock_data, goldms_revenue_data, "Goldman Sachs")
```



In [13]:
```
#Morgan Stanley
stock_symbol = 'MS'
morgans_stock_data = get_stock_data(stock_symbol)
morgans_revenue_data = get_revenue_data(urls["Morgan Stanley"])

make_graph(morgans_stock_data, morgans_revenue_data, "Morgan Stanley")
```



In [14]:
```
#Capital One
stock_symbol = 'COF'
capitalo_stock_data = get_stock_data(stock_symbol)
capitalo_revenue_data = get_revenue_data(urls["Capital One"])
```

```
make_graph(capitalo_stock_data, capitalo_revenue_data, "Capital One")
```

****Exploratory Data Analysis****

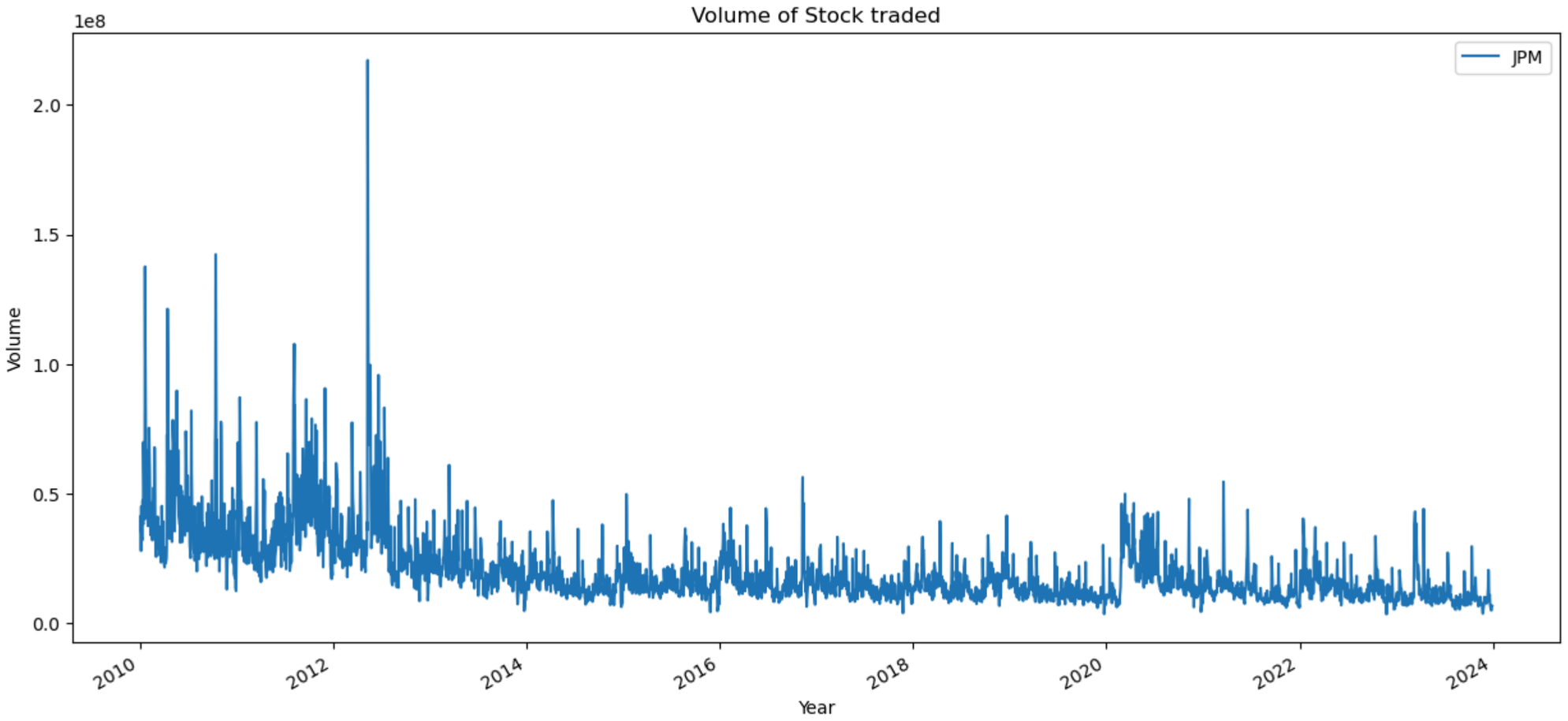
In [16]:
```
start = "2010-01-01"
end = '2024-1-01'
JPM = yf.download('JPM',start,end)
BOA = yf.download('BAC',start,end)
WF = yf.download('WFC',start,end)
GS = yf.download('GS',start,end)
MS = yf.download('MS',start,end)
CF = yf.download('COF',start,end)
```

```
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
```

**Volume **

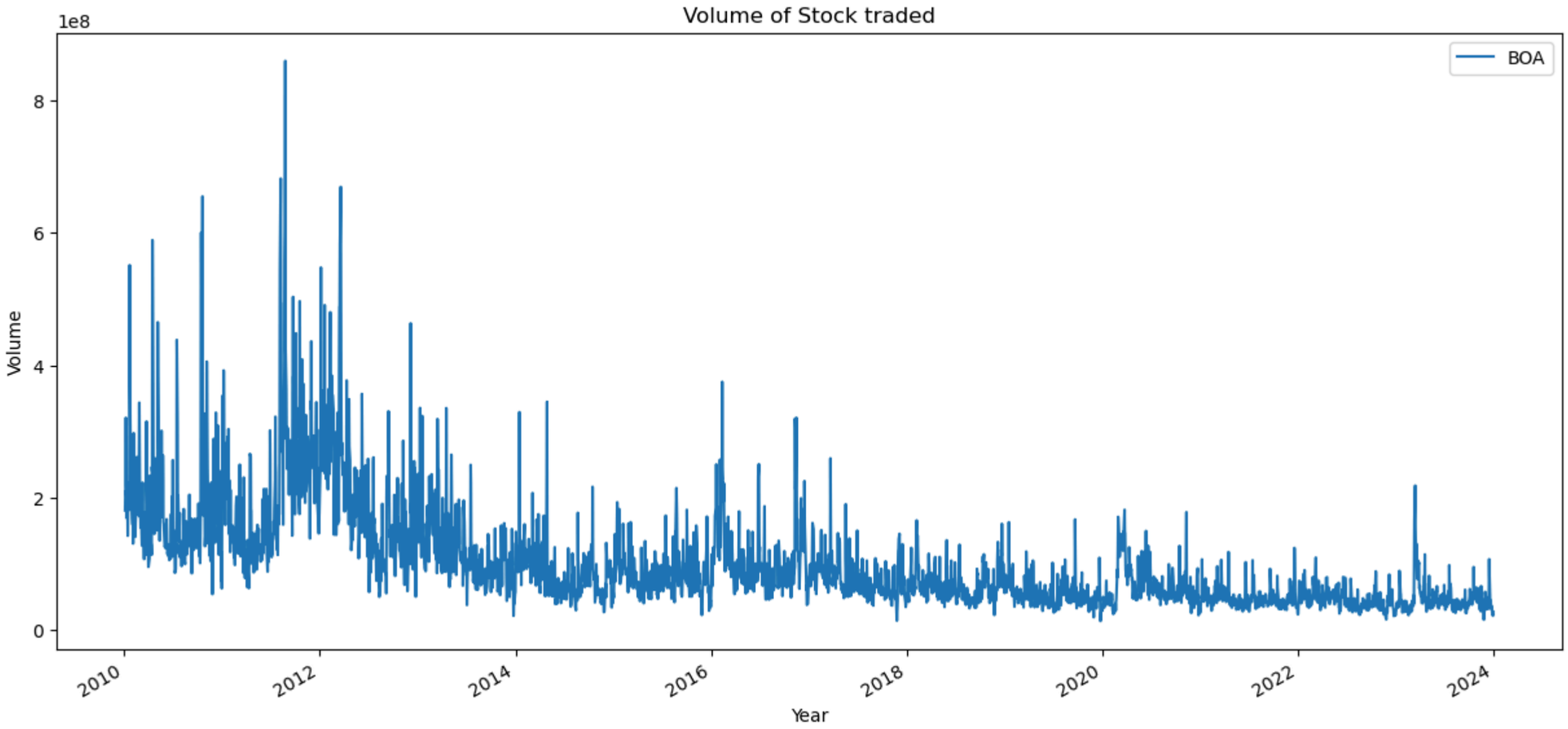
In [18]:
```
import matplotlib.pyplot as plt
JPM['Volume'].plot(label = 'JPM', figsize = (15,7))
plt.xlabel('Year')
plt.ylabel('Volume')
plt.title('Volume of Stock traded')
plt.legend()
```

Out[18]:
```
<matplotlib.legend.Legend at 0x26f0fd82630>
```



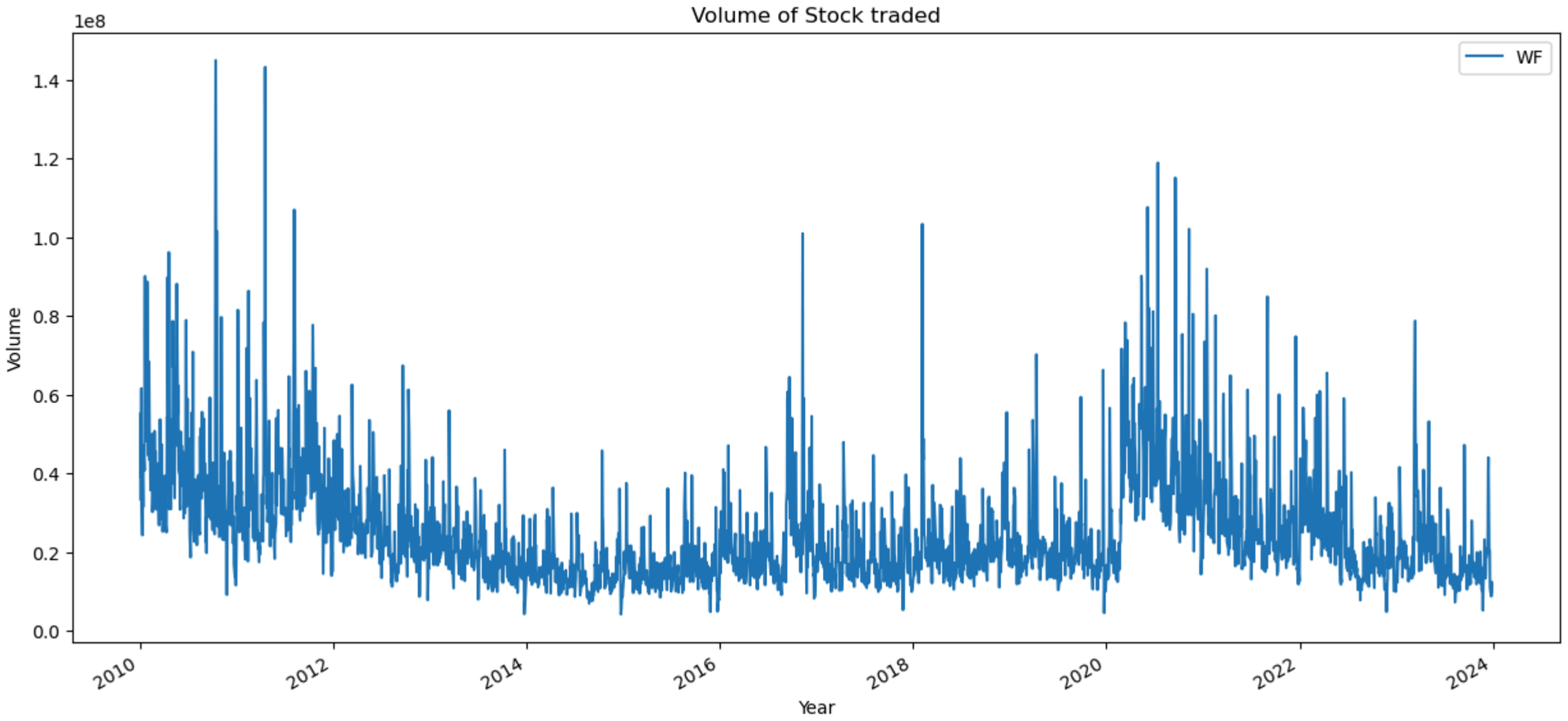
In [19]:
```
BOA['Volume'].plot(label = "BOA", figsize = (15,7))
plt.xlabel('Year')
plt.ylabel('Volume')
plt.title('Volume of Stock traded')
plt.legend()
```

Out[19]:
```
<matplotlib.legend.Legend at 0x26f0d8c60c0>
```



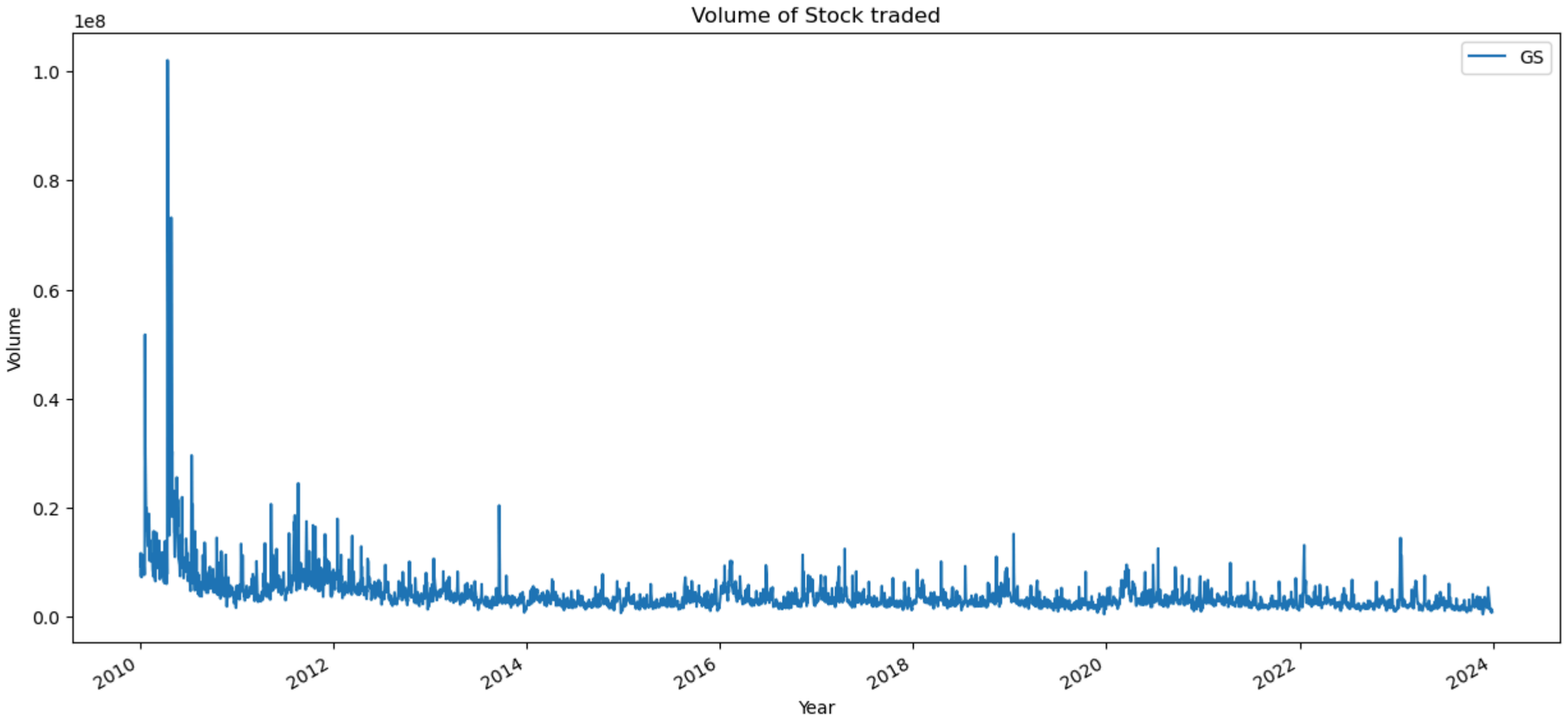
In [20]:
```
WF['Volume'].plot(label = 'WF', figsize = (15,7))
plt.xlabel('Year')
plt.ylabel('Volume')
plt.title('Volume of Stock traded')
plt.legend()
```

Out[20]: <matplotlib.legend.Legend at 0x26f0d882570>



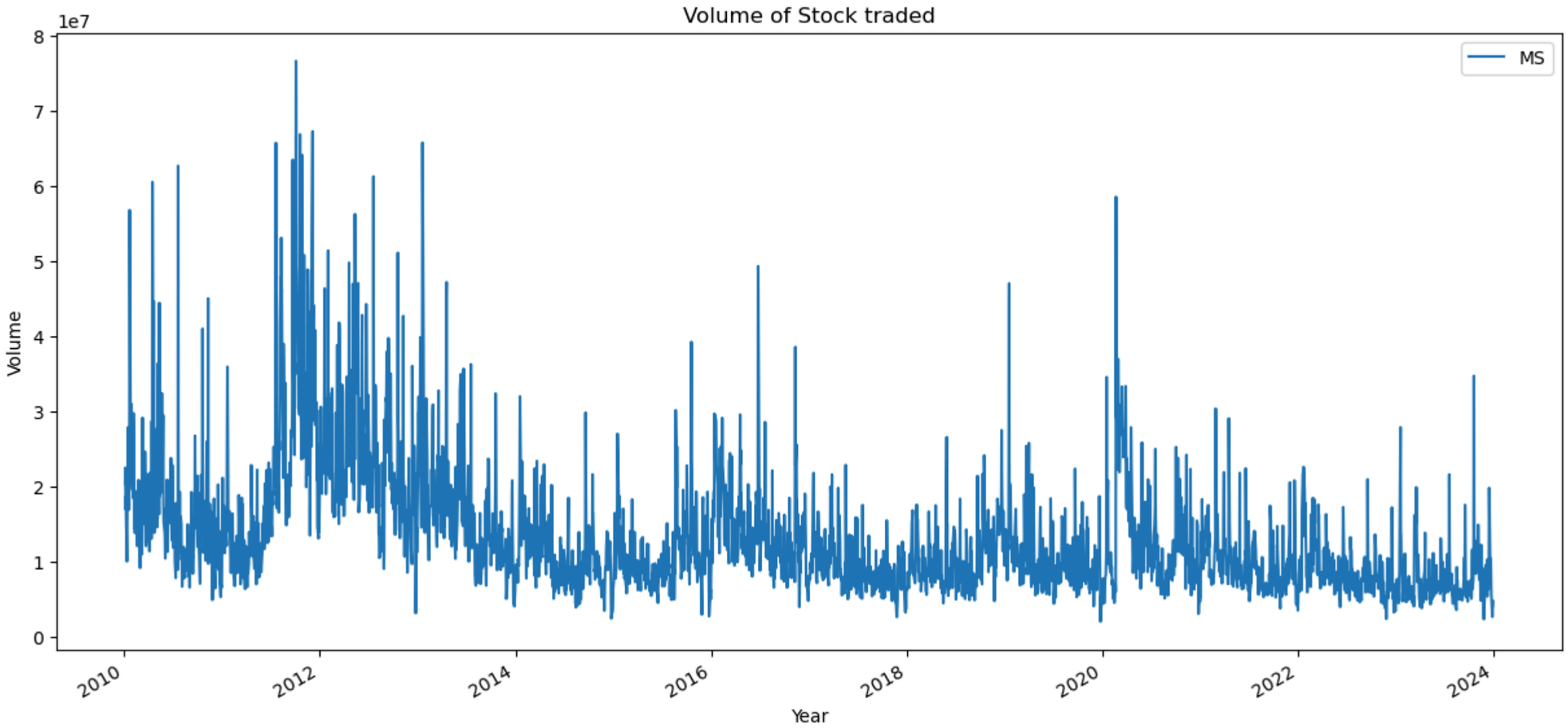
In [21]:
```
GS['Volume'].plot(label = 'GS', figsize = (15,7))
plt.xlabel('Year')
plt.ylabel('Volume')
plt.title('Volume of Stock traded')
plt.legend()
```

Out[21]:
```
<matplotlib.legend.Legend at 0x26f0f2bf8f0>
```



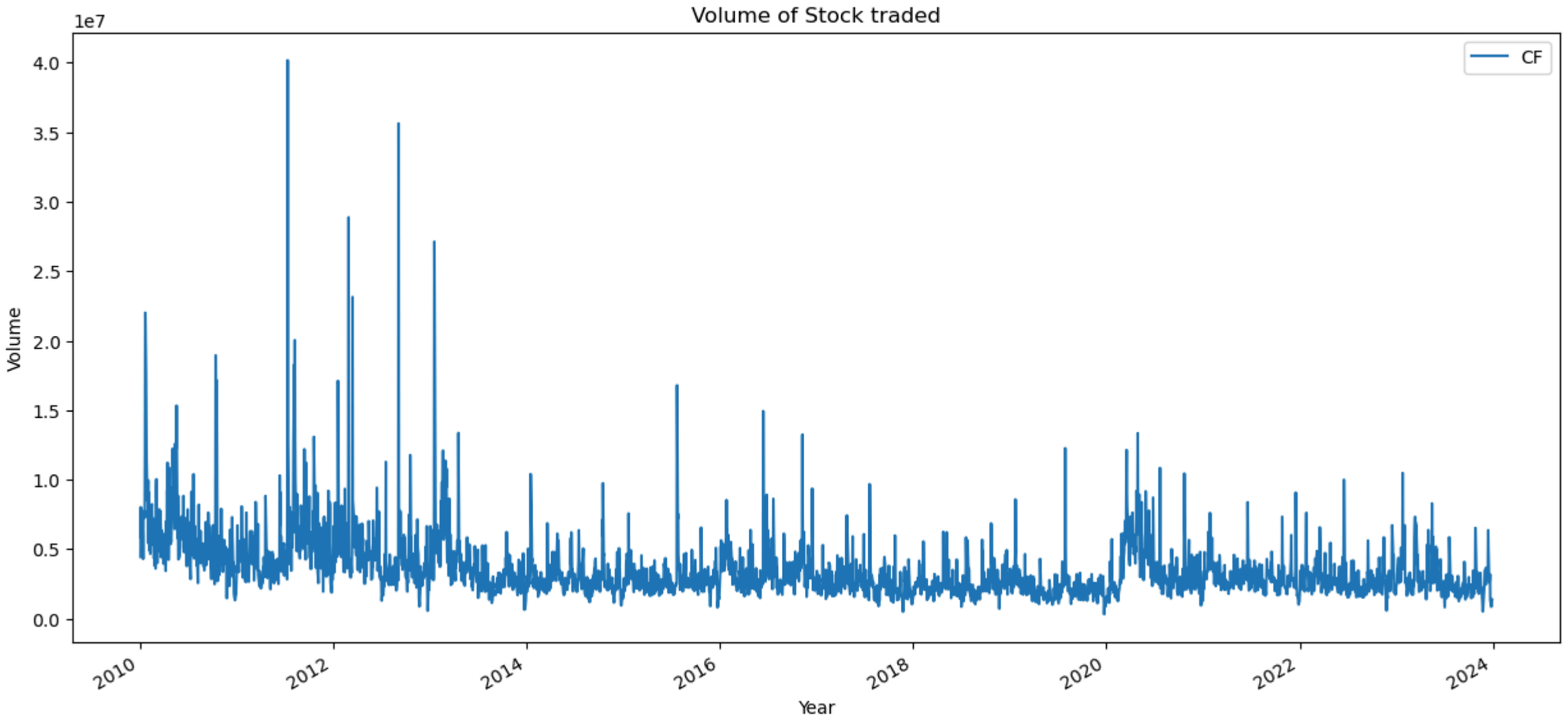
In [22]:
```
MS['Volume'].plot(label = 'MS', figsize = (15,7))
plt.xlabel('Year')
plt.ylabel('Volume')
plt.title('Volume of Stock traded')
plt.legend()
```

Out[22]:
```
<matplotlib.legend.Legend at 0x26f0f2e46e0>
```



In [23]:
```
CF['Volume'].plot(label = 'CF', figsize = (15,7))
plt.xlabel('Year')
plt.ylabel('Volume')
plt.title('Volume of Stock traded')
plt.legend()
```

Out[23]:
```
<matplotlib.legend.Legend at 0x26f135d40e0>
```



Moving Averages

In [25]:
```
#JPM
JPM['MA50'] = JPM['Open'].rolling(50).mean()
JPM['MA200'] = JPM['Open'].rolling(200).mean()
JPM['Open'].plot(figsize = (15,7))
JPM['MA50'].plot()
JPM['MA200'].plot()
```

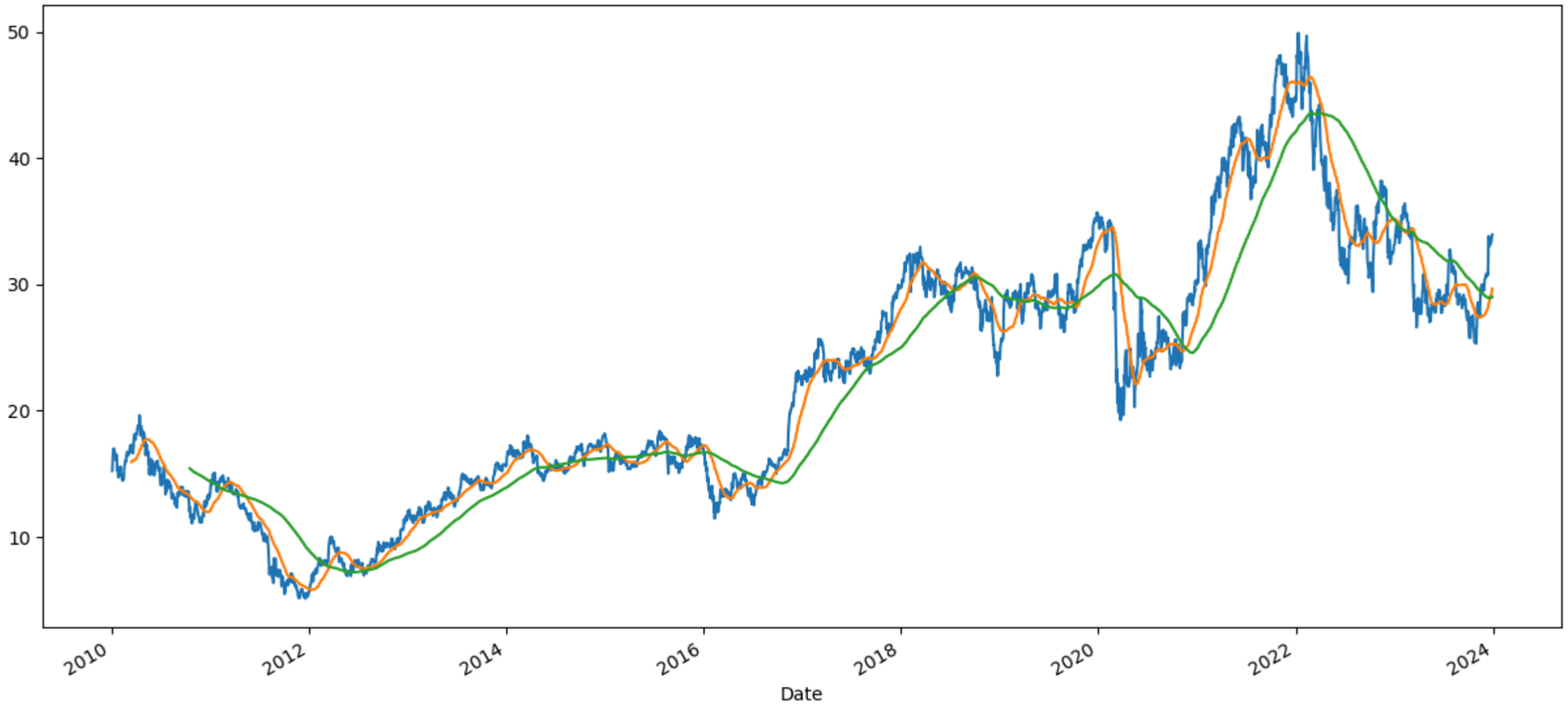
Out[25]:
```
<Axes: xlabel='Date'>
```

In [26]:
```
#JPM
JPM['MA50'] = JPM['Open'].rolling(50).mean()
JPM['MA200'] = JPM['Open'].rolling(200).mean()
JPM['Open'].plot(figsize = (15,7))
JPM['MA50'].plot()
JPM['MA200'].plot()
```

Out[26]: <Axes: xlabel='Date'>

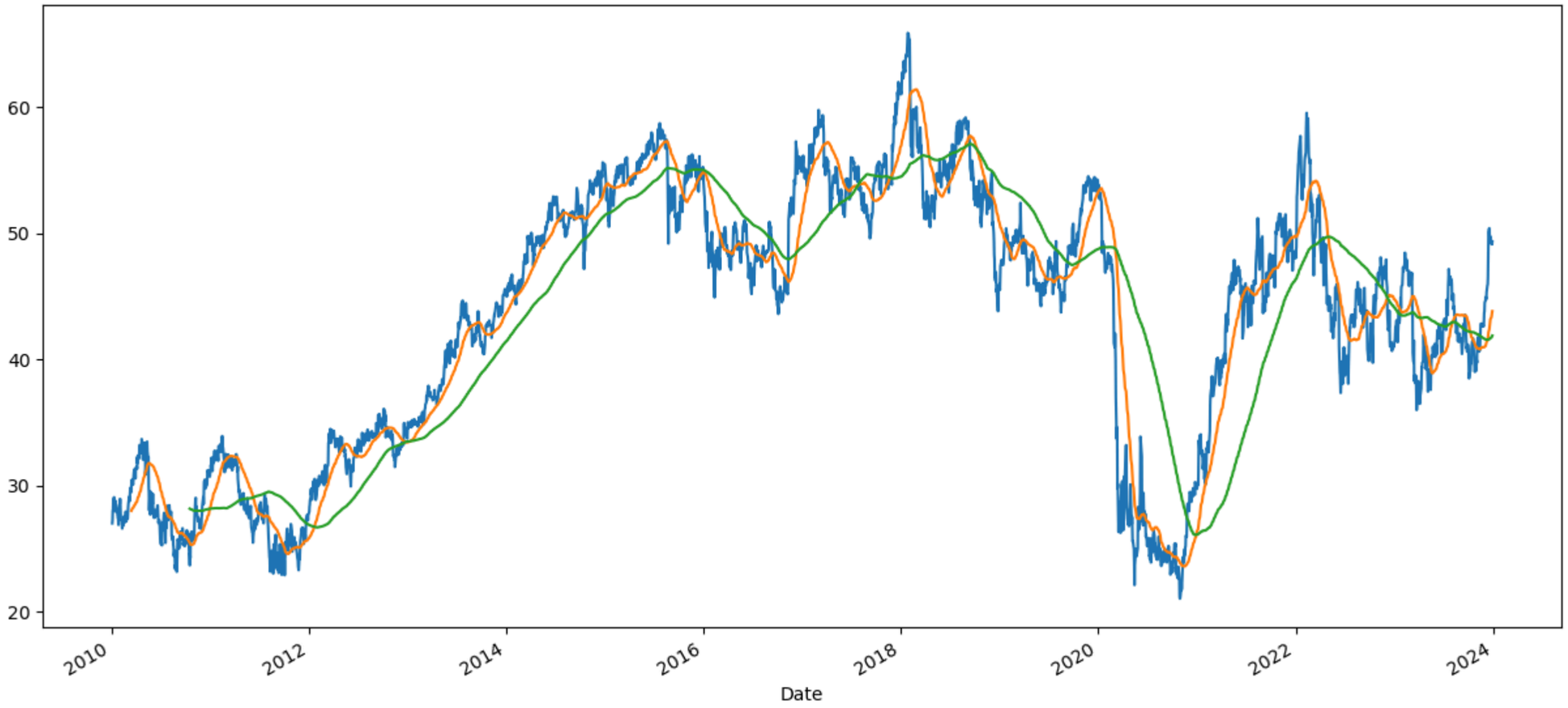
In [27]:
```
#BOA
BOA['MA50'] = BOA['Open'].rolling(50).mean()
BOA['MA200'] = BOA['Open'].rolling(200).mean()
BOA['Open'].plot(figsize = (15,7))
BOA['MA50'].plot()
BOA['MA200'].plot()
```

Out[27]: <Axes: xlabel='Date'>



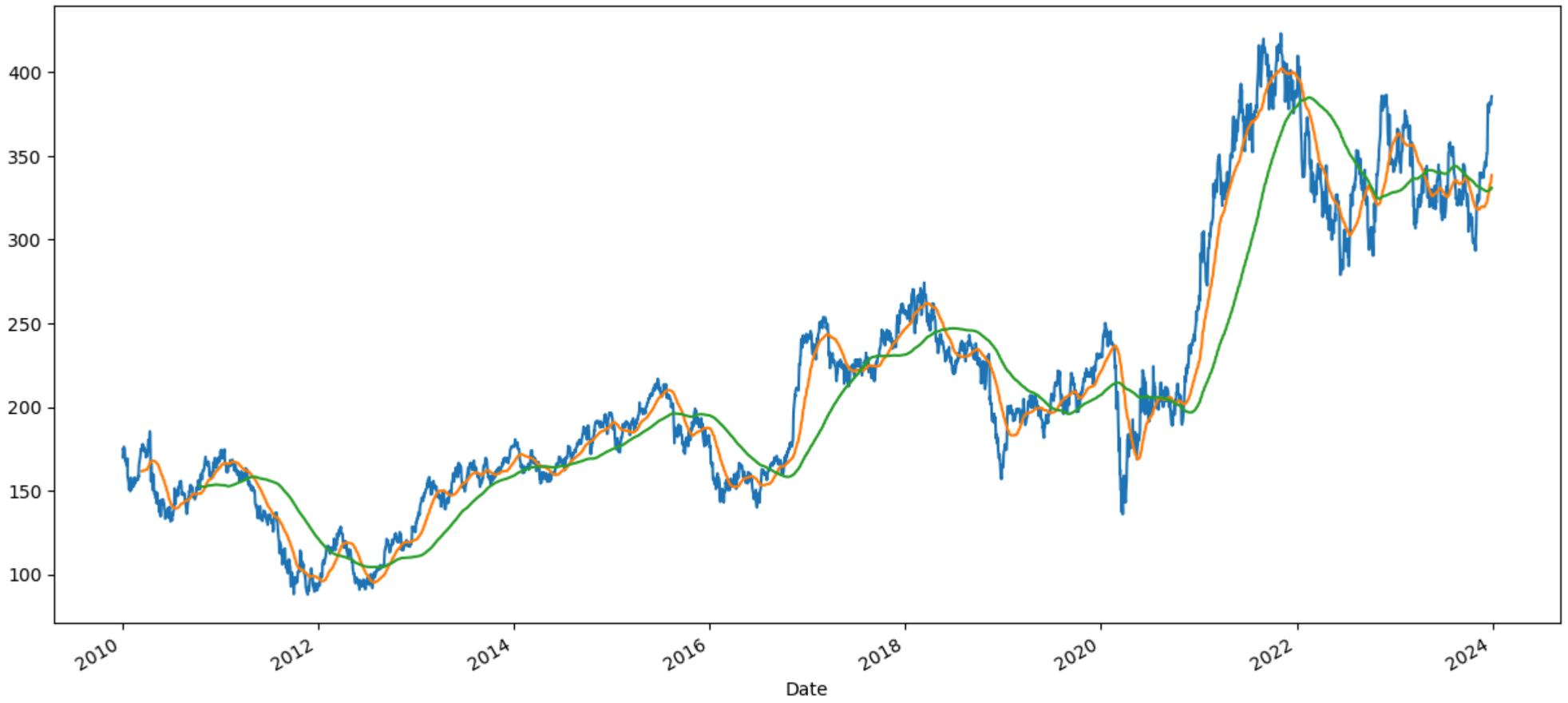
In [28]:
```
#WF
WF['MA50'] = WF['Open'].rolling(50).mean()
WF['MA200'] = WF['Open'].rolling(200).mean()
WF['Open'].plot(figsize = (15,7))
WF['MA50'].plot()
WF['MA200'].plot()
```

Out[28]: <Axes: xlabel='Date'>



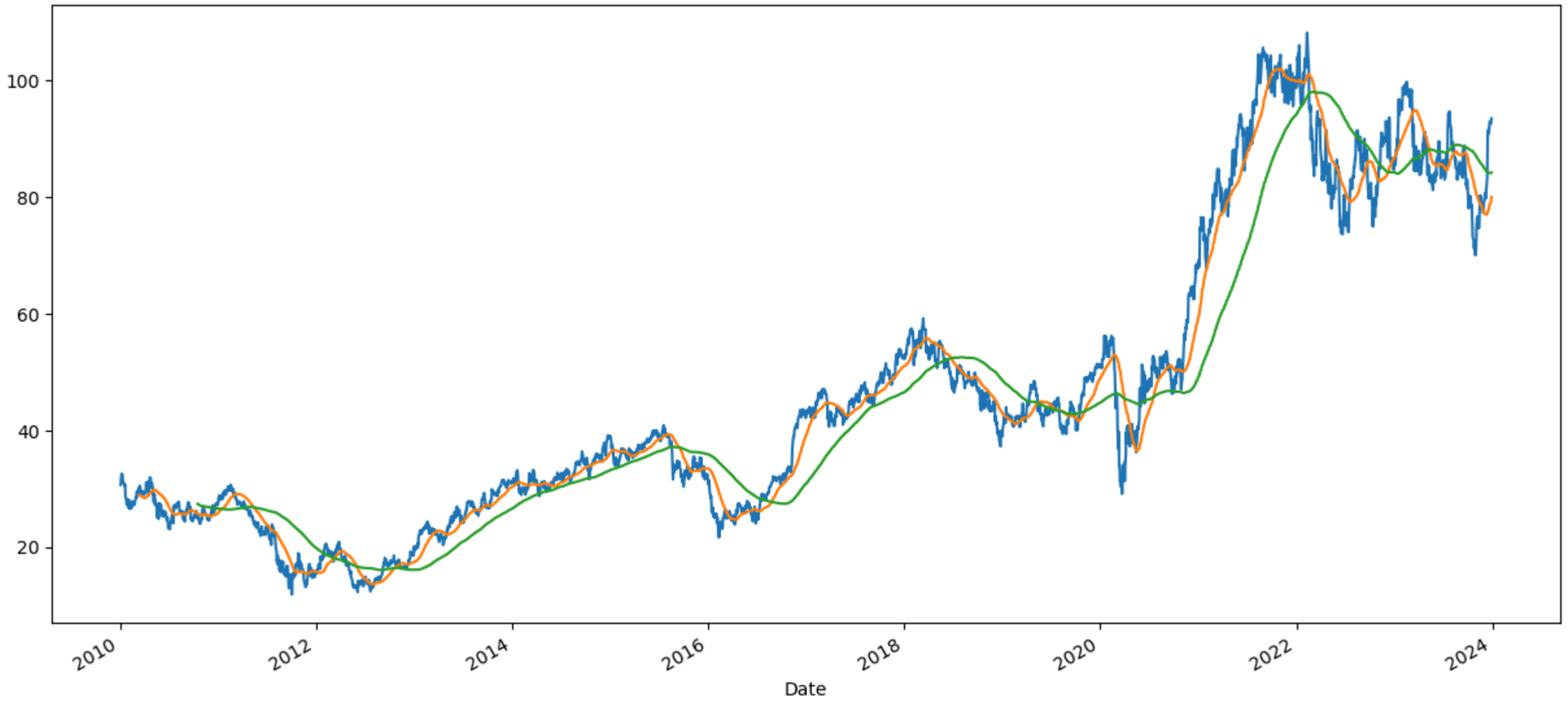
In [29]:
```
#GS
GS['MA50'] = GS['Open'].rolling(50).mean()
GS['MA200'] = GS['Open'].rolling(200).mean()
GS['Open'].plot(figsize = (15,7))
GS['MA50'].plot()
GS['MA200'].plot()
```

Out[29]: <Axes: xlabel='Date'>



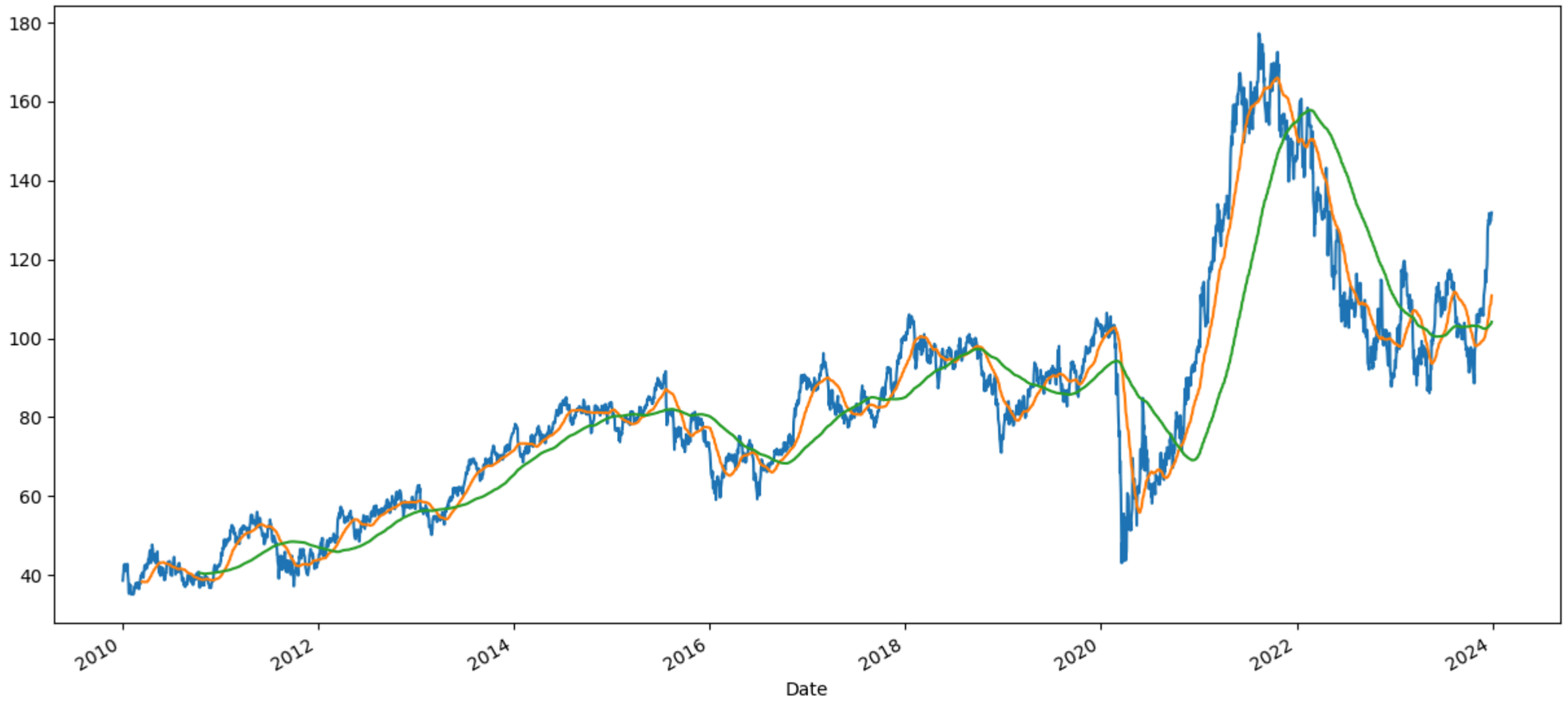
In [30]:
```
#MS
MS['MA50'] = MS['Open'].rolling(50).mean()
MS['MA200'] = MS['Open'].rolling(200).mean()
MS['Open'].plot(figsize = (15,7))
MS['MA50'].plot()
MS['MA200'].plot()
```

Out[30]: <Axes: xlabel='Date'>



In [31]:
```
#CF
CF['MA50'] = CF['Open'].rolling(50).mean()
CF['MA200'] = CF['Open'].rolling(200).mean()
CF['Open'].plot(figsize = (15,7))
CF['MA50'].plot()
CF['MA200'].plot()
```

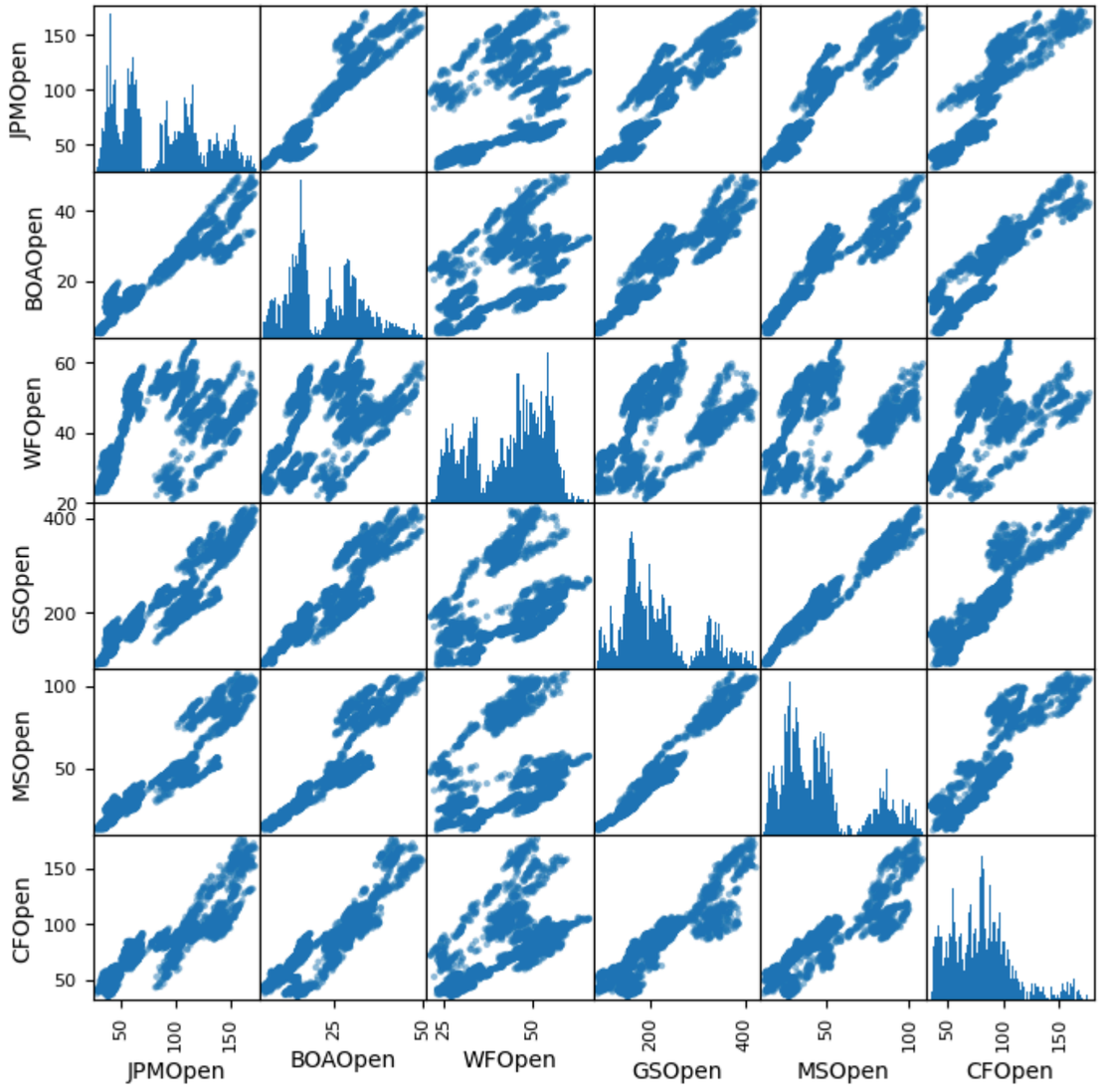
Out[31]: <Axes: xlabel='Date'>



ScatterPlot Matrix

In [33]:
```
from pandas.plotting import scatter_matrix
data = pd.concat([JPM['Open'],BOA['Open'],WF['Open'],GS['Open'],MS['Open'],CF['Open']],axis = 1)
data.columns = ['JPMOpen','BOAOpen','WFOpen','GSOpen','MSOpen','CFOpen']
scatter_matrix(data, figsize = (8,8), hist_kwds= {'bins':250})
```

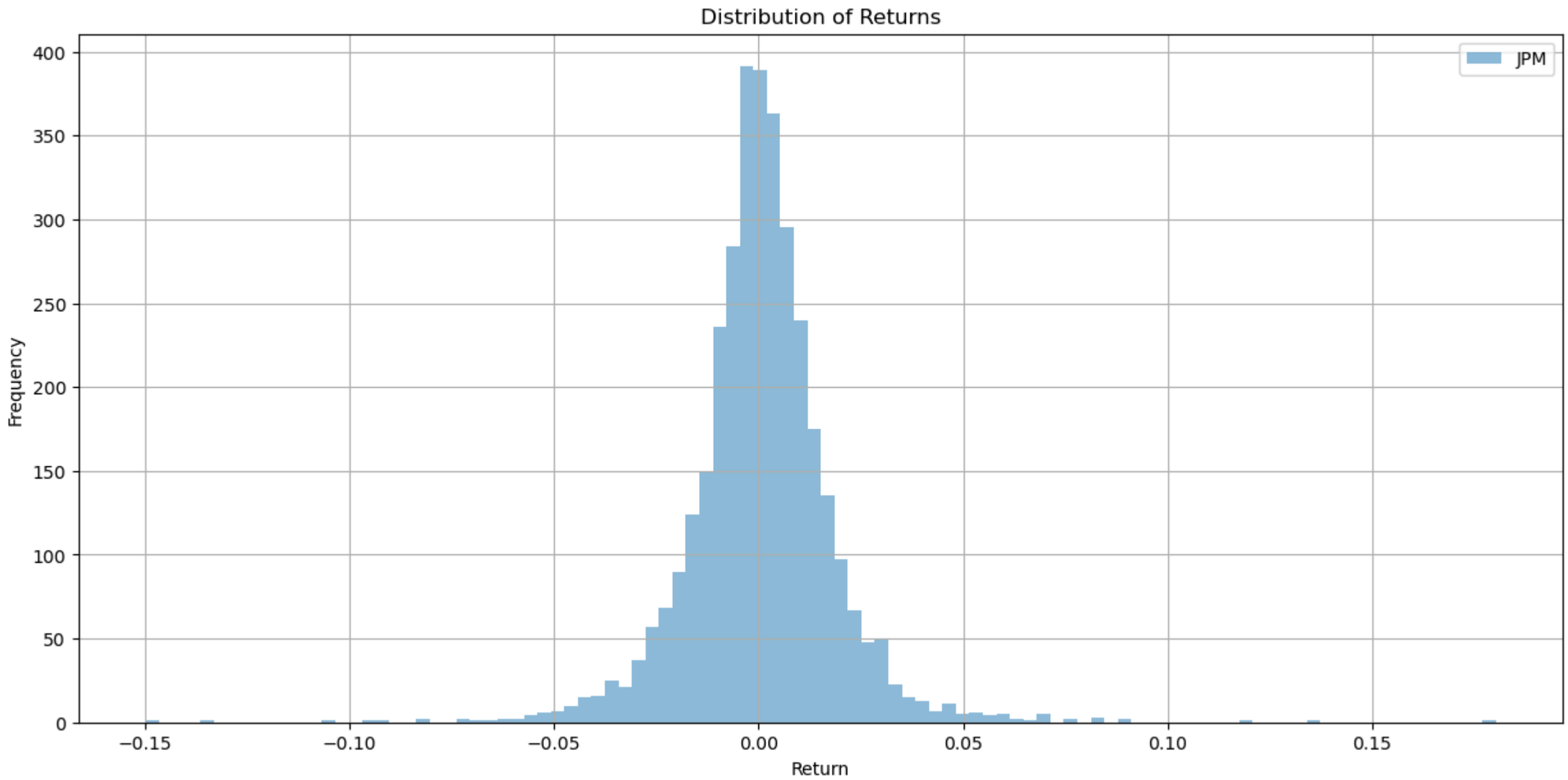
Out[33]:
```
array([[<Axes: xlabel='JPMOpen', ylabel='JPMOpen'>,
        <Axes: xlabel='BOAOpen', ylabel='JPMOpen'>,
        <Axes: xlabel='WFOpen', ylabel='JPMOpen'>,
        <Axes: xlabel='GSOpen', ylabel='JPMOpen'>,
        <Axes: xlabel='MSOpen', ylabel='JPMOpen'>,
        <Axes: xlabel='CFOpen', ylabel='JPMOpen'>],
       [<Axes: xlabel='JPMOpen', ylabel='BOAOpen'>,
        <Axes: xlabel='BOAOpen', ylabel='BOAOpen'>,
        <Axes: xlabel='WFOpen', ylabel='BOAOpen'>,
        <Axes: xlabel='GSOpen', ylabel='BOAOpen'>,
        <Axes: xlabel='MSOpen', ylabel='BOAOpen'>,
        <Axes: xlabel='CFOpen', ylabel='BOAOpen'>],
       [<Axes: xlabel='JPMOpen', ylabel='WFOpen'>,
        <Axes: xlabel='BOAOpen', ylabel='WFOpen'>,
        <Axes: xlabel='WFOpen', ylabel='WFOpen'>,
        <Axes: xlabel='GSOpen', ylabel='WFOpen'>,
        <Axes: xlabel='MSOpen', ylabel='WFOpen'>,
        <Axes: xlabel='CFOpen', ylabel='WFOpen'>],
       [<Axes: xlabel='JPMOpen', ylabel='GSOpen'>,
        <Axes: xlabel='BOAOpen', ylabel='GSOpen'>,
        <Axes: xlabel='WFOpen', ylabel='GSOpen'>,
        <Axes: xlabel='GSOpen', ylabel='GSOpen'>,
        <Axes: xlabel='MSOpen', ylabel='GSOpen'>,
        <Axes: xlabel='CFOpen', ylabel='GSOpen'>],
       [<Axes: xlabel='JPMOpen', ylabel='MSOpen'>,
        <Axes: xlabel='BOAOpen', ylabel='MSOpen'>,
        <Axes: xlabel='WFOpen', ylabel='MSOpen'>,
        <Axes: xlabel='GSOpen', ylabel='MSOpen'>,
        <Axes: xlabel='MSOpen', ylabel='MSOpen'>,
        <Axes: xlabel='CFOpen', ylabel='MSOpen'>],
       [<Axes: xlabel='JPMOpen', ylabel='CFOpen'>,
        <Axes: xlabel='BOAOpen', ylabel='CFOpen'>,
        <Axes: xlabel='WFOpen', ylabel='CFOpen'>,
        <Axes: xlabel='GSOpen', ylabel='CFOpen'>,
        <Axes: xlabel='MSOpen', ylabel='CFOpen'>,
        <Axes: xlabel='CFOpen', ylabel='CFOpen'>]], dtype=object)
```



Percentage Increase in Stock Value

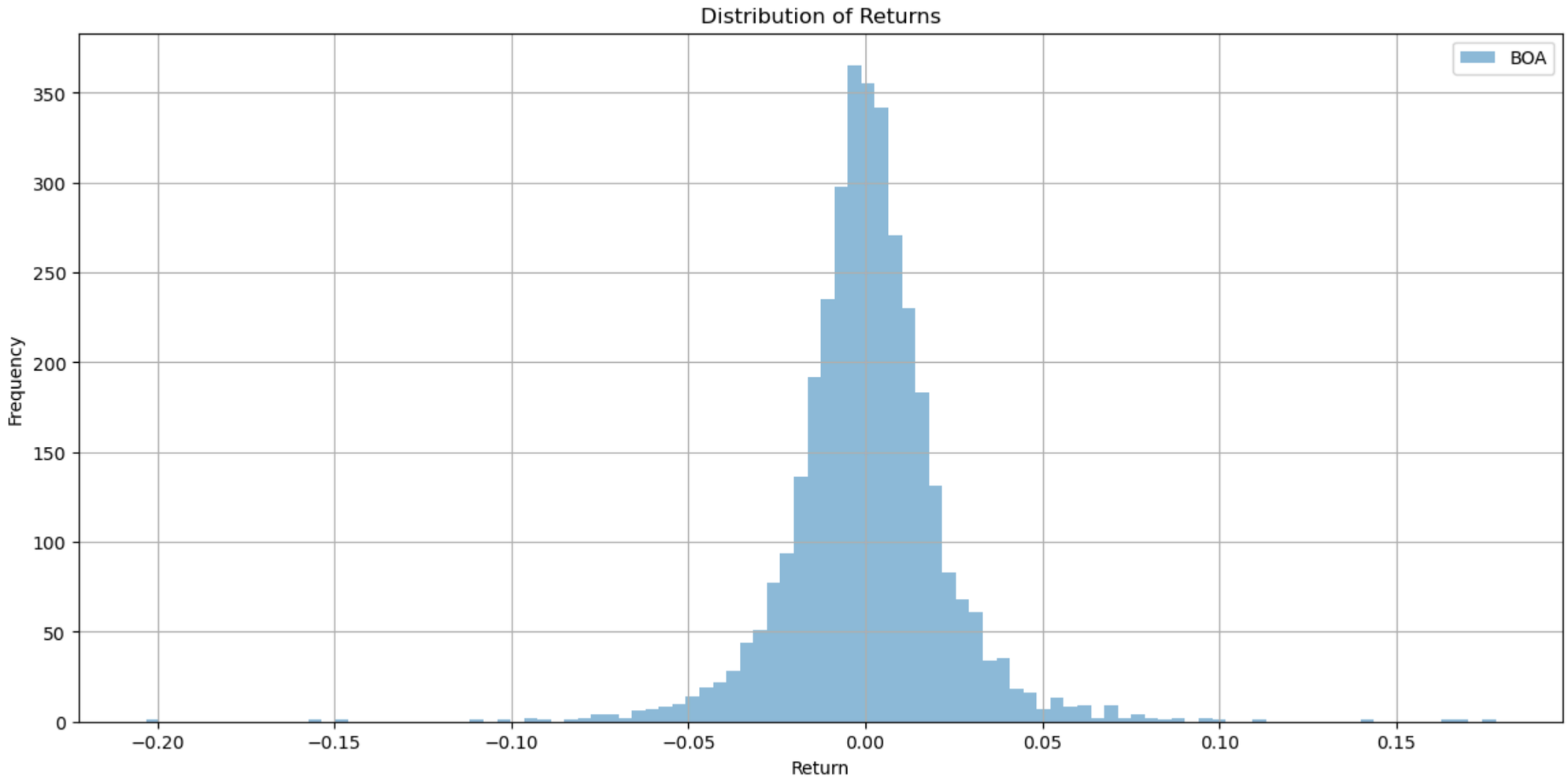
In [35]:
```
#Volatility
JPM['returns'] = (JPM['Close']/JPM['Close'].shift(1)) -1
JPM['returns'].hist(bins = 100, label = 'JPM', alpha = 0.5, figsize = (15,7))
plt.xlabel('Return')
plt.ylabel('Frequency')
plt.title('Distribution of Returns')
plt.legend()
```

Out[35]:
```
<matplotlib.legend.Legend at 0x26f19ae1370>
```



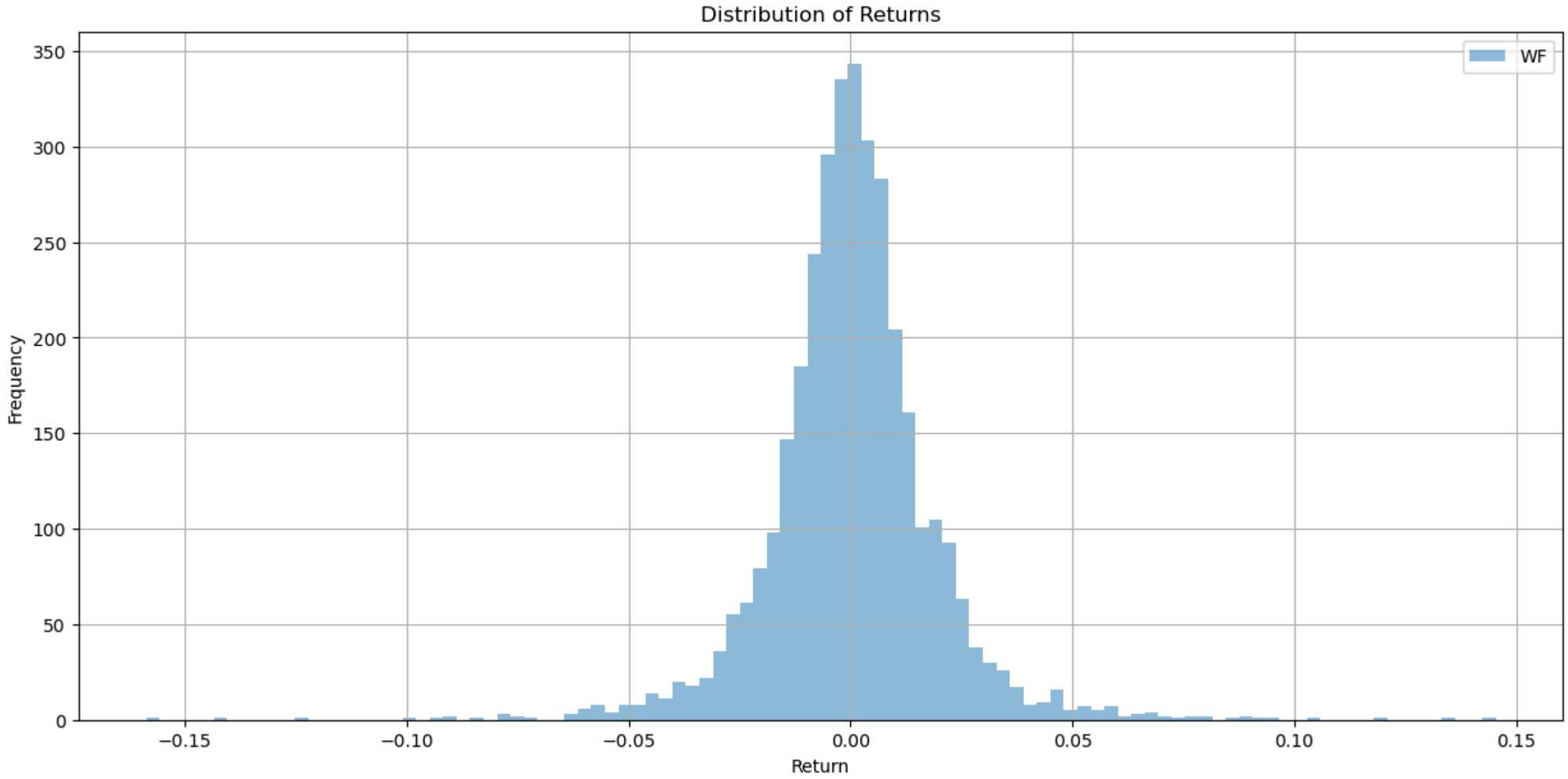
In [36]:
```
BOA['returns'] = (BOA['Close']/BOA['Close'].shift(1))-1
BOA['returns'].hist(bins = 100, label = 'BOA', alpha = 0.5, figsize = (15,7))
plt.xlabel('Return')
plt.ylabel('Frequency')
plt.title('Distribution of Returns')
plt.legend()
```

Out[36]:
```
<matplotlib.legend.Legend at 0x26f16f13890>
```



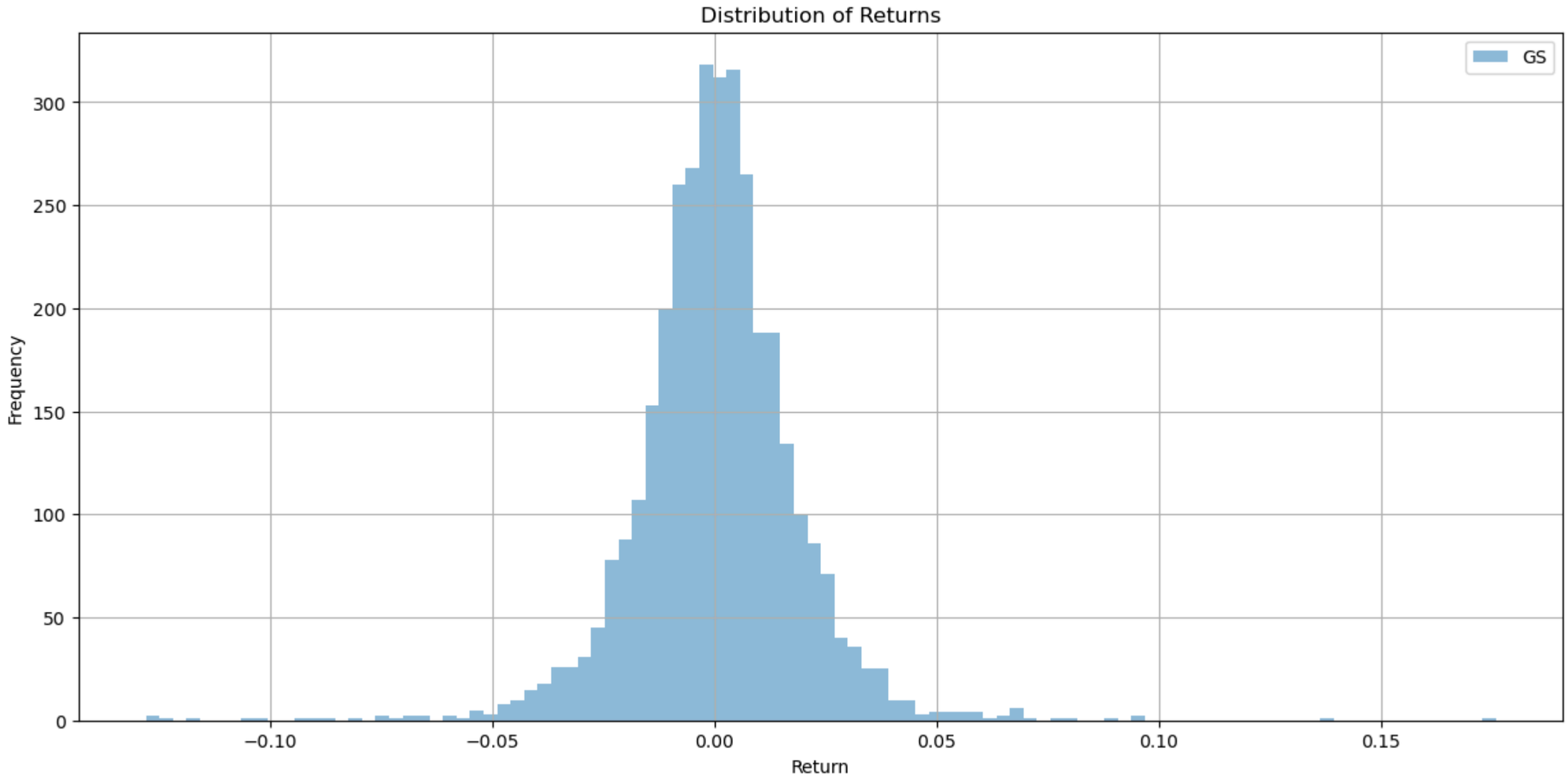
In [37]:
```
WF['returns'] = (WF['Close']/WF['Close'].shift(1)) - 1
WF['returns'].hist(bins = 100, label = 'WF', alpha = 0.5, figsize = (15,7))
plt.xlabel('Return')
plt.ylabel('Frequency')
plt.title('Distribution of Returns')
plt.legend()
```

Out[37]:
```
<matplotlib.legend.Legend at 0x26f1a0ee960>
```



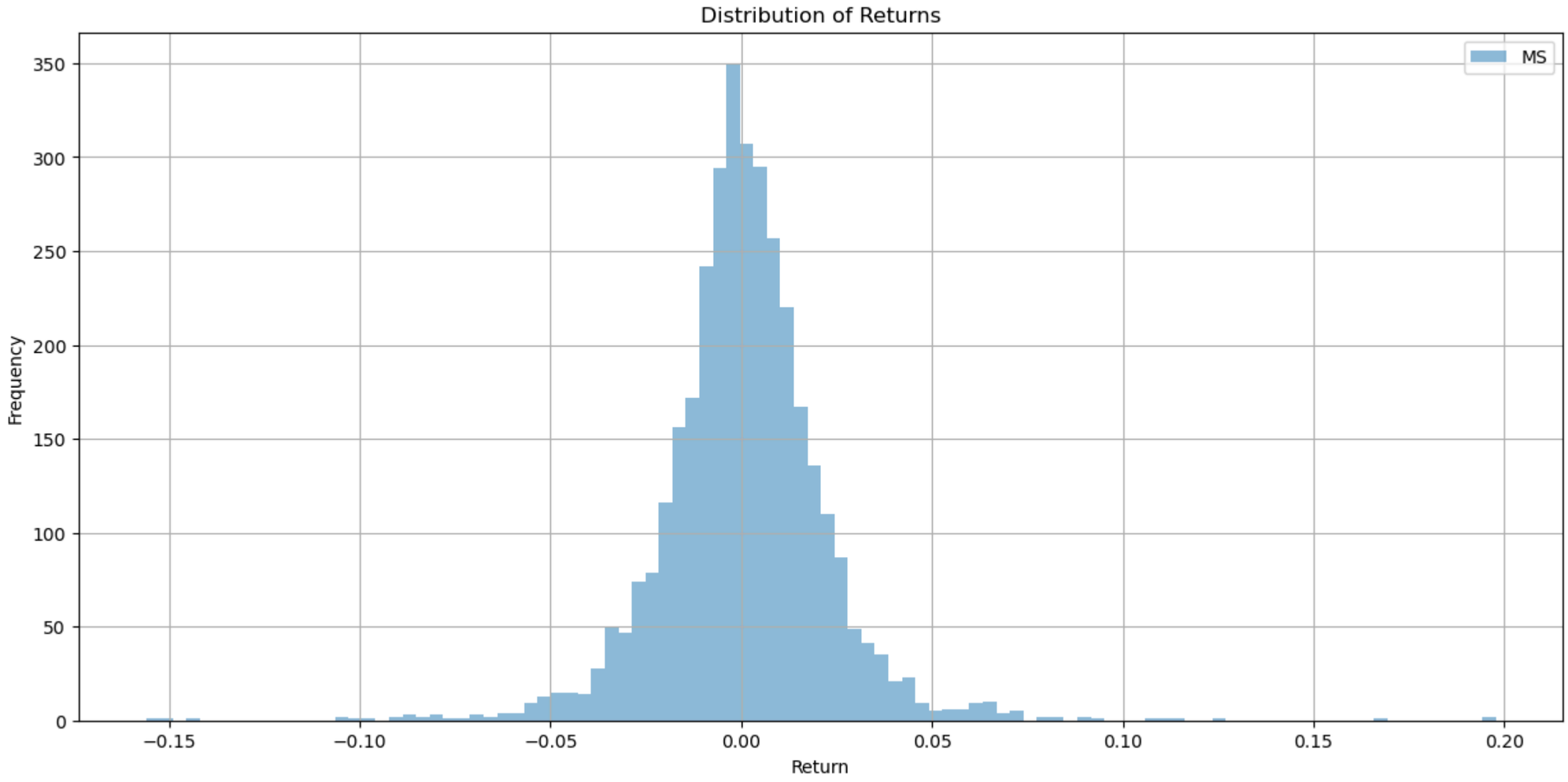
In [38]:
```
GS['returns'] = (GS['Close']/GS['Close'].shift(1)) -1
GS['returns'].hist(bins = 100, label = 'GS', alpha = 0.5, figsize = (15,7))
plt.xlabel('Return')
plt.ylabel('Frequency')
plt.title('Distribution of Returns')
plt.legend()
```

Out[38]: <matplotlib.legend.Legend at 0x26f19e0f530>



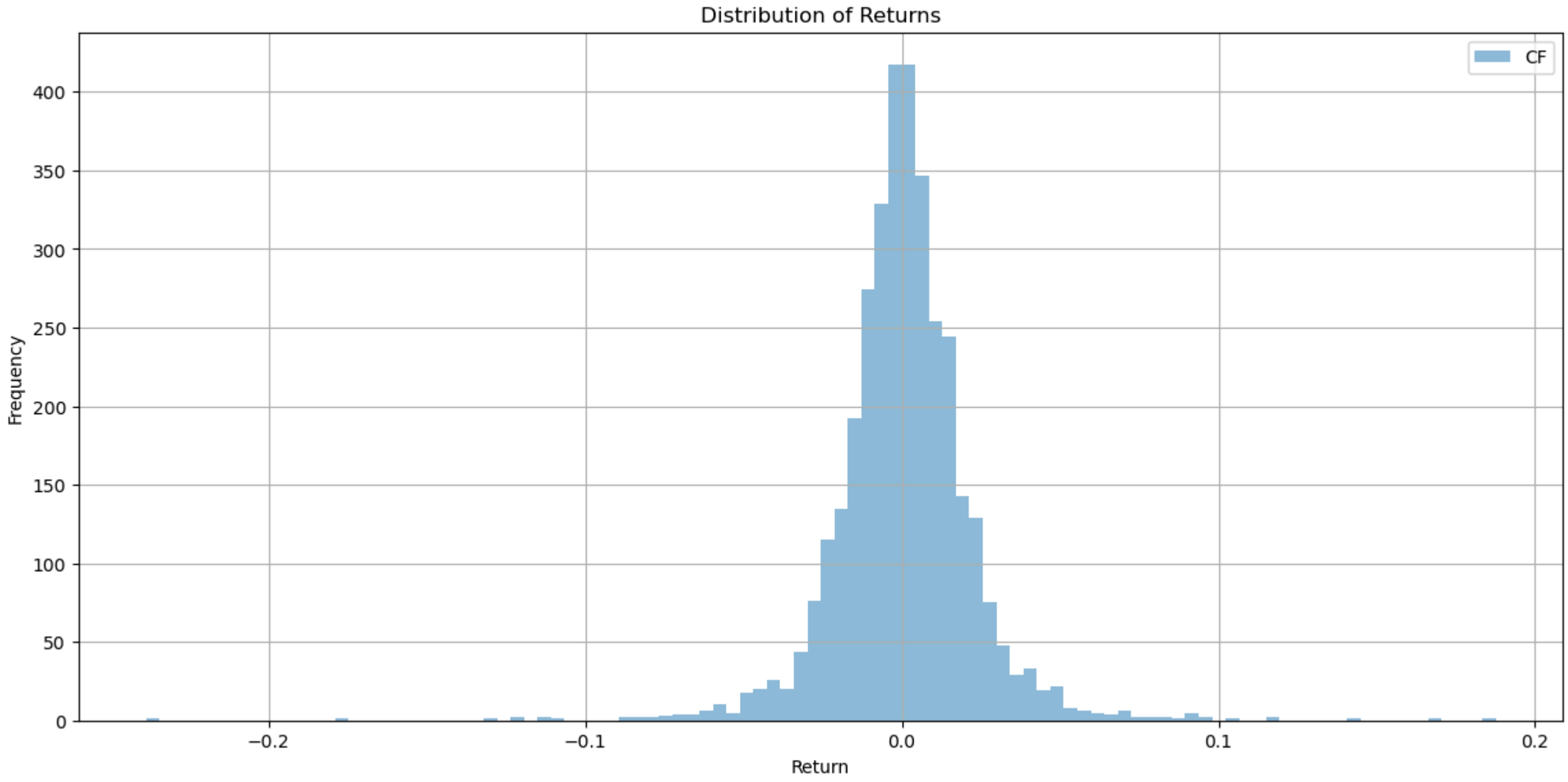
In [39]:
```
MS['returns'] = (MS['Close']/MS['Close'].shift(1)) -1
MS['returns'].hist(bins = 100, label = 'MS', alpha = 0.5, figsize = (15,7))
plt.xlabel('Return')
plt.ylabel('Frequency')
plt.title('Distribution of Returns')
plt.legend()
```

Out[39]:
```
<matplotlib.legend.Legend at 0x26f19ccd760>
```



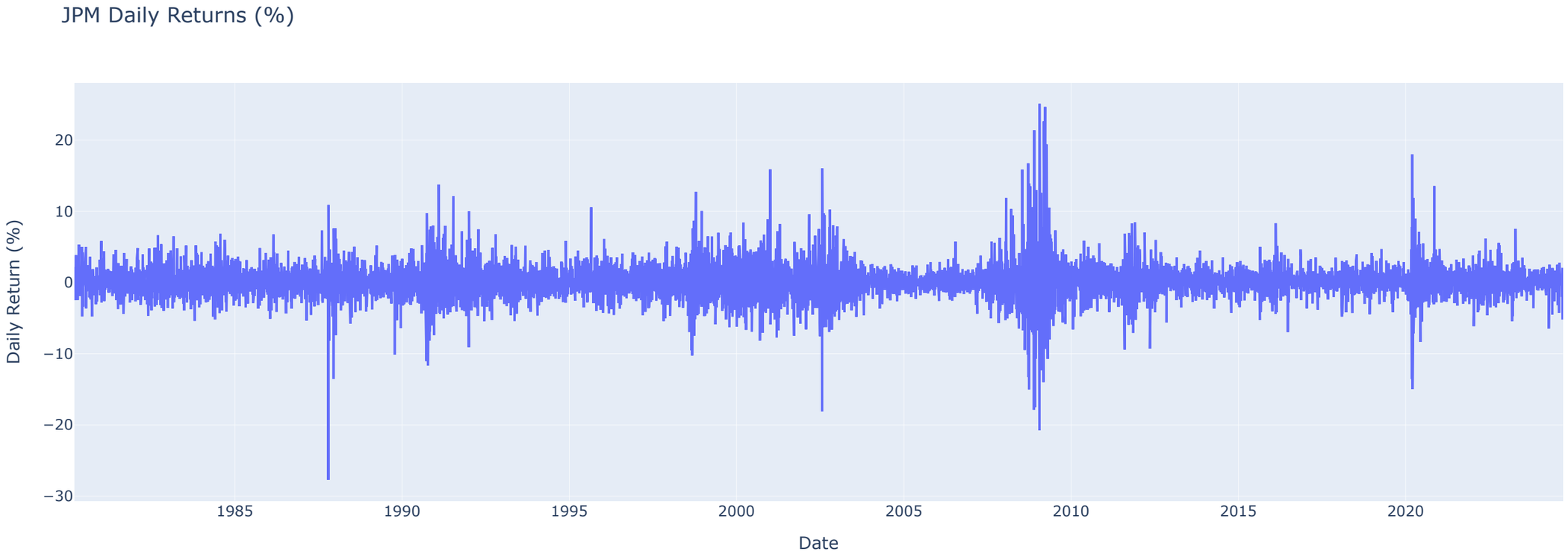
In [40]:
```
CF['returns'] = (CF['Close']/CF['Close'].shift(1)) -1
CF['returns'].hist(bins = 100, label = 'CF', alpha = 0.5, figsize = (15,7))
plt.xlabel('Return')
plt.ylabel('Frequency')
plt.title('Distribution of Returns')
plt.legend()
```

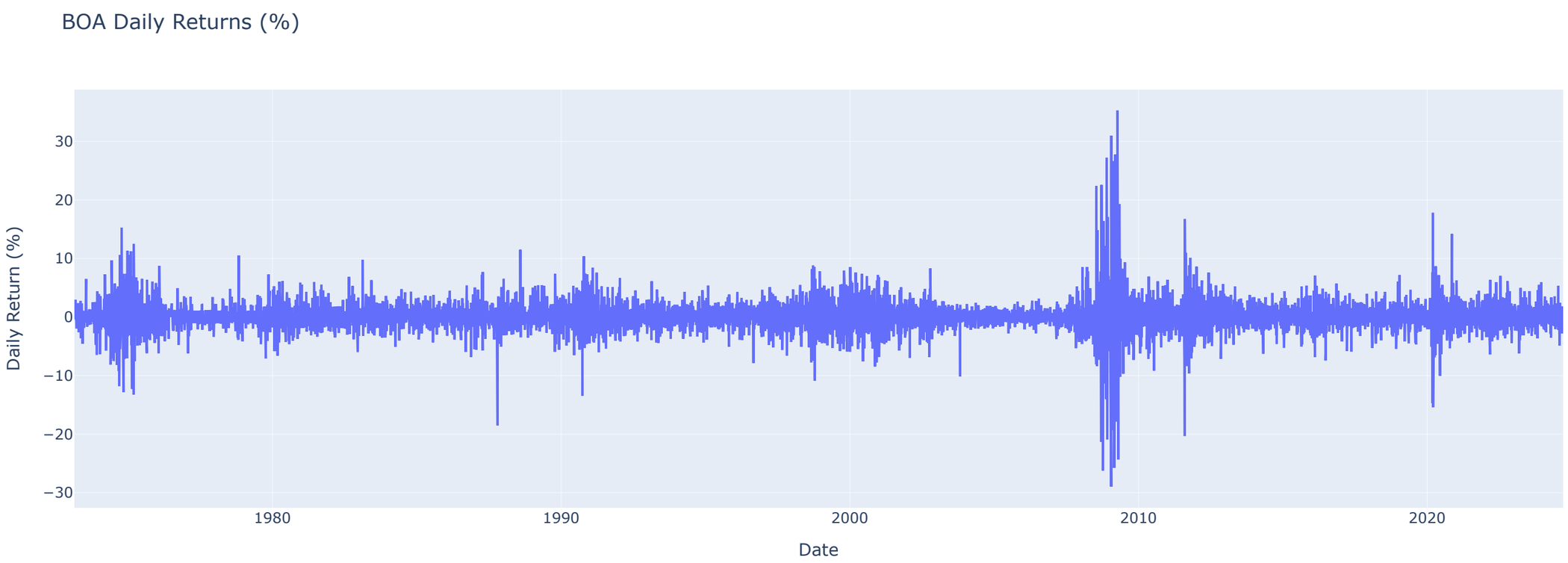
Out[40]:
```
<matplotlib.legend.Legend at 0x26f1a419fd0>
```

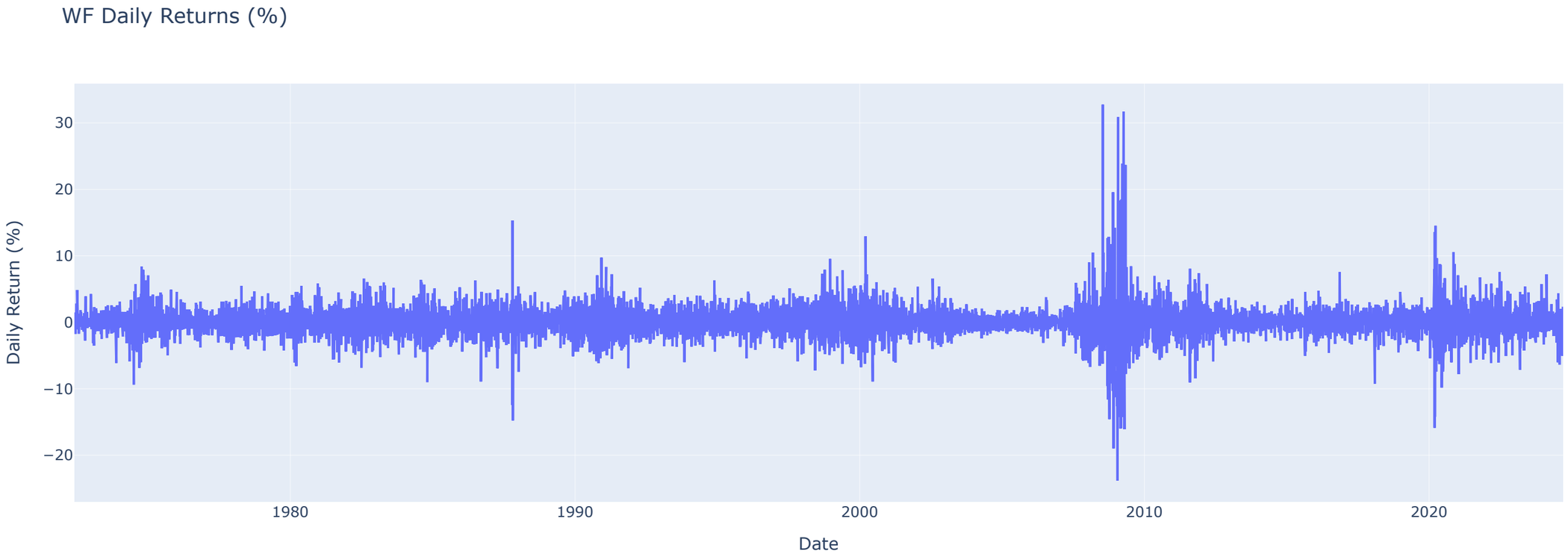


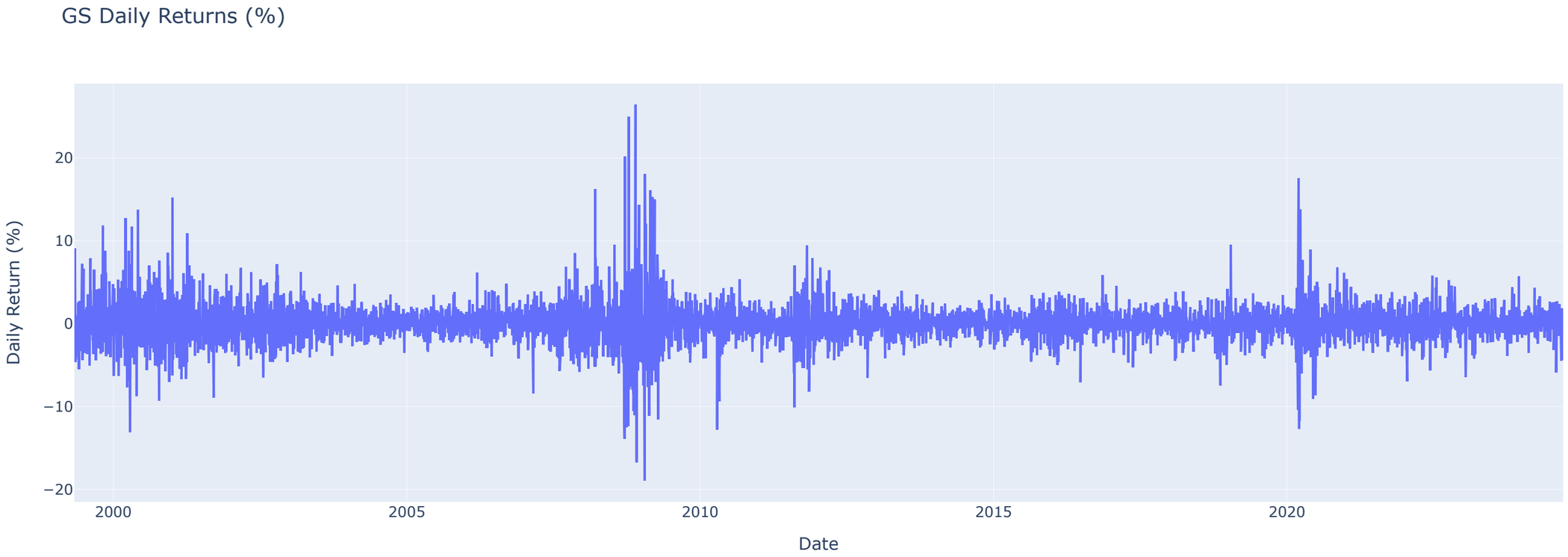
Daily Returns

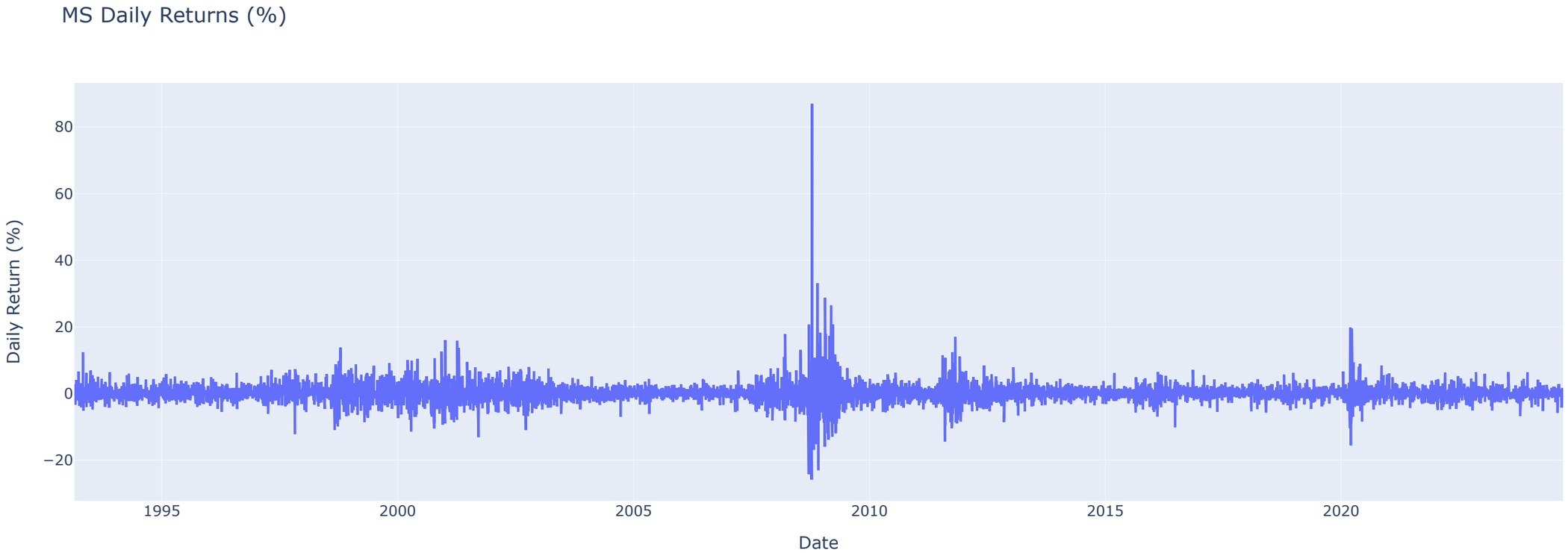
In [42]:
```
import plotly.graph_objs as go

def plot_daily_returns(stock_data, stock):
    stock_data['Daily Return'] = stock_data['Close'].pct_change() * 100

    fig = go.Figure()
    fig.add_trace(go.Scatter(x=stock_data['Date'], y=stock_data['Daily Return'], mode='lines', name='Daily Return (%)'))

    fig.update_layout(title=f'{stock} Daily Returns (%)', xaxis_title='Date', yaxis_title='Daily Return (%)')
    fig.show()


plot_daily_returns(jpmc_stock_data, 'JPM')
plot_daily_returns(boa_stock_data, 'BOA')
plot_daily_returns(wellsf_stock_data, 'WF')
plot_daily_returns(goldms_stock_data, 'GS')
plot_daily_returns(morgans_stock_data, 'MS')
plot_daily_returns(capitalo_stock_data, 'CF')
```

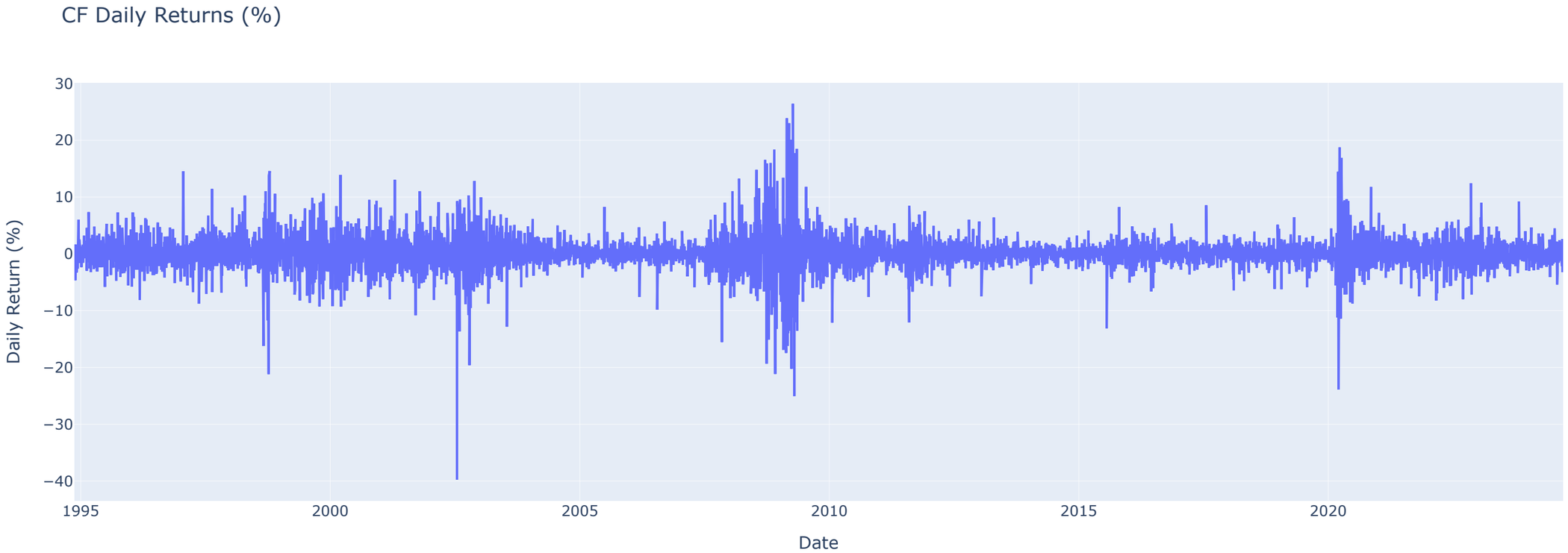

JPM Daily Returns (%)
Daily Return (%)
20
10
0
−10
−20
−30
1985
1990
1995
2000
2005
2010
2015
2020
Date


BOA Daily Returns (%)
Daily Return (%)
30
20
10
0
−10
−20
−30
1980
1990
2000
2010
2020
Date


WF Daily Returns (%)
Daily Return (%)
30
20
10
0
−10
−20
1980
1990
2000
2010
2020
Date

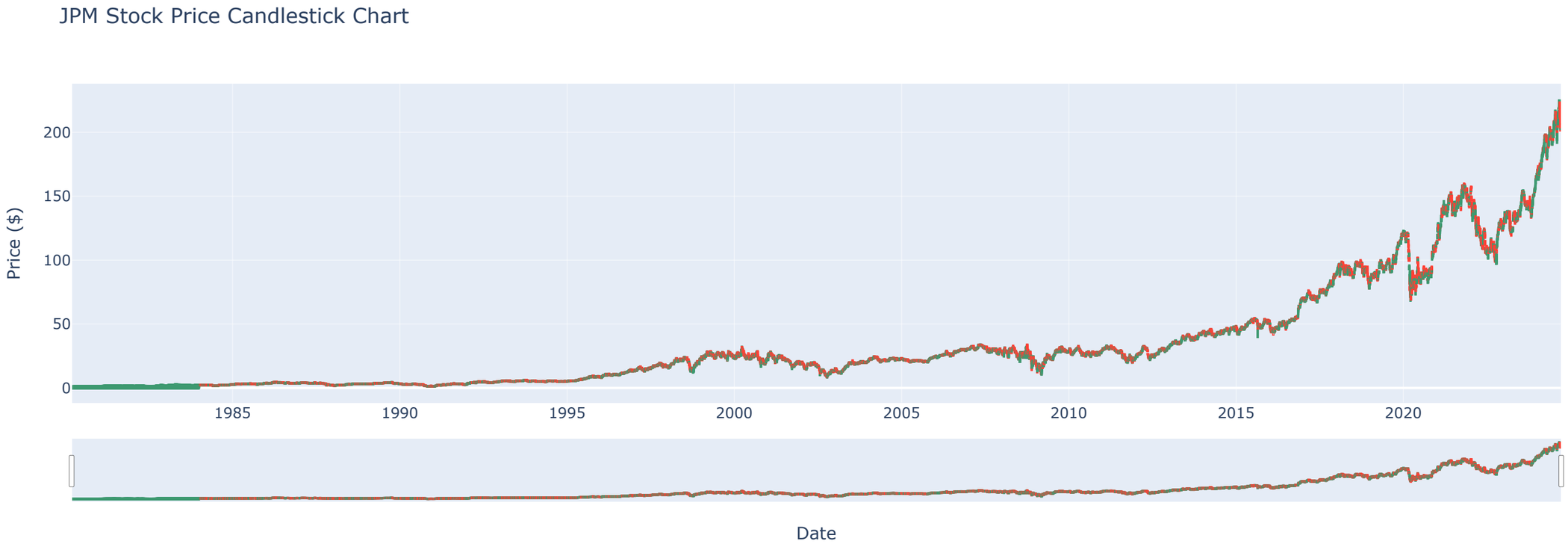

GS Daily Returns (%)
Daily Return (%)
20
10
0
−10
−20
2000
2005
2010
2015
2020
Date

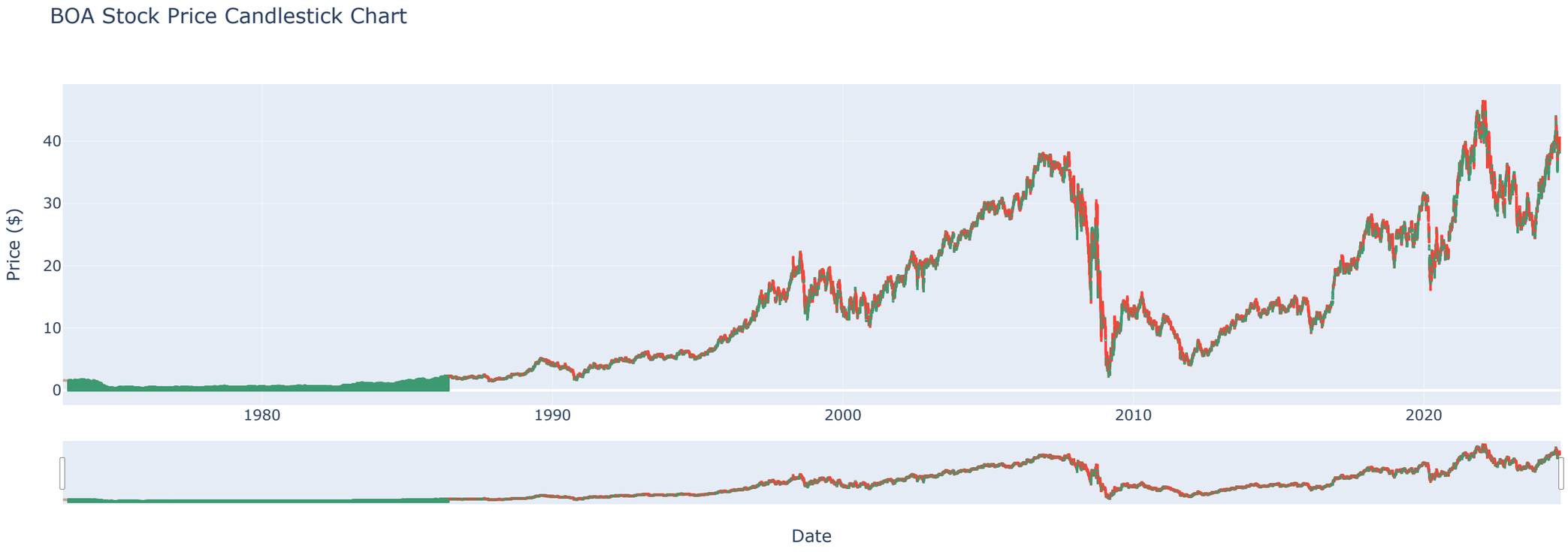

MS Daily Returns (%)
Daily Return (%)
80
60
40
20
0
−20
1995
2000
2005
2010
2015
2020
Date

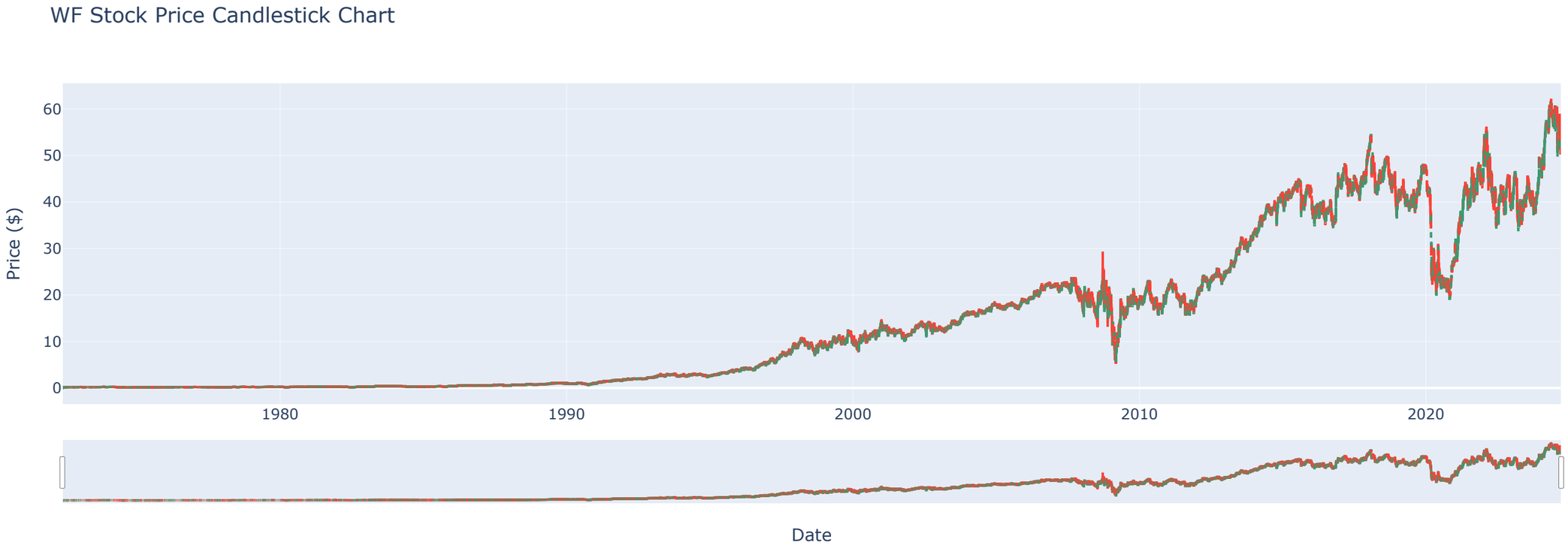


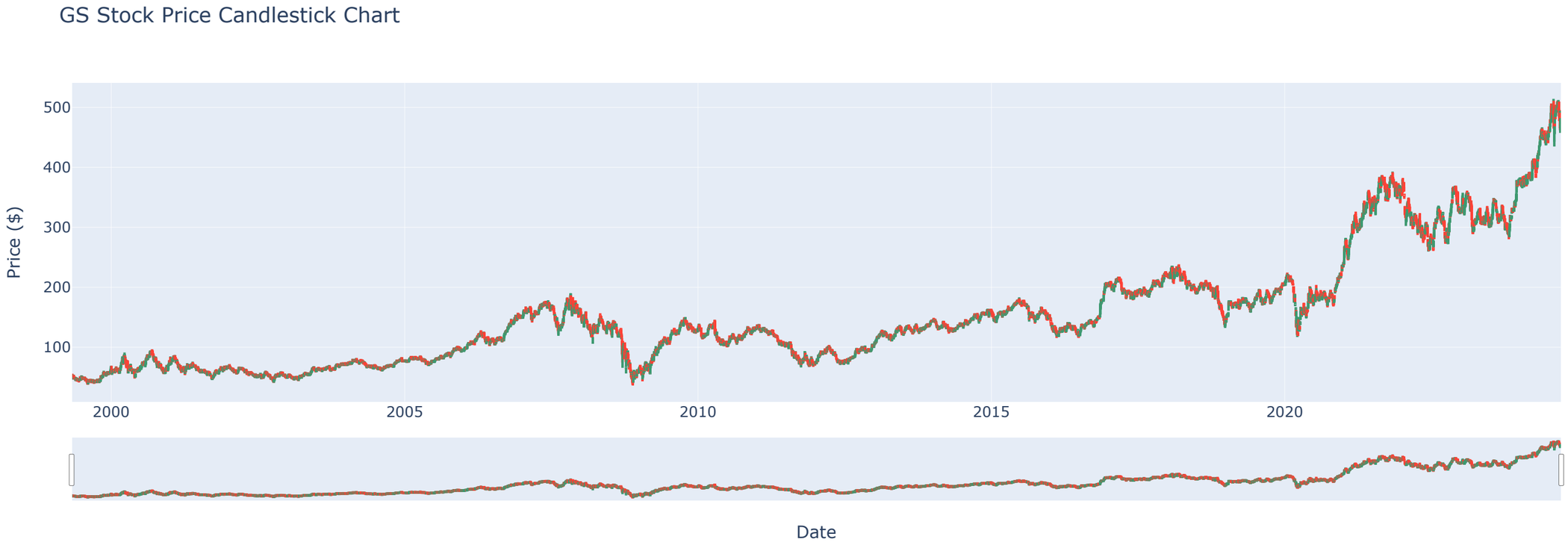
Candlestick Chart

In [44]:
```
import plotly.graph_objs as go

def plot_candlestick(stock_data, stock):
    fig = go.Figure(data=[go.Candlestick(x=stock_data['Date'],
                                         open=stock_data['Open'],
                                         high=stock_data['High'],
                                         low=stock_data['Low'],
                                         close=stock_data['Close'])])

    fig.update_layout(title=f'{stock} Stock Price Candlestick Chart', xaxis_title='Date', yaxis_title='Price ($)')
    fig.show()

plot_candlestick(jpmc_stock_data, 'JPM')
plot_candlestick(boa_stock_data, 'BOA')
plot_candlestick(wellsf_stock_data, 'WF')
plot_candlestick(goldms_stock_data, 'GS')
plot_candlestick(morgans_stock_data, 'MS')
plot_candlestick(capitalo_stock_data, 'CF')
```

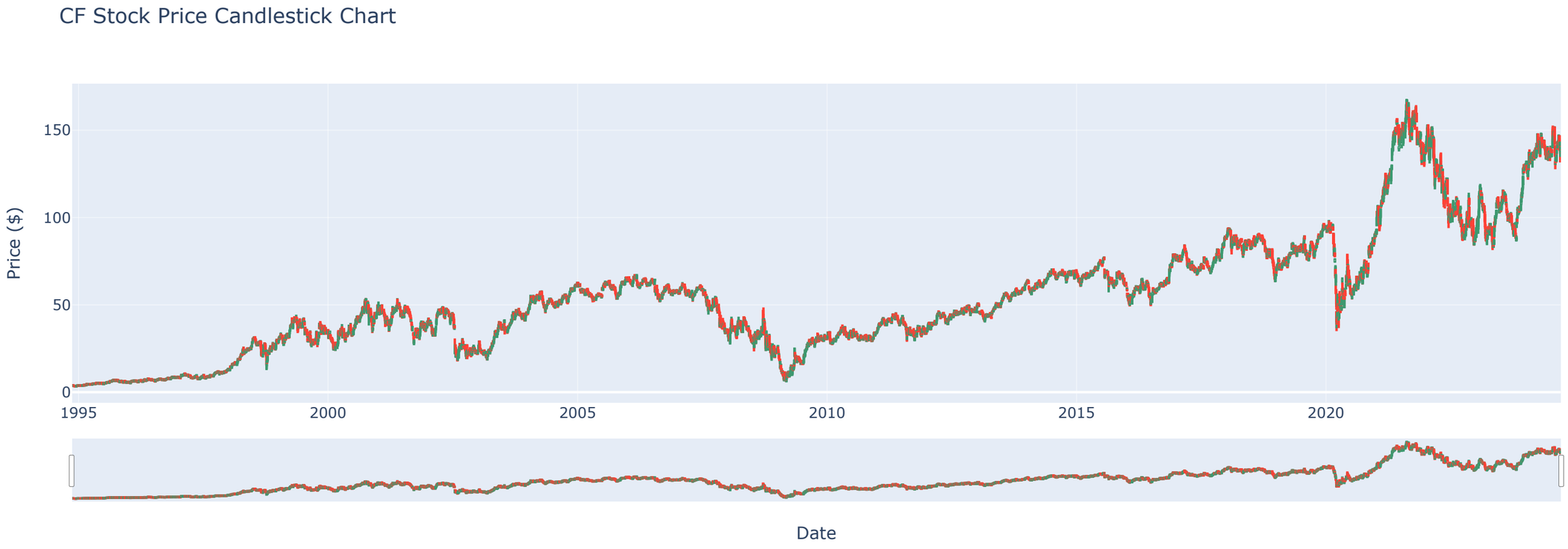

JPM Stock Price Candlestick Chart
Price ($)
200
150
100
50
0
1985
1990
1995
2000
2005
2010
2015
2020
Date


BOA Stock Price Candlestick Chart
Price ($)
0
10
20
30
40
1980
1990
2000
2010
2020
Date


WF Stock Price Candlestick Chart
Price ($)
60
50
40
30
20
10
0
1980
1990
2000
2010
2020
Date


GS Stock Price Candlestick Chart
Price ($)
500
400
300
200
100
2000
2005
2010
2015
2020
Date


MS Stock Price Candlestick Chart
Price ($)
0
20
40
60
80
100
1995
2000
2005
2010
2015
2020
Date


CF Stock Price Candlestick Chart
Price ($)
150
100
50
0
1995
2000
2005
2010
2015
2020
Date