```python
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from warnings import filterwarnings
filterwarnings('ignore')
```

```python
In [2]: from os import listdir
```

Data sources: https://www.marketwatch.com/investing/stock/dxc/download-data

```python
In [3]: path = r"C:\Users\rkuppi\OneDrive - DXC Production\Desktop\DataSets\TimeseriesDataAnalysis"
files = list(filter(lambda x: '.csv' in x, listdir(path)))
```

```python
In [4]: files
```

```
Out[4]: ['AAPL_data.csv',
 'ADP_data.csv',
 'AMD_data.csv',
 'AMZN_data.csv',
 'DXC_data.csv',
 'GOOGL_data.csv',
 'IBM_data.csv',
 'NVDA_data.csv']
```

```python
In [5]: Stock_data = pd.DataFrame()
```

```python
In [6]: for file in files:
    df = pd.read_csv(file)
    Stock_data = pd.concat([Stock_data, df])
```

```python
In [7]: Stock_data.head()
```

Out[7]:

	date	open	high	low	close	volume	Name
0	2013-02-08	67.7142	68.4014	66.8928	67.8542	158168416	AAPL
1	2013-02-11	68.0714	69.2771	67.6071	68.5614	129029425	AAPL
2	2013-02-12	68.5014	68.9114	66.8205	66.8428	151829363	AAPL
3	2013-02-13	66.7442	67.6628	66.1742	66.7156	118721995	AAPL
4	2013-02-14	66.3599	67.3771	66.2885	66.6556	88809154	AAPL

```python
In [8]: Stock_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9028 entries, 0 to 1258
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   date    9028 non-null   object
 1   open    9028 non-null   float64
 2   high    9028 non-null   float64
 3   low     9028 non-null   float64
 4   close   9028 non-null   float64
 5   volume  9028 non-null   int64
 6   Name    9028 non-null   object
dtypes: float64(4), int64(1), object(2)
memory usage: 564.2+ KB
```

```python
In [9]: Stock_data.shape
```

```
Out[9]: (9028, 7)
```

We can see that data column dtype is Object, converting it into pandas datetime object

```python
In [10]: Stock_data['date'] = pd.to_datetime(Stock_data['date'])
```

```python
In [11]: Stock_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9028 entries, 0 to 1258
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   date    9028 non-null   datetime64[ns]
 1   open    9028 non-null   float64
 2   high    9028 non-null   float64
 3   low     9028 non-null   float64
 4   close   9028 non-null   float64
 5   volume  9028 non-null   int64
 6   Name    9028 non-null   object
dtypes: datetime64[ns](1), float64(4), int64(1), object(1)
memory usage: 564.2+ KB
```

```python
In [12]: Stock_data.head()
```

Out[12]:

	date	open	high	low	close	volume	Name
0	2013-02-08	67.7142	68.4014	66.8928	67.8542	158168416	AAPL
1	2013-02-11	68.0714	69.2771	67.6071	68.5614	129029425	AAPL
2	2013-02-12	68.5014	68.9114	66.8205	66.8428	151829363	AAPL
3	2013-02-13	66.7442	67.6628	66.1742	66.7156	118721995	AAPL
4	2013-02-14	66.3599	67.3771	66.2885	66.6556	88809154	AAPL

```python
In [13]: Stock_data['Year'] = pd.DatetimeIndex(Stock_data['date']).year
Stock_data['Month'] = pd.DatetimeIndex(Stock_data['date']).month
```

```python
In [14]: Stock_data['Close - Open'] = Stock_data['close'] - Stock_data['open']
```

```python
In [15]: Stock_data.head()
```

Out[15]:

	date	open	high	low	close	volume	Name	Year	Month	Close - Open
0	2013-02-08	67.7142	68.4014	66.8928	67.8542	158168416	AAPL	2013	2	0.1400
1	2013-02-11	68.0714	69.2771	67.6071	68.5614	129029425	AAPL	2013	2	0.4900
2	2013-02-12	68.5014	68.9114	66.8205	66.8428	151829363	AAPL	2013	2	-1.6586
3	2013-02-13	66.7442	67.6628	66.1742	66.7156	118721995	AAPL	2013	2	-0.0286
4	2013-02-14	66.3599	67.3771	66.2885	66.6556	88809154	AAPL	2013	2	0.2957

```python
In [16]: Stock_data.Year.value_counts()
```

```
Out[16]: 2017    1946
2014    1764
2015    1764
2016    1764
2013    1582
2018     208
Name: Year, dtype: int64
```

```python
In [17]: AAPL = Stock_data.groupby(by = 'Name').get_group('AAPL').reset_index(drop = True)
```

```python
In [18]: AAPL.head()
```

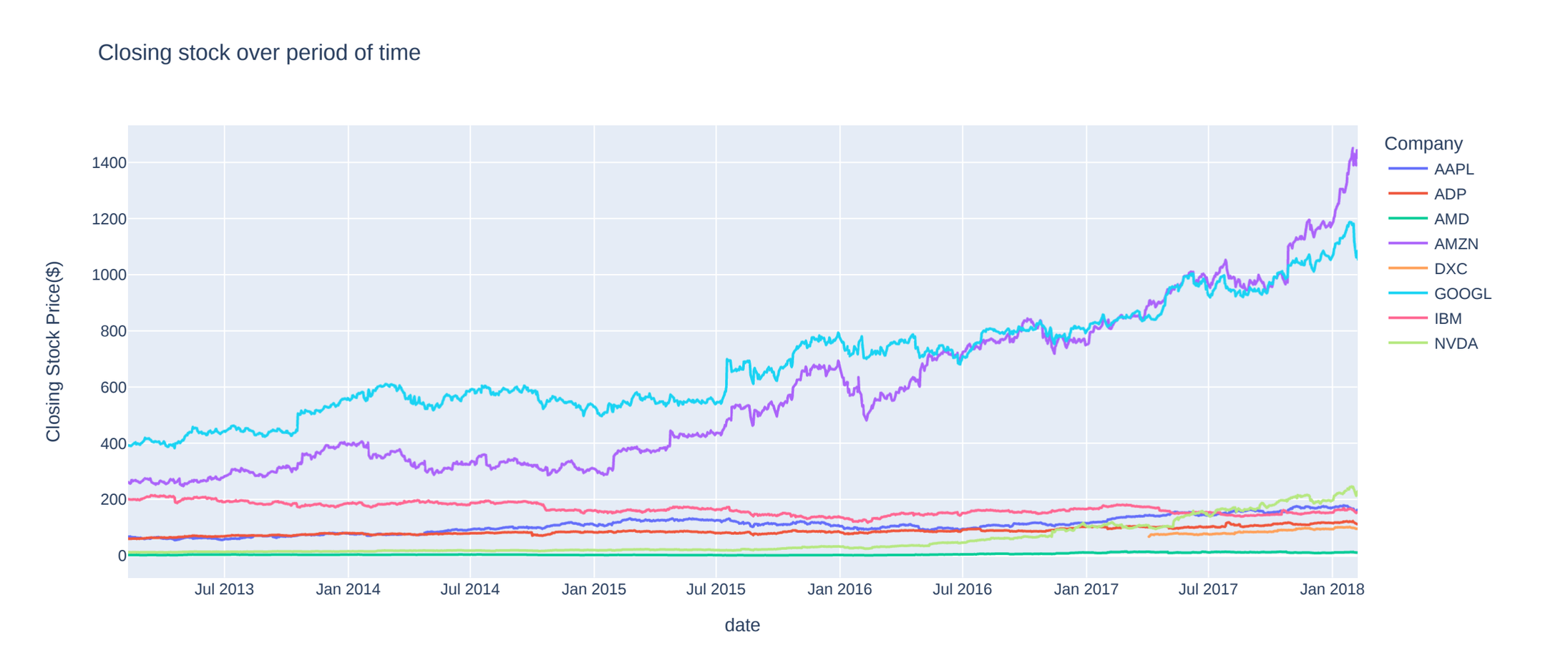
Out[18]:

	date	open	high	low	close	volume	Name	Year	Month	Close - Open
0	2013-02-08	67.7142	68.4014	66.8928	67.8542	158168416	AAPL	2013	2	0.1400
1	2013-02-11	68.0714	69.2771	67.6071	68.5614	129029425	AAPL	2013	2	0.4900
2	2013-02-12	68.5014	68.9114	66.8205	66.8428	151829363	AAPL	2013	2	-1.6586
3	2013-02-13	66.7442	67.6628	66.1742	66.7156	118721995	AAPL	2013	2	-0.0286
4	2013-02-14	66.3599	67.3771	66.2885	66.6556	88809154	AAPL	2013	2	0.2957

```python
In [19]: ADP = Stock_data.groupby(by = 'Name').get_group('ADP').reset_index(drop = True)
AMD = Stock_data.groupby(by = 'Name').get_group('AMD').reset_index(drop = True)
AMZN = Stock_data.groupby(by = 'Name').get_group('AMZN').reset_index(drop = True)
GOOGL = Stock_data.groupby(by = 'Name').get_group('GOOGL').reset_index(drop = True)
IBM = Stock_data.groupby(by = 'Name').get_group('IBM').reset_index(drop = True)
NVDA = Stock_data.groupby(by = 'Name').get_group('NVDA').reset_index(drop = True)
DXC = Stock_data.groupby(by = 'Name').get_group('DXC').reset_index(drop = True)
```

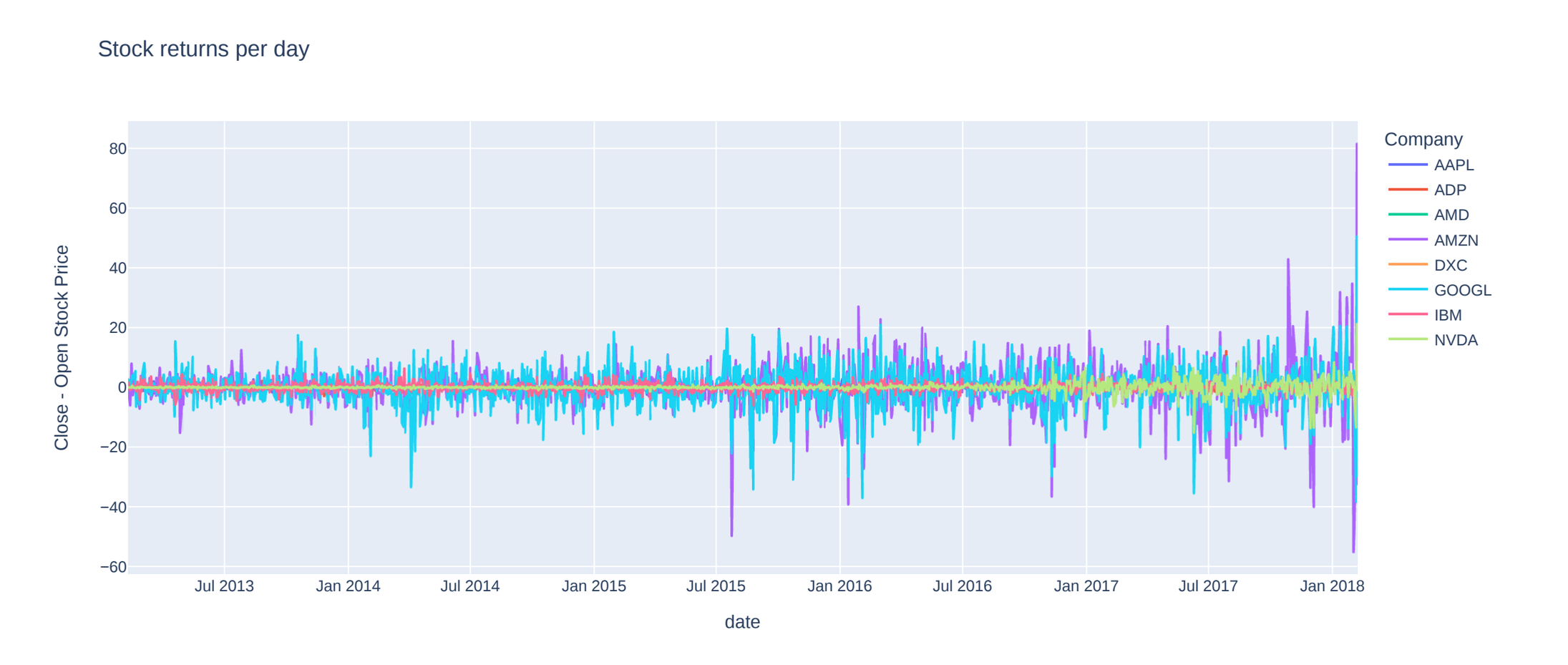
Closing price of trend

```python
In [20]: import plotly.express as px
fig = px.line(Stock_data, x="date", y="close", color='Name', labels=dict(close="Closing Stock Price($)", Name="Company"), title = "Closing stock over period of time")
fig.show()
```



Daily Price Chnage

```python
In [21]: fig = px.line(data_frame= Stock_data.loc[:, ['date','Name', 'Close - Open']], x = 'date', y = 'Close - Open', color='Name', labels={"Close - Open":'Close - Open Stock Price', "Name":"Company"}, title='S
fig.show()
```




```python
In [22]: Stock_data.groupby(by = ['Name'])[['Close - Open', 'close', 'open']].agg(['sum','min', 'max', 'std']).T
```

Out[22]:

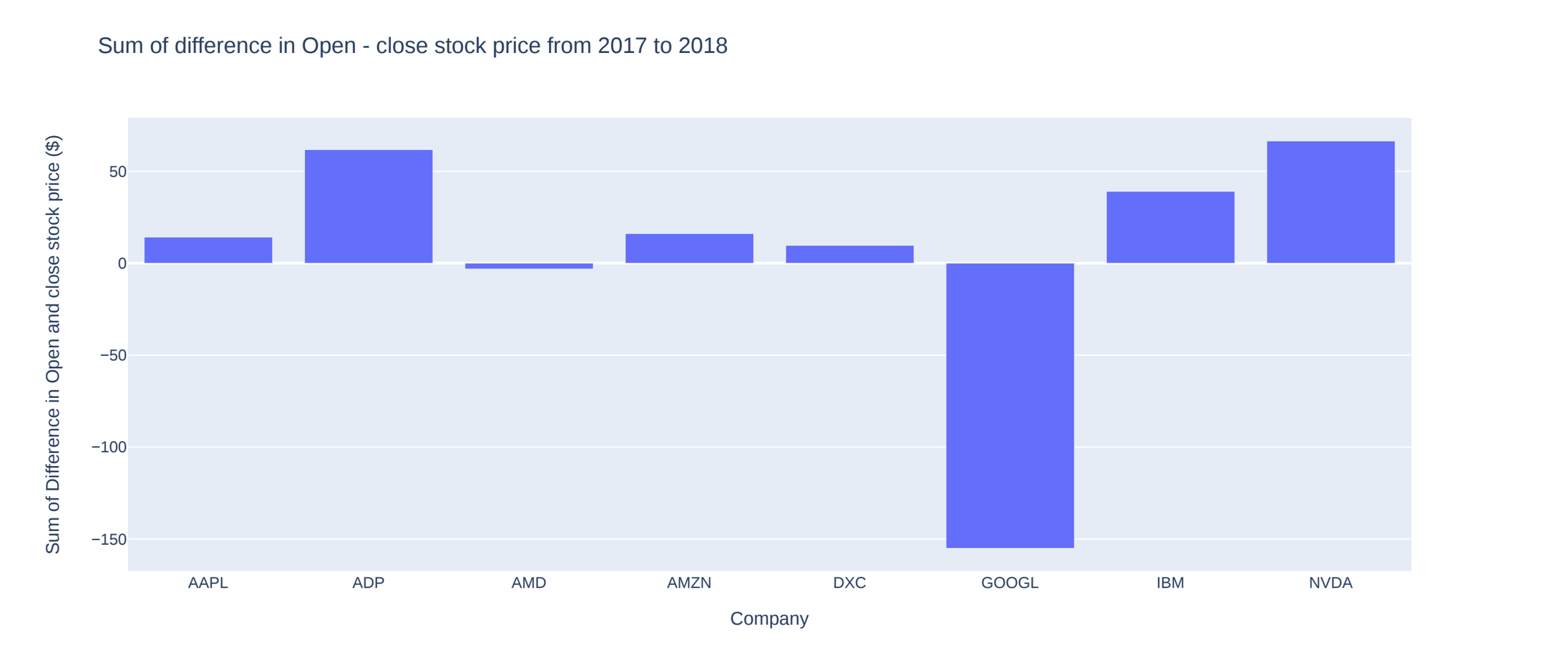
	Name	AAPL	ADP	AMD	AMZN	DXC	GOOGL	IBM	NVDA
Close - Open	sum	14.188400	61.791000	-3.087900	16.085900	9.665000	-155.100600	39.098800	66.503500
	min	-7.370000	-6.010000	-1.410000	-55.000000	-2.610000	-38.220000	-5.980000	-15.140000
	max	8.250000	12.130000	1.170000	81.380000	3.300000	50.450000	6.280000	21.180000
	std	1.283588	0.871989	0.190449	8.311859	0.869715	7.239046	1.438456	1.746502
close	sum	137314.973400	109544.836000	7052.745000	726291.971000	18475.470000	858932.413600	210582.765000	70969.034500
	min	55.789900	60.270000	1.620000	248.230000	67.950000	383.340000	117.850000	12.126000
	max	179.260000	123.630000	15.200000	1450.890000	102.500000	1187.560000	215.800000	246.850000
	std	30.556812	13.280746	3.971779	282.500395	8.807176	187.573892	20.207108	59.624642
open	sum	137300.785000	109483.045000	7055.832900	726275.885100	18465.805000	859087.514200	210543.666200	70902.531000
	min	55.424200	60.240000	1.620000	248.940000	68.580000	384.964600	118.460000	12.070000
	max	179.370000	122.700000	15.450000	1477.390000	102.440000	1188.000000	215.380000	245.770000
	std	30.549220	13.274823	3.979483	282.500019	8.885593	187.409986	20.184908	59.590152

```python
In [23]: Stock_data.groupby(by = ['Name'])['Close - Open'].agg(['sum','min', 'max', 'std'])
```

Out[23]:

Name	sum	min	max	std
AAPL	14.1884	-7.37	8.25	1.283588
ADP	61.7910	-6.01	12.13	0.871989
AMD	-3.0879	-1.41	1.17	0.190449
AMZN	16.0859	-55.00	81.38	8.311859
DXC	9.6650	-2.61	3.30	0.869715
GOOGL	-155.1006	-38.22	50.45	7.239046
IBM	39.0988	-5.98	6.28	1.438456
NVDA	66.5035	-15.14	21.18	1.746502

```python
In [24]: fig = px.bar(data_frame=Stock_data.groupby(by = ['Name'])['Close - Open'].agg(['sum','min', 'max', 'std']),
             x = Stock_data.groupby(by = ['Name'])['Close - Open'].agg(['sum','min', 'max', 'std']).index.to_list(), y = 'sum',
             labels={'x': 'Company', 'sum': "Sum of Difference in Open and close stock price ($)"},
             title="Sum of difference in Open - close stock price from 2017 to 2018")
fig.show()
```



By this we can understand that sum of open -close prices for Google and AMD are < 0;

comparision of stock price among companies

```python
In [25]: Closing_stock = pd.DataFrame()
```

```python
In [26]: Closing_stock['AAPL'] = AAPL['close']
Closing_stock['ADP'] = ADP['close']
Closing_stock['AMD'] = AMD['close']
Closing_stock['AMZN'] = AMZN['close']
Closing_stock['DXC'] = DXC['close']
Closing_stock['GOOGL'] = GOOGL['close']
Closing_stock['IBM'] = IBM['close']
Closing_stock['NVDA'] = NVDA['close']
```

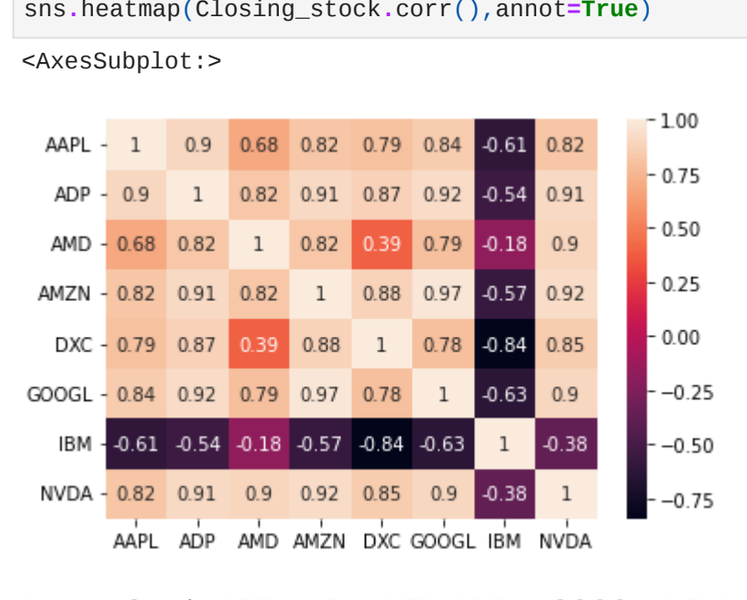
```python
In [27]: Closing_stock.head()
```

Out[27]:

	AAPL	ADP	AMD	AMZN	DXC	GOOGL	IBM	NVDA
0	67.8542	60.925	2.59	261.95	67.95	393.0777	201.68	12.37
1	68.5614	60.340	2.67	257.21	69.40	391.6012	200.16	12.51
2	66.8428	60.360	2.77	258.70	71.79	390.7403	200.04	12.43
3	66.7156	60.450	2.75	269.47	74.68	391.8214	200.09	12.37
4	66.6556	60.740	2.75	269.24	75.87	394.3039	199.65	12.73

```python
In [28]: sns.heatmap(Closing_stock.corr(),annot=True)
```

```
Out[28]: <AxesSubplot:>
```



Amazon-Google, ADP- AAPL, AMZN -NVDIA, GOOGL-NVDIA, ADP -NVDIA, ADP-GOOGLE are highly correlated with each other

Correlation of closing price between companies

Trading volumes

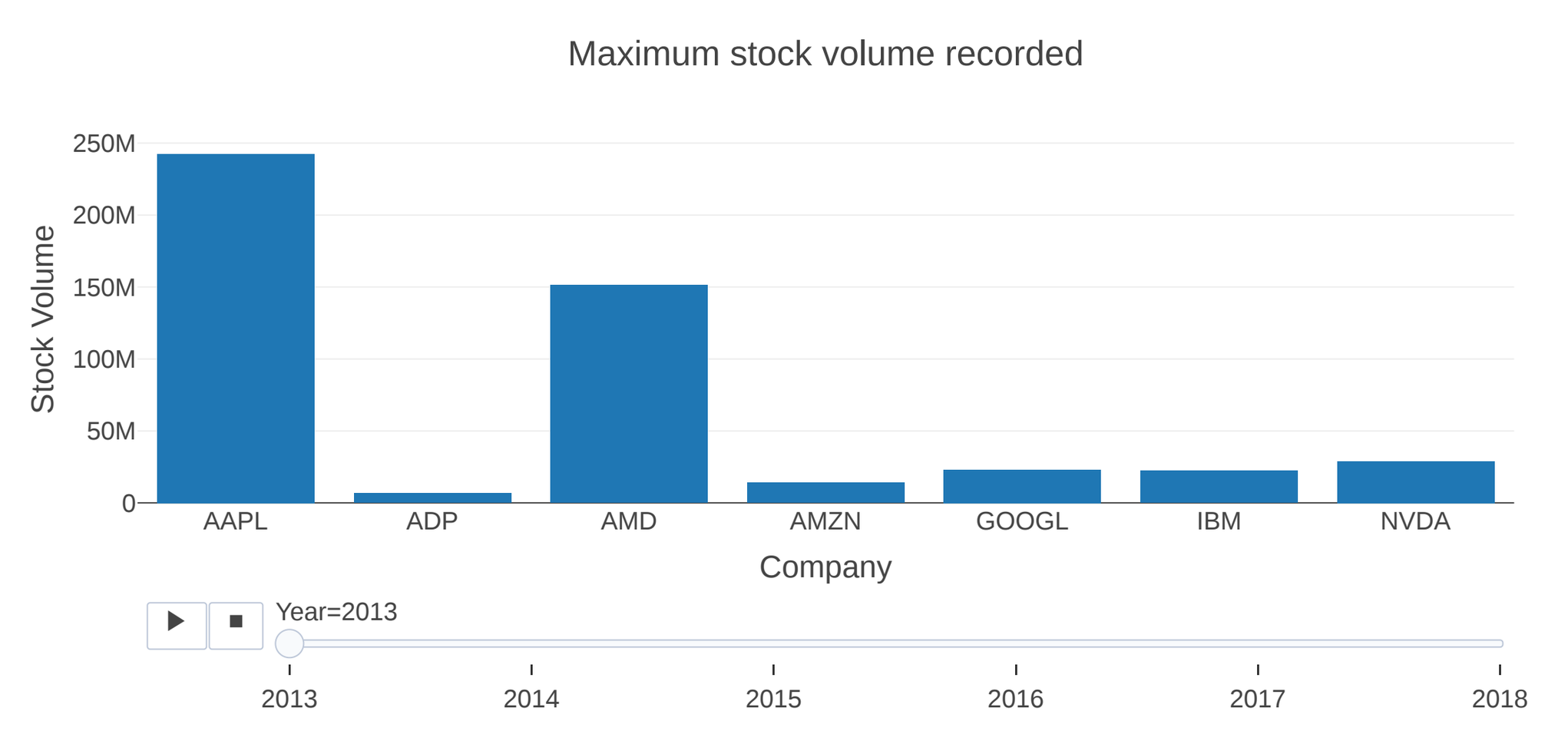
Maximun volume, opening, closeing and difference in open and close for each year and each company

```python
In [29]: Stock_data.groupby(by = ['Year', 'Name']).max().loc[:, ['volume', 'open', 'close', 'Close - Open']].reset_index()
```

Out[29]:

	Year	Name	volume	open	close	Close - Open
0	2013	AAPL	242387530	81.8071	81.4413	2.6614
1	2013	ADP	6780229	83.1800	81.0800	2.1400
2	2013	AMD	151454259	4.4800	4.6400	0.4100
3	2013	AMZN	14029220	404.6500	404.3900	12.4600
4	2013	GOOGL	23145371	560.7302	560.9154	17.4314
5	2013	IBM	22359380	215.3800	215.8000	4.6300
6	2013	NVDA	28989638	16.2800	16.2200	0.6400
7	2014	AAPL	266833581	119.2700	119.0000	3.9300
8	2014	ADP	4637467	86.4500	86.0700	2.0650
9	2014	AMD	156257503	4.6300	4.6600	0.2000
10	2014	AMZN	19805911	408.0000	407.0500	15.4700
11	2014	GOOGL	11128460	614.0134	610.6951	12.4600
12	2014	IBM	23416511	198.0500	197.7700	6.2800
13	2014	NVDA	22572405	21.1100	21.1400	0.7800
14	2015	AAPL	162206292	134.4550	133.0000	8.2500
15	2015	ADP	6545621	90.2300	90.5300	2.1800
16	2015	AMD	97054249	3.2000	3.3100	0.4300
17	2015	AMZN	23856060	691.8900	693.9700	19.5500
18	2015	GOOGL	12858136	793.9600	793.9600	19.6200
19	2015	IBM	16025591	174.4700	174.4000	5.0100
20	2015	NVDA	35131171	33.7400	33.7500	1.3200
21	2016	AAPL	133369674	118.1800	118.2500	3.2900
22	2016	ADP	5682594	103.5300	103.4500	3.0100
23	2016	AMD	170828036	12.2800	12.0700	0.8000
24	2016	AMZN	14677550	845.7900	844.3600	26.9800
25	2016	GOOGL	7039948	838.5000	835.7400	20.8700
26	2016	IBM	16157796	168.9700	168.5100	4.1900
27	2016	NVDA	57394116	119.6000	117.3200	8.4600
28	2017	AAPL	111985040	175.1100	176.4200	3.9800
29	2017	ADP	29837591	119.2600	118.9100	12.1300
30	2017	AMD	268336455	15.4500	15.2000	1.1700
31	2017	AMZN	16565021	1204.8800	1195.8300	42.8100
32	2017	DXC	34496800	98.8100	99.0200	3.3000
33	2017	GOOGL	5933602	1083.0200	1085.0900	17.1600
34	2017	IBM	30490192	182.0000	181.9500	4.5400
35	2017	NVDA	92323196	217.3100	216.9600	8.7300
36	2018	AAPL	86593825	179.3700	179.2600	8.2000
37	2018	ADP	3415867	122.7000	123.6300	3.3600
38	2018	AMD	154066661	13.6200	13.7400	0.6100
39	2018	AMZN	11494985	1477.3900	1450.8900	81.3800
40	2018	DXC	2246075	102.4400	102.5000	1.8000
41	2018	GOOGL	5892122	1188.0000	1187.5600	50.4500
42	2018	IBM	21172488	170.0000	169.1200	5.0500
43	2018	NVDA	29130140	245.7700	246.8500	21.1800

```python
In [30]: pl = px.bar(data_frame=Stock_data.groupby(by = ['Year', 'Name']).max().loc[:, ['volume', 'open', 'close']].reset_index(),
            x = 'Name', y = 'volume', animation_frame='Year', template='presentation', title='Maximum stock volume recorded',
            labels={
                'Name': 'Company', 'volume': 'Stock Volume'
            })
pl.show()
```



Simimarly we can plot the maximum open price recorded, maximum close price recorded and maximum Close -Open recorded for each year and each company

Same thing goes with minimuim volume, close, open and close - open recorded, with small change we, replace max() with min()

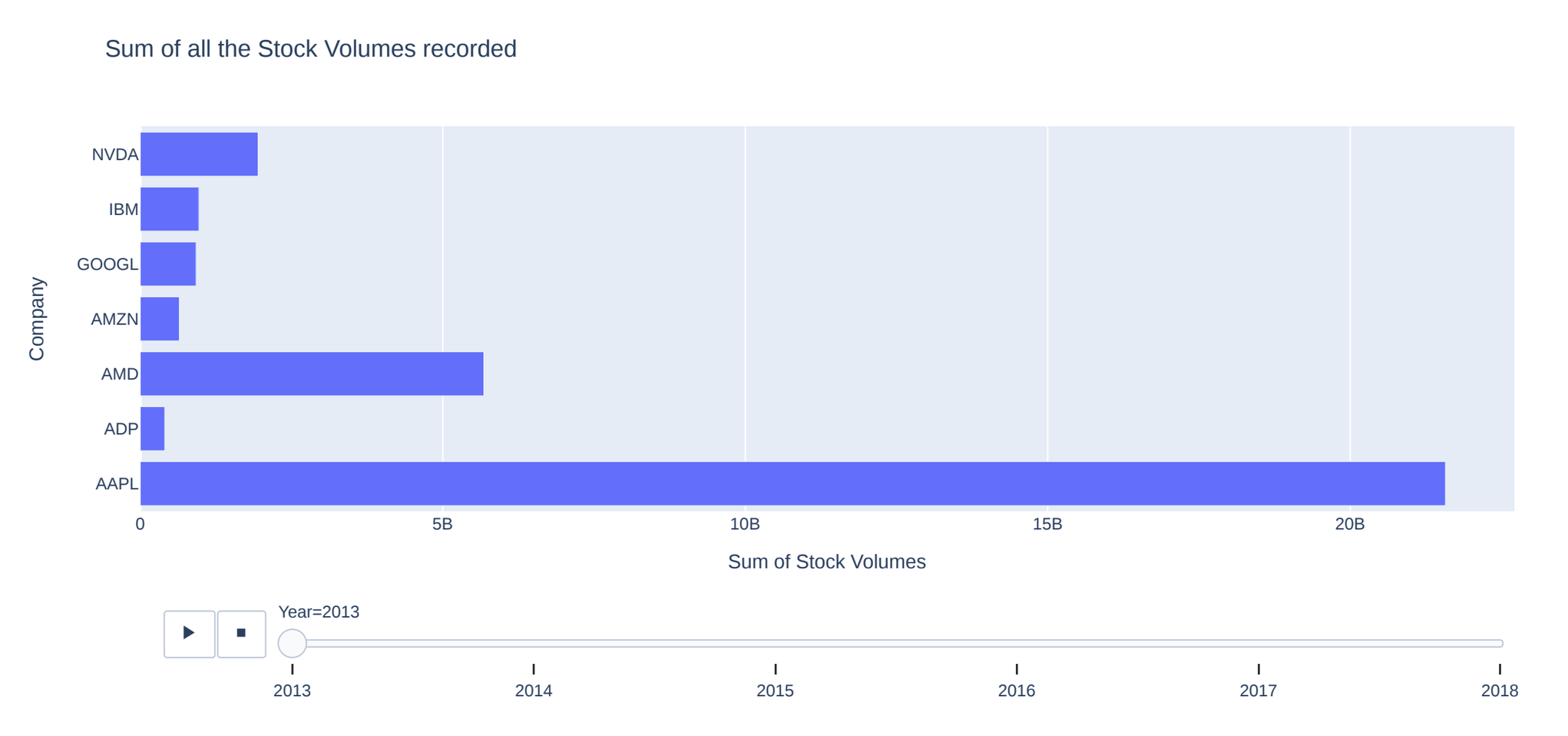
Sum of volumes, open stock, close, open - close per each year and each company

```python
In [31]: Stock_data.groupby(by = ['Year', 'Name']).sum().loc[:, ['volume', 'open', 'close', 'Close - Open']].reset_index()
```

Out[31]:

	Year	Name	volume	open	close	Close - Open
0	2013	AAPL	21574452606	15207.0891	15195.7517	-11.3374
1	2013	ADP	403059249	15998.0950	16012.9610	14.8660
2	2013	AMD	5677217586	783.0849	782.7700	-0.3149
3	2013	AMZN	643035218	68088.1101	68153.6700	65.5599
4	2013	GOOGL	918998892	101828.0329	101885.1835	57.1506
5	2013	IBM	965295204	43754.3052	43770.2600	15.9548
6	2013	NVDA	1945396984	3250.2250	3254.3990	4.1740
7	2014	AAPL	15934013092	23239.3734	23250.6617	11.2883
8	2014	ADP	432530663	20089.8000	20081.1400	-8.6600
9	2014	AMD	5983316677	924.3358	922.8400	-1.4958
10	2014	AMZN	1030203688	83865.2050	83802.8460	-62.3590
11	2014	GOOGL	672910845	143377.7938	143166.7951	-210.9987
12	2014	IBM	1140815080	45929.9600	45937.9800	8.0200
13	2014	NVDA	1789047591	4672.8500	4673.7955	0.9455
14	2015	AAPL	13068421452	30282.6400	30250.0450	-32.5950
15	2015	ADP	475937123	21252.4200	21262.0200	9.6000
16	2015	AMD	3444500704	586.5750	587.6050	1.0300
17	2015	AMZN	957259202	120487.8100	120490.8250	3.0150
18	2015	GOOGL	546889106	156256.4025	156237.0350	-19.3675
19	2015	IBM	1105809640	39154.5100	39148.6900	-5.8200
20	2015	NVDA	1954756332	5967.4900	5976.8900	9.4000
21	2016	AAPL	9686251480	26335.9400	26360.2100	24.2700
22	2016	ADP	479241479	22415.4100	22440.2700	24.8600
23	2016	AMD	7739018950	1320.8600	1324.2050	3.3450
24	2016	AMZN	1038805155	176338.6600	176279.8300	-58.8300
25	2016	GOOGL	496268875	192441.9050	192329.9800	-111.9250
26	2016	IBM	1019307452	37886.3150	37929.7800	43.4650
27	2016	NVDA	2790035685	13514.9700	13547.8200	32.8500
28	2017	AAPL	6797072145	37771.1225	37796.8550	25.7325
29	2017	ADP	702756127	26629.9900	26651.2950	21.3050
30	2017	AMD	16355699307	3121.5622	3115.3150	-6.2472
31	2017	AMZN	881017394	243031.7100	243006.0000	-25.7100
32	2017	DXC	385340473	15871.6850	15880.4700	8.7850
33	2017	GOOGL	405865333	235802.8300	235883.1200	80.2900
34	2017	IBM	1069078580	39591.0260	39572.0150	-19.0110
35	2017	NVDA	4302783980	37613.4060	37595.7100	-17.6960
36	2018	AAPL	986094992	4464.6200	4461.4500	-3.1700
37	2018	ADP	53148104	3097.3300	3097.1500	-0.1800
38	2018	AMD	1741715362	319.4150	320.0100	0.5950
39	2018	AMZN	146335365	34464.3900	34558.8000	94.4100
40	2018	DXC	39188922	2594.1200	2595.0000	0.8800
41	2018	GOOGL	53061331	29380.5500	29430.3000	49.7500
42	2018	IBM	179536214	4227.5500	4224.0400	-3.5100
43	2018	NVDA	394421054	5883.5900	5920.4200	36.8300

```python
In [32]: pl = px.bar(data_frame=Stock_data.groupby(by = ['Year', 'Name']).sum().loc[:, ['volume', 'open', 'close']].reset_index(),
            x = 'volume', y = 'Name',orientation='h', animation_frame='Year', title='Sum of all the Stock Volumes recorded',
            labels = {"Name": 'Company', 'volume': 'Sum of Stock Volumes'})
pl.show()
```




```python
In [33]: pl = px.bar(data_frame=Stock_data.groupby(by = ['Year', 'Name']).sum().loc[:, ['volume', 'open', 'close', 'Close - Open']].reset_index(),
            x = 'Name', y = 'Close - Open', animation_frame='Year', title='Sum of Difference of Close - Open Price',
            labels = {"Name": 'Company', 'Close - Open': 'Sum of Difference in Close - Open price ($)'})
pl.show()
```

Intraday max possible profit

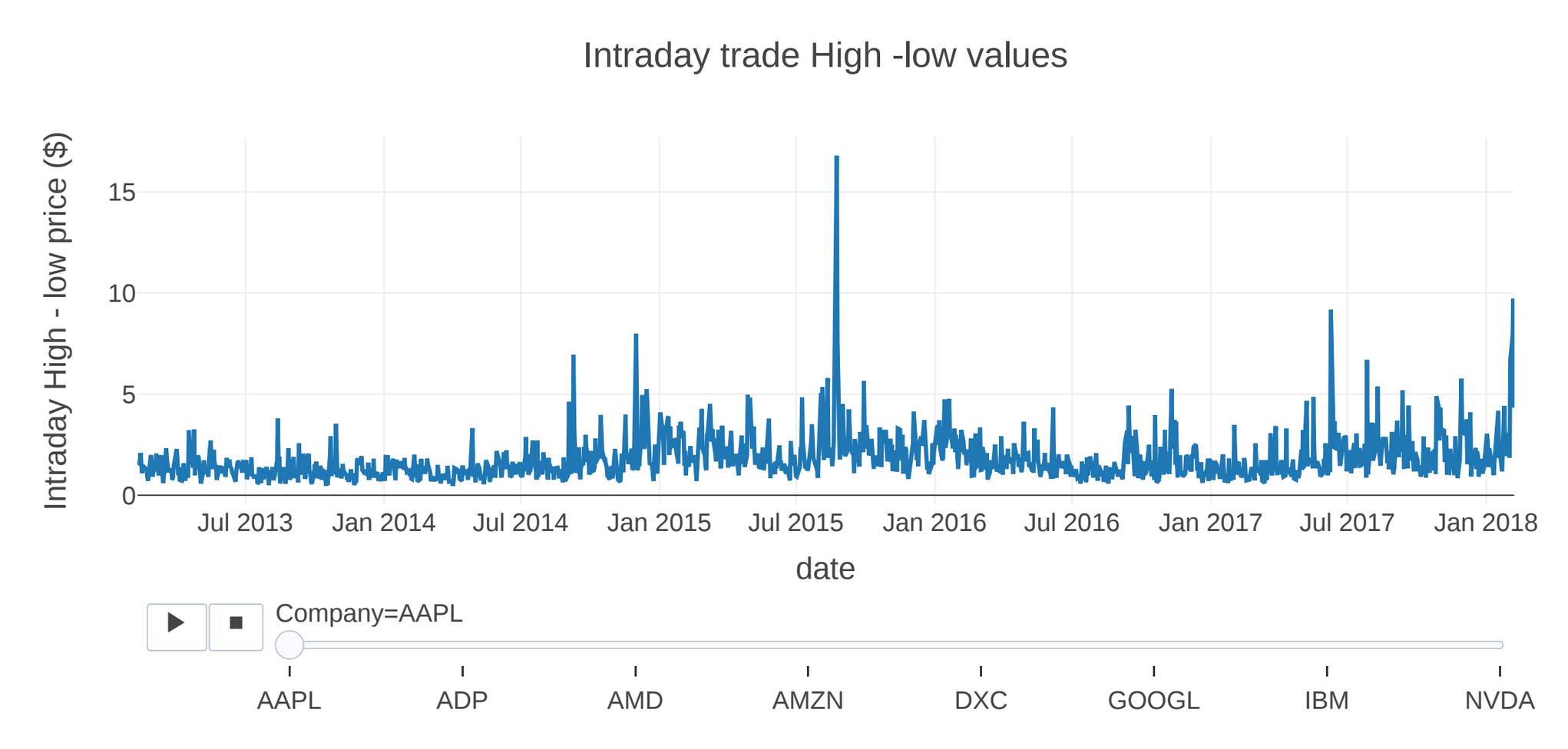
```python
In [34]: Stock_data['Intraday'] = Stock_data['high'] - Stock_data['low']
```

```python
In [35]: Stock_data.head()
```

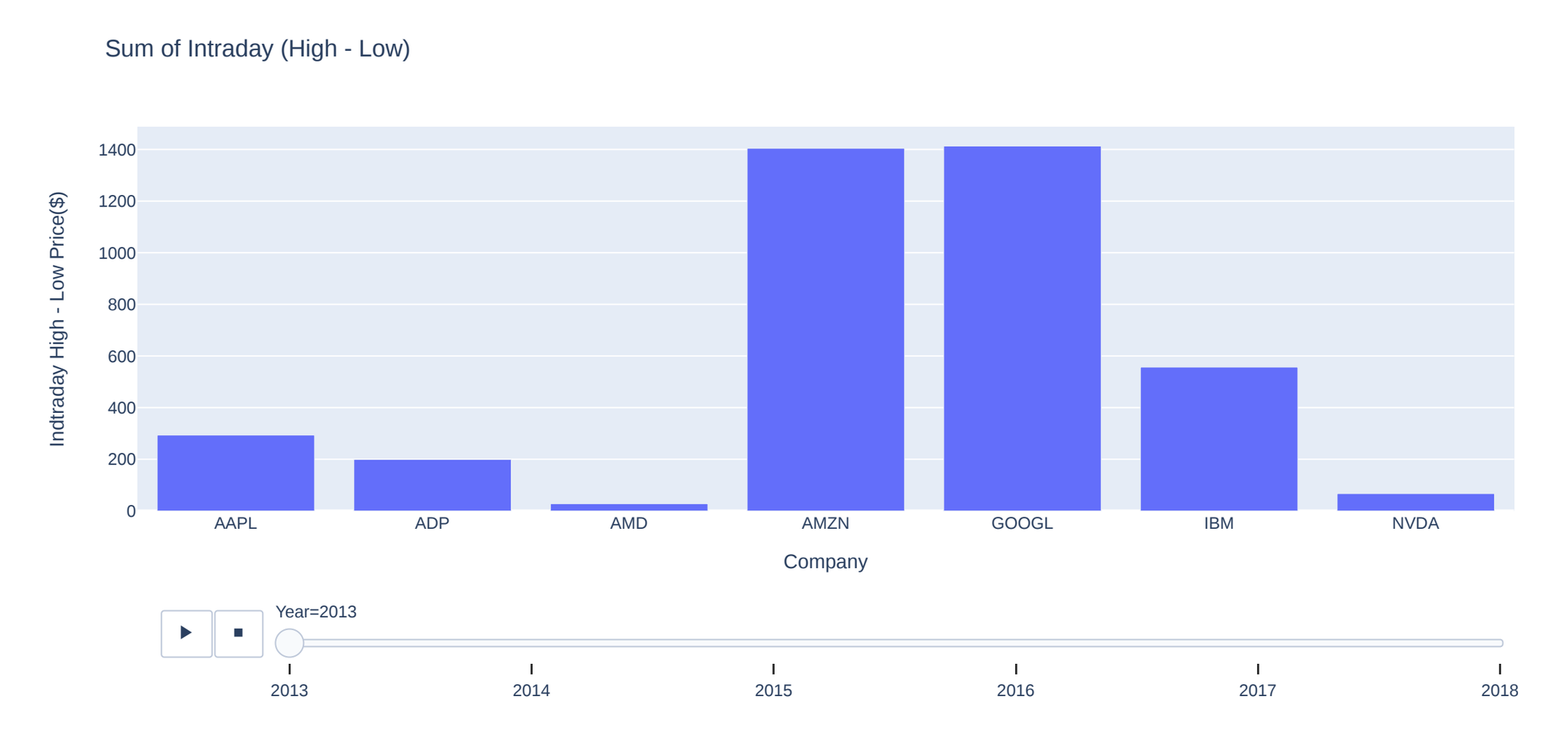
Out[35]:

	date	open	high	low	close	volume	Name	Year	Month	Close - Open	Intraday
0	2013-02-08	67.7142	68.4014	66.8928	67.8542	158168416	AAPL	2013	2	0.1400	1.5086
1	2013-02-11	68.0714	69.2771	67.6071	68.5614	129029425	AAPL	2013	2	0.4900	1.6700
2	2013-02-12	68.5014	68.9114	66.8205	66.8428	151829363	AAPL	2013	2	-1.6586	2.0909
3	2013-02-13	66.7442	67.6628	66.1742	66.7156	118721995	AAPL	2013	2	-0.0286	1.4886
4	2013-02-14	66.3599	67.3771	66.2885	66.6556	88809154	AAPL	2013	2	0.2957	1.0886

```python
In [36]: pl = px.line(data_frame=Stock_data, x="date", y="Intraday", animation_frame='Name', labels={
    "Name": "Company", "Intraday": "Intraday High - low price ($)"}, title="Intraday trade High -low values", template='presentation')
pl.show()
```




```python
In [37]: pl = px.bar(data_frame=Stock_data.groupby(by = ['Year', 'Name']).sum().loc[:, ['Intraday', 'Close - Open']].reset_index(),
            x = 'Name', y = 'Intraday', animation_frame='Year',
           labels = {"Name": 'Company', 'Intraday': 'Indtraday High - Low Price($)'}, title = "Sum of Intraday (High - Low)")
pl.show()
```



Value of Risk for Google

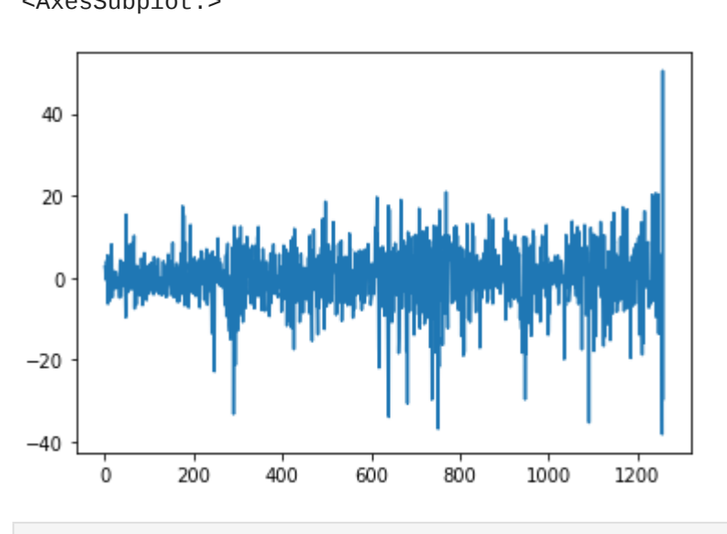
```python
In [38]: GOOGL.head()
```

Out[38]:

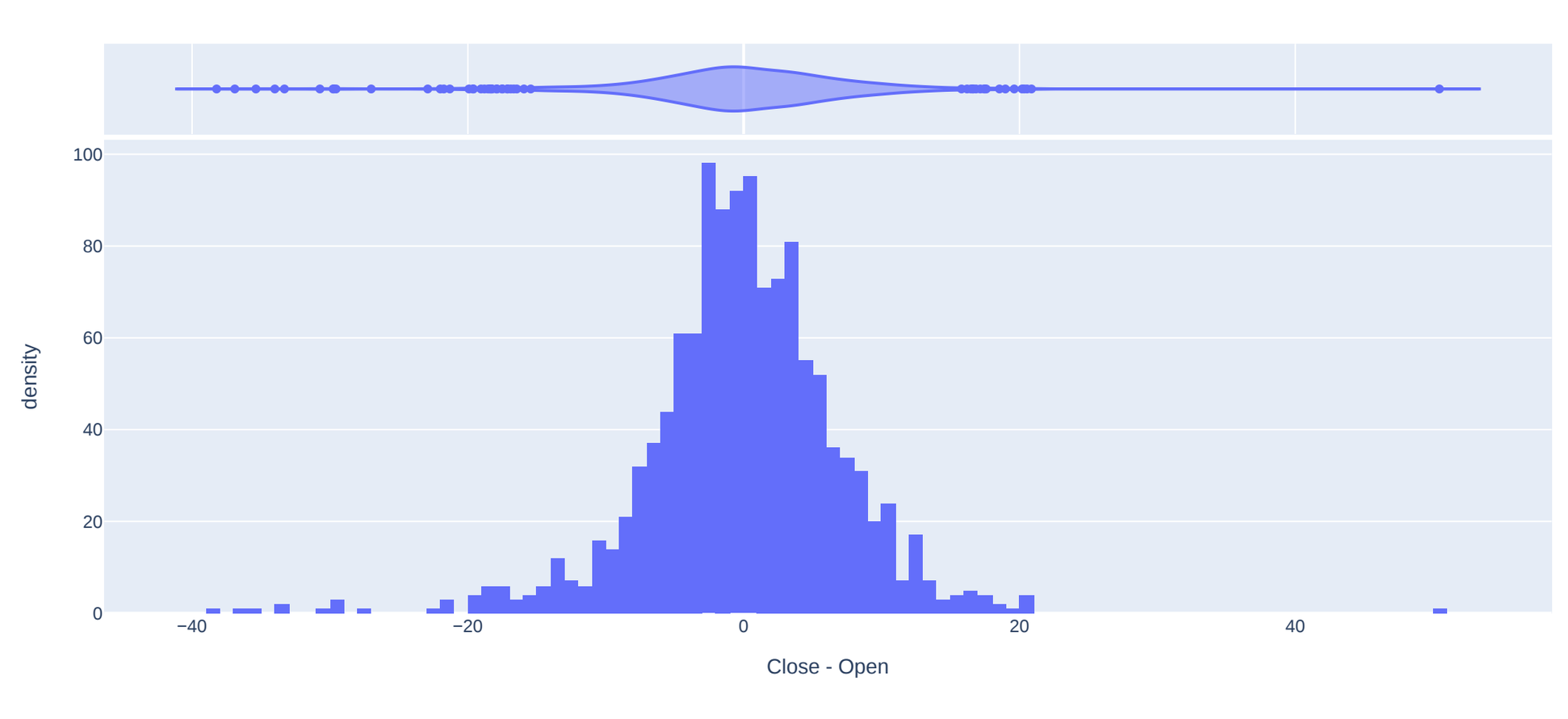
	date	open	high	low	close	volume	Name	Year	Month	Close - Open
0	2013-02-08	390.4551	393.7283	390.1698	393.0777	6031199	GOOGL	2013	2	2.6226
1	2013-02-11	389.5892	391.8915	387.2619	391.6012	4330781	GOOGL	2013	2	2.0120
2	2013-02-12	391.2659	394.3440	390.0747	390.7403	3714176	GOOGL	2013	2	-0.5256
3	2013-02-13	390.4551	393.0677	390.3750	391.8214	2393946	GOOGL	2013	2	1.3663
4	2013-02-14	390.2549	394.7644	389.2739	394.3039	3466971	GOOGL	2013	2	4.0490

```python
In [39]: GOOGL['Close - Open'].plot()
```

```
Out[39]: <AxesSubplot:>
```



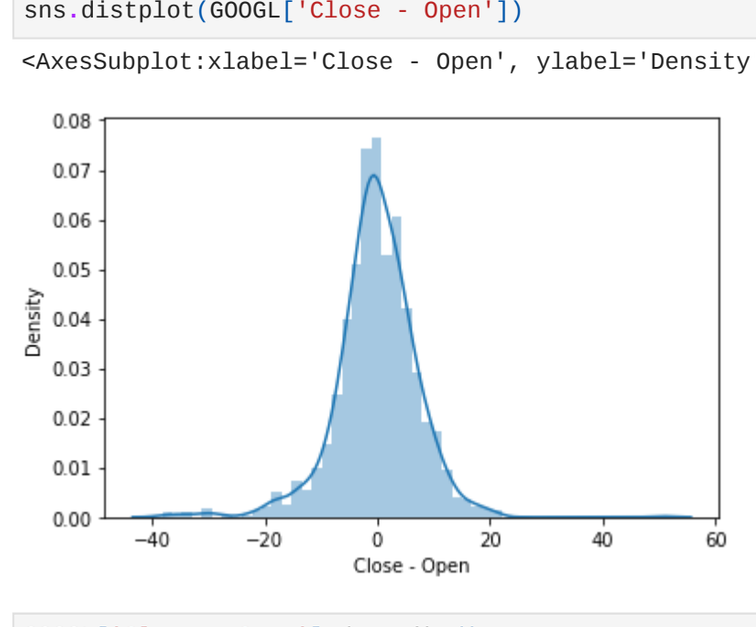

```python
In [40]: pl = px.histogram(GOOGL, x = 'Close - Open',  marginal="violin", histnorm= 'density')
pl.show()
```



Open - Close for Goole lies between 3.92 ($) and -3.77

```python
In [41]: sns.distplot(GOOGL['Close - Open'])
```

```
Out[41]: <AxesSubplot:xlabel='Close - Open', ylabel='Density'>
```




```python
In [42]: GOOGL['Close - Open'].describe()
```

```
Out[42]: count    1259.000000
mean       -0.123193
std         7.239046
min       -38.220000
25%        -3.775000
50%        -0.030000
75%         3.920000
max        50.450000
Name: Close - Open, dtype: float64
```

IQR range is 3.920000 - (-3.775000)