In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv('jpmorgan_data.csv')
df
```

Out[2]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	03-01-2000	49.833332	50.250000	48.083332	48.583332	24.406227	12019200
1	04-01-2000	47.083332	47.458332	46.125000	47.250000	23.870712	11723400
2	05-01-2000	46.833332	48.375000	46.000000	46.958332	23.723372	8714550
3	06-01-2000	46.750000	48.625000	46.500000	47.625000	24.060173	8369250
4	07-01-2000	48.416668	49.000000	47.333332	48.500000	24.502220	6571950
...	...	...	...	...	...	...	...
5858	17-04-2023	139.949997	140.059998	137.660004	139.830002	139.830002	16050500
5859	18-04-2023	140.270004	141.779999	139.029999	141.399994	141.399994	13760100
5860	19-04-2023	141.229996	141.500000	140.399994	141.220001	141.220001	9158100
5861	20-04-2023	139.910004	141.429993	139.839996	140.809998	140.809998	10586200
5862	21-04-2023	139.740005	141.110001	138.779999	140.539993	140.539993	11841800

5863 rows × 7 columns

In [3]:
```
df.head()
```

Out[3]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	03-01-2000	49.833332	50.250000	48.083332	48.583332	24.406227	12019200
1	04-01-2000	47.083332	47.458332	46.125000	47.250000	23.870712	11723400
2	05-01-2000	46.833332	48.375000	46.000000	46.958332	23.723372	8714550
3	06-01-2000	46.750000	48.625000	46.500000	47.625000	24.060173	8369250
4	07-01-2000	48.416668	49.000000	47.333332	48.500000	24.502220	6571950

In [4]:
```
df.tail()
```

Out[4]:

	Date	Open	High	Low	Close	Adj Close	Volume
5858	17-04-2023	139.949997	140.059998	137.660004	139.830002	139.830002	16050500
5859	18-04-2023	140.270004	141.779999	139.029999	141.399994	141.399994	13760100
5860	19-04-2023	141.229996	141.500000	140.399994	141.220001	141.220001	9158100
5861	20-04-2023	139.910004	141.429993	139.839996	140.809998	140.809998	10586200
5862	21-04-2023	139.740005	141.110001	138.779999	140.539993	140.539993	11841800

In [5]:
```
df.columns
```

Out[5]:
```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')
```

In [6]:
```
df.shape
```

Out[6]:
```
(5863, 7)
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

In [8]:
```
df.describe()
```

Out[8]:

	Open	High	Low	Close	Adj Close	Volume
count	5863.000000	5863.000000	5863.000000	5863.000000	5863.000000	5.863000e+03
mean	64.414728	65.132362	63.682219	64.413200	51.506623	2.032749e+07
std	36.405645	36.676471	36.138749	36.404863	38.360588	1.887252e+07
min	15.370000	16.350000	14.960000	15.450000	8.618587	1.347300e+06
25%	38.660000	39.250000	38.164999	38.715000	24.377334	9.768100e+06
50%	47.880001	48.450001	47.320000	47.820000	31.579823	1.381860e+07
75%	89.834999	90.790001	88.295002	89.465000	76.367790	2.361360e+07
max	172.710007	172.960007	170.539993	171.779999	164.015747	2.172942e+08

In [9]:
```
df.duplicated().sum()
```

Out[9]:
```
0
```

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5863 entries, 0 to 5862
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       5863 non-null   object 
 1   Open       5863 non-null   float64
 2   High       5863 non-null   float64
 3   Low        5863 non-null   float64
 4   Close      5863 non-null   float64
 5   Adj Close  5863 non-null   float64
 6   Volume     5863 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 320.8+ KB
```

In [11]:
```
df = df.drop(columns=['Adj Close'])
```

In [12]:
```
sd = df.iloc[0][0]
ed = df.iloc[-1][0]
print('starting Date',sd)
print('Ending Date',ed)
```

```
starting Date 03-01-2000
Ending Date 21-04-2023
```

In [13]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [14]:
```
start_date = pd.Timestamp('03-01-2000')
end_date = pd.Timestamp('21-04-2023')
```

In [15]:
```
df['Date'] = pd.to_datetime(df['Date'])
```

```
df.set_index('Date',inplace=True)
```

In [16]:
```
yearly_data = {}
```

In [17]:
```
for year in range(start_date.year,end_date.year + 1):
    start_of_year = pd.Timestamp(year=year, month=1,day=1)
    end_of_year = pd.Timestamp(year=year,month=12,day=31)
    yearly_df = df.loc[(df.index >=start_of_year) & (df.index <=end_of_year)]
    yearly_df = yearly_df.drop('Volume',axis=1)
    yearly_data[year]= yearly_df
```

In [18]:
```
df_2023 = yearly_data[2023]
```

In [19]:
```
df_2023
```

Out[19]:

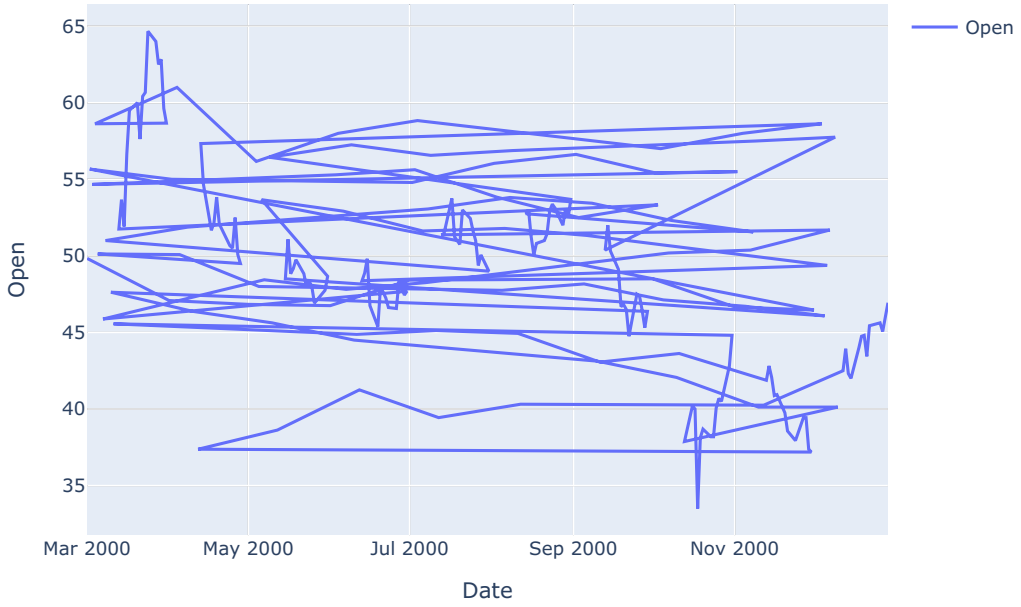
Date	Open	High	Low	Close
2023-03-01	135.240005	136.740005	133.889999	135.119995
2023-04-01	135.990005	137.679993	135.570007	136.380005
2023-05-01	135.660004	135.710007	133.699997	135.350006
2023-06-01	136.130005	138.380005	134.490005	137.940002
2023-09-01	138.600006	138.880005	136.880005	137.369995
...	...	...	...	...
2023-04-17	139.949997	140.059998	137.660004	139.830002
2023-04-18	140.270004	141.779999	139.029999	141.399994
2023-04-19	141.229996	141.500000	140.399994	141.220001
2023-04-20	139.910004	141.429993	139.839996	140.809998
2023-04-21	139.740005	141.110001	138.779999	140.539993

76 rows × 4 columns

In [20]:
```
import plotly.graph_objects as go
fig = go.Figure()
```

In [21]:
```
# Iterate over each year
for year, yearly_df in yearly_data.items():
    # Filter the dataframe based on the specified start and end dates
    filtered_df = yearly_df.loc[(yearly_df.index >= sd) & (yearly_df.index <= ed)]

    # Create a figure object for the current year
    fig = go.Figure()

    # Add a trace for the 'Open' values
    fig.add_trace(go.Scatter(
        x=filtered_df.index,
        y=filtered_df['Open'],
        mode='lines',
        name='Open'
    ))

    # Customize the layout for the current year
    fig.update_layout(
        title=f'Open Values Line Plot for {year}',
        xaxis_title='Date',
        yaxis_title='Open',
        showlegend=True,
    )

    # Show the plot for the current year
    fig.show()
```

Open Values Line Plot for 2000



In [22]:

```python
# Iterate over each year
for year, yearly_df in yearly_data.items():
    # Filter the dataframe based on the specified start and end dates
    filtered_df = yearly_df.loc[(yearly_df.index >= sd) & (yearly_df.index <= ed)]

    # Create a figure object for the current year
    fig = go.Figure()

    # Add a trace for the 'Open' values
    fig.add_trace(go.Scatter(
        x=filtered_df.index,
        y=filtered_df['Open'],
        mode='markers',
        name='Open'
    ))

    # Customize the layout for the current year
    fig.update_layout(
        title=f'Open Values Scatter Plot for {year}',
        xaxis_title='Date',
        yaxis_title='Open',
```

```
        showlegend=True,
    )

    # Show the plot for the current year
    fig.show()
```

In [23]:
```
# Iterate over each year
for year, yearly_df in yearly_data.items():
    # Filter the dataframe based on the specified start and end dates
    filtered_df = yearly_df.loc[(yearly_df.index >= sd) & (yearly_df.index <= ed)]

    # Create a figure object for the current year
    fig = go.Figure()

    # Add a trace for the 'Open' values
    fig.add_trace(go.Scatter(
        x=filtered_df.index,
        y=filtered_df['Close'],
        mode='lines',
        name='Close'
    ))

    # Customize the layout for the current year
    fig.update_layout(
        title=f'Close Values Line Plot for {year}',
        xaxis_title='Date',
        yaxis_title='Close',
```

```
        showlegend=True,
    )

    # Show the plot for the current year
    fig.show()
```

In [24]:
```
#Iterate over each year
for year, yearly_df in yearly_data.items():
    # Filter the dataframe based on the specified start and end dates
    filtered_df = yearly_df.loc[(yearly_df.index >= sd) & (yearly_df.index <= ed)]

    # Create a figure object for the current year
    fig = go.Figure()

    # Add a trace for the 'Open' values
    fig.add_trace(go.Scatter(
        x=filtered_df.index,
        y=filtered_df['Close'],
        mode='markers',
        name='Close'
    ))

    # Customize the layout for the current year
    fig.update_layout(
        title=f'Close Values Line Plot for {year}',
        xaxis_title='Date',
        yaxis_title='Close',
```

```
        showlegend=True,
    )

    # Show the plot for the current year
    fig.show()
```

In [25]:
```
# Iterate over each year
for year, yearly_df in yearly_data.items():
    # Filter the dataframe based on the specified start and end dates
    filtered_df = yearly_df.loc[(yearly_df.index >= sd) & (yearly_df.index <= ed)]

    # Create a figure object for the current year
    fig = go.Figure()

    # Add a trace for the 'Open' values
    fig.add_trace(go.Scatter(
        x=filtered_df.index,
        y=filtered_df['Open'],
        mode='lines',
        name='Open'
    ))

    # Add a trace for the 'Close' values
    fig.add_trace(go.Scatter(
        x=filtered_df.index,
        y=filtered_df['Close'],
        mode='lines',
```

```
        name='Close'
    ))

    # Customize the layout for the current year
    fig.update_layout(
        title=f'Open vs Close Values for {year}',
        xaxis_title='Date',
        yaxis_title='Price',
        showlegend=True,
    )

    # Show the plot for the current year
    fig.show()
```

In [26]:
```
# Iterate over each year
for year, yearly_df in yearly_data.items():
    # Filter the dataframe based on the specified start and end dates
    filtered_df = yearly_df.loc[(yearly_df.index >= sd) & (yearly_df.index <= ed)]

    # Create a figure object for the current year
    fig = go.Figure()

    # Add a trace for the 'Open' values
    fig.add_trace(go.Scatter(
        x=filtered_df.index,
        y=filtered_df['Open'],
        mode='markers',
        name='Open'
    ))

    # Add a trace for the 'Close' values
    fig.add_trace(go.Scatter(
        x=filtered_df.index,
        y=filtered_df['Close'],
        mode='markers',
```

```
        name='Close'
    ))

    # Customize the layout for the current year
    fig.update_layout(
        title=f'Scatter Plot: Open vs Close Values for {year}',
        xaxis_title='Date',
        yaxis_title='Price',
        showlegend=True,
    )

    # Show the plot for the current year
    fig.show()
```

In [27]:

```python
import plotly.express as px

# Iterate over each year
for year, yearly_df in yearly_data.items():
    # Filter the data based on the starting and ending dates
    yearly_df = yearly_df.loc[(yearly_df.index >= start_date) & (yearly_df.index <= end_date)]

    # Reset the index and add 'Date' column
    yearly_df = yearly_df.reset_index()
    yearly_df['Date'] = yearly_df['Date'].dt.strftime('%B')

    # Calculate the mean of 'Open' values for each month
    monthly_mean = yearly_df.groupby('Date')['Open'].mean().reset_index()

    # Create a line plot for the mean 'Open' values for the current year
    fig = px.line(monthly_mean, x='Date', y='Open', title=f'Mean Open Values for Year {year} ({start_date.year}
    fig.show()
```

In [28]:

```
import plotly.express as px

# Iterate over each year
for year, yearly_df in yearly_data.items():
    # Filter the data based on the starting and ending dates
    yearly_df = yearly_df.loc[(yearly_df.index >= start_date) & (yearly_df.index <= end_date)]

    # Reset the index and add 'Date' column
    yearly_df = yearly_df.reset_index()
    yearly_df['Date'] = yearly_df['Date'].dt.strftime('%B')

    # Calculate the mean of 'Open' values for each month
    monthly_mean = yearly_df.groupby('Date')['Open'].mean().reset_index()

    # Create a scatter plot for the mean 'Open' values for the current year
    fig = px.scatter(monthly_mean, x='Date', y='Open', title=f'Mean Open Values for Year {year} ({start_date.ye
    fig.show()
```

In [29]:

```
import plotly.express as px

# Iterate over each year
for year, yearly_df in yearly_data.items():
    # Filter the data based on the starting and ending dates
    yearly_df = yearly_df.loc[(yearly_df.index >= start_date) & (yearly_df.index <= end_date)]

    # Reset the index and add 'Date' column
    yearly_df = yearly_df.reset_index()
    yearly_df['Date'] = yearly_df['Date'].dt.strftime('%B')

    # Calculate the mean of 'Close' values for each month
    monthly_mean = yearly_df.groupby('Date')['Close'].mean().reset_index()

    # Create a line plot for the mean 'Close' values for the current year
    fig = px.line(monthly_mean, x='Date', y='Close', title=f'Mean Close Values for Year {year} ({start_date.yea
    fig.show()
```

In [30]:

```
import plotly.express as px

# Iterate over each year
for year, yearly_df in yearly_data.items():
    # Filter the data based on the starting and ending dates
    yearly_df = yearly_df.loc[(yearly_df.index >= start_date) & (yearly_df.index <= end_date)]

    # Reset the index and add 'Date' column
    yearly_df = yearly_df.reset_index()
    yearly_df['Date'] = yearly_df['Date'].dt.strftime('%B')

    # Calculate the mean of 'Close' values for each month
    monthly_mean = yearly_df.groupby('Date')['Close'].mean().reset_index()

    # Create a scatter plot for the mean 'Close' values for the current year
    fig = px.scatter(monthly_mean, x='Date', y='Close', title=f'Mean Close Values for Year {year} ({start_date.
    fig.show()
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: