In [1]:

```
#import libraries and set preferences
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
pd.set_option("display.max_rows", 50)
np.set_printoptions(suppress = True)
pd.set_option('display.float_format', lambda x: '%.2f' % x)
```

In [2]:

```
#import data
rawdata = pd.read_csv('raw_data\S+P_500_Stock_Prices_2014-2017.csv')

#copy data to new variable for safety
data_wip = rawdata.copy()

#first glance at the data
print(data_wip.info())
print(data_wip)

#Verified that correct number of fields and records are available.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 497472 entries, 0 to 497471
Data columns (total 7 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   symbol  497472 non-null  object 
 1   date    497472 non-null  object 
 2   open    497461 non-null  float64
 3   high    497464 non-null  float64
 4   low     497464 non-null  float64
 5   close   497472 non-null  float64
 6   volume  497472 non-null  int64  
dtypes: float64(4), int64(1), object(2)
memory usage: 26.6+ MB
None
       symbol        date   open   high    low  close    volume
0         AAL  2014-01-02  25.07  25.82  25.06  25.36   8998943
1        AAPL  2014-01-02  79.38  79.58  78.86  79.02  58791957
2         AAP  2014-01-02 110.36 111.88 109.29 109.74    542711
3        ABBV  2014-01-02  52.12  52.33  51.52  51.98   4569061
4         ABC  2014-01-02  70.11  70.23  69.48  69.89   1148391
...       ...         ...    ...    ...    ...    ...       ...
497467    XYL  2017-12-29  68.53  68.80  67.92  68.20   1046677
497468    YUM  2017-12-29  82.64  82.71  81.59  81.61   1347613
497469    ZBH  2017-12-29 121.75 121.95 120.62 120.67   1023624
497470   ZION  2017-12-29  51.28  51.55  50.81  50.83   1261916
497471    ZTS  2017-12-29  72.55  72.76  72.04  72.04   1704122

[497472 rows x 7 columns]
```

In [3]:

```
##______CLEANING

#rename column "symbol" to "company", for user friendliness
data_wip = data_wip.rename(columns = {"symbol": "company"})

#check for any duplicated rows - there are none - all good.
data_wip[data_wip.duplicated(subset=None, keep='first')]
```

Out[3]:

company	date	open	high	low	close	volume

In [4]:

```
#check for any duplicated rows specifically on company and date combinations - there are none - all good.
data_wip[data_wip.duplicated(subset=["company", "date"], keep='first')]
```

Out[4]:

company	date	open	high	low	close	volume

In [5]:

```
#change dtype of date column to datetime.
data_wip["date"] = pd.to_datetime(data_wip["date"], format = "%Y-%m-%d")

#drop records where the high is lower than the low, as these are not reliable.
data_wip = data_wip.drop(np.argwhere(data_wip["low"] > data_wip["high"])[0])

#check for NaNs in the numerical fields
np.isnan(data_wip.iloc[:, 2:]).sum()
```

Out[5]:

```
open      11
high       8
low        8
close      0
volume     0
dtype: int64
```

In [6]:

```
#there are several NaNs within the DF.  Let's find out more...
#display rows containing atleast 1 NaN
data_wip.iloc[np.argwhere(np.isnan(data_wip.iloc[:, 2:]))[:,0]].drop_duplicates()
```

Out[6]:

	company	date	open	high	low	close	volume
166348	VRTX	2015-05-12	NaN	NaN	NaN	124.08	569747
175557	REGN	2015-06-09	NaN	NaN	NaN	526.09	12135
182011	WRK	2015-06-26	NaN	NaN	NaN	61.90	100
188547	DHR	2015-07-17	NaN	88.76	88.24	88.72	2056819
188578	ES	2015-07-17	NaN	48.49	47.85	47.92	1246786
188760	O	2015-07-17	NaN	47.31	46.83	46.99	1229513
249223	DHR	2016-01-12	NaN	NaN	NaN	88.55	0
249438	O	2016-01-12	NaN	NaN	NaN	52.43	0
278801	UA	2016-04-07	NaN	NaN	NaN	41.56	0
308365	FTV	2016-07-01	NaN	NaN	NaN	49.54	0
442107	BHF	2017-07-26	NaN	NaN	NaN	69.08	3

In [7]:

```
#drop records containing NAN values, as they are null in the source data, and are therefore no use to us and may skew the analysis
data_wip = data_wip.drop(data_wip.iloc[np.argwhere(np.isnan(data_wip.iloc[:, 2:]))[:,0]].drop_duplicates().index)

#show records where there was 0 volume traded
data_wip.iloc[list(np.argwhere(data_wip["volume"] == 0).squeeze())]
```

Out[7]:

company	date	open	high	low	close	volume

In [8]:

```
#there are no 0-volume records now, only the NaN records showed 0 volume, these have been dropped.
#calculate some new columns...

#create a day of the week column
data_wip["day"] = data_wip["date"].dt.day_name()

#create a percent day change column - ie (close-open)/open * 100
data_wip["pc_day_change"] = (data_wip["close"] - data_wip["open"]) / data_wip["open"] * 100

#create an absolute day change column - ie close - open
data_wip["abs_day_change"] = data_wip["close"] - data_wip["open"]

#create a percent volatility column - ie (high-low)/low * 100
data_wip["pc_volatility"] = (data_wip["high"] - data_wip["low"]) / data_wip["low"] * 100

#create an absolute volatility column - ie high - low
data_wip["abs_volatility"] = data_wip["high"] - data_wip["low"]

#re-order columns
data_wip = data_wip[["company", "date", "day", "open", "close", "abs_day_change", "pc_day_change", "low", "high", "abs_volatility", \
                     "pc_volatility", "volume"]]

#re-order rows - sort by date firstly, then by company, then reset the index so that rows are numbered correctly.
data_wip = data_wip.sort_values(by=['date', "company"], ascending=True).reset_index(drop = True)

#eyeball the cleaned data
data_wip
```

Out[8]:

	company	date	day	open	close	abs_day_change	pc_day_change	low	high	abs_volatility	pc_volatility	volume
0	A	2014-01-02	Thursday	57.10	56.21	-0.89	-1.56	56.15	57.10	0.95	1.69	1916160
1	AAL	2014-01-02	Thursday	25.07	25.36	0.29	1.16	25.06	25.82	0.76	3.03	8998943
2	AAP	2014-01-02	Thursday	110.36	109.74	-0.62	-0.56	109.29	111.88	2.59	2.37	542711
3	AAPL	2014-01-02	Thursday	79.38	79.02	-0.36	-0.46	78.86	79.58	0.72	0.91	58791957
4	ABBV	2014-01-02	Thursday	52.12	51.98	-0.14	-0.27	51.52	52.33	0.81	1.57	4569061
...	...	...	...	...	...	...	...	...	...	...	...	...
497455	XYL	2017-12-29	Friday	68.53	68.20	-0.33	-0.48	67.92	68.80	0.88	1.30	1046677
497456	YUM	2017-12-29	Friday	82.64	81.61	-1.03	-1.25	81.59	82.71	1.12	1.37	1347613
497457	ZBH	2017-12-29	Friday	121.75	120.67	-1.08	-0.89	120.62	121.95	1.33	1.10	1023624
497458	ZION	2017-12-29	Friday	51.28	50.83	-0.45	-0.88	50.81	51.55	0.74	1.46	1261916
497459	ZTS	2017-12-29	Friday	72.55	72.04	-0.51	-0.70	72.04	72.76	0.72	1.00	1704122

497460 rows × 12 columns

In [ ]:

In [ ]:

In [9]:

```
##_____ANALYSIS

#Look at min, mean and max values for each field
```

```
print("Min:")
print(np.min(data_wip.iloc[:,3:], axis = 0))
print("\nMean:")
print(np.mean(data_wip.iloc[:,3:], axis = 0))
print("\nMax:")
print(np.max(data_wip.iloc[:,3:], axis = 0))
```

```
Min:
open                1.62
close               1.59
abs_day_change   -100.98
pc_day_change     -49.22
low                 1.50
high                1.69
abs_volatility      0.00
pc_volatility       0.00
volume            101.00
dtype: float64

Mean:
open                   86.35
close                  86.37
abs_day_change          0.02
pc_day_change           0.03
low                    85.55
high                   87.13
abs_volatility          1.58
pc_volatility           1.95
volume            4253699.31
dtype: float64

Max:
open                   2044.00
close                  2049.00
abs_day_change           52.95
pc_day_change            80.58
low                    2035.11
high                   2067.99
abs_volatility          109.90
pc_volatility           596.67
volume            618237630.00
dtype: float64
```

In [10]:

```
#show mean values for each date
data_wip.groupby(["date"])[["open", "close", "abs_day_change","pc_day_change", "low", "high", "abs_volatility", "pc_volatility", \
                             "volume"]].mean()
```

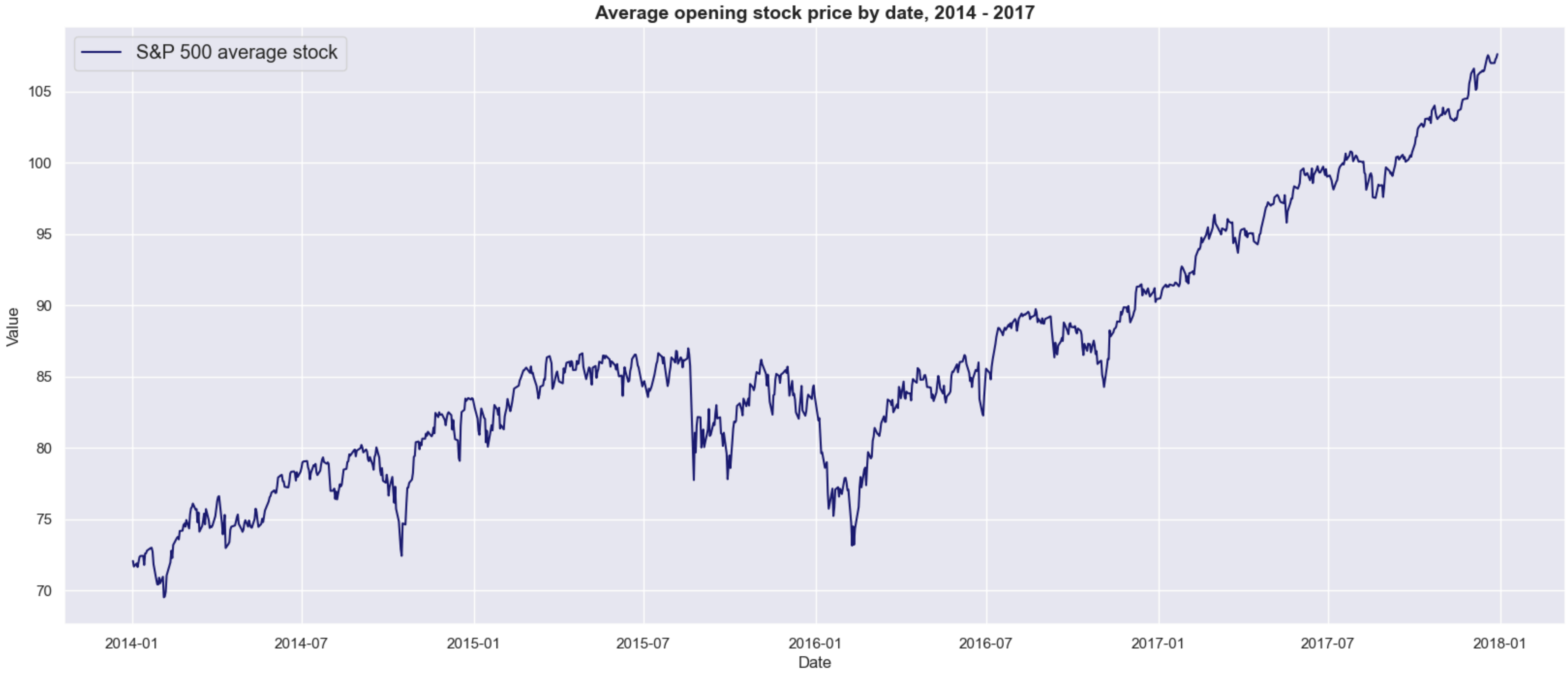
Out[10]:

	open	close	abs_day_change	pc_day_change	low	high	abs_volatility	pc_volatility	volume
date									
2014-01-02	72.07	71.61	-0.46	-0.62	71.20	72.44	1.24	1.75	3723074.50
2014-01-03	71.68	71.67	-0.01	0.01	71.24	72.25	1.02	1.44	3478364.01
2014-01-06	71.90	71.42	-0.48	-0.68	71.01	72.29	1.28	1.77	4383140.52
2014-01-07	71.64	72.01	0.36	0.38	71.24	72.47	1.23	1.72	4455832.71
2014-01-08	72.01	72.11	0.10	0.11	71.41	72.61	1.20	1.72	4793434.16
...	...	...	...	...	...	...	...	...	...
2017-12-22	106.98	107.05	0.06	0.08	106.31	107.66	1.35	1.37	2879367.58
2017-12-26	106.99	107.12	0.13	0.15	106.44	107.77	1.33	1.36	2186340.80
2017-12-27	107.24	107.19	-0.05	-0.11	106.62	107.77	1.15	1.14	2420034.87
2017-12-28	107.36	107.48	0.12	0.09	106.69	107.84	1.15	1.14	2282111.61
2017-12-29	107.61	106.89	-0.72	-0.66	106.70	108.06	1.35	1.32	2703152.98

1007 rows × 9 columns

In [11]:

```
#visualise mean opening stock value, over time.
plt.figure(figsize = (20,8))
plt.plot(data_wip.groupby(["date"])[["open"]].mean(), color = "midnightblue")
plt.title("Average opening stock price by date, 2014 - 2017", fontsize = 14, fontweight = "bold")
plt.ylabel("Value")
plt.xlabel("Date")
plt.legend(labels = ["S&P 500 average stock"], fontsize = "large")
plt.show()
```



In [12]:

```
#show mean values for each company
data_wip.groupby(["company"])[["open", "close", "abs_day_change","pc_day_change", "low", "high", "abs_volatility", "pc_volatility", \
                                "volume"]].mean()
```

Out[12]:

	open	close	abs_day_change	pc_day_change	low	high	abs_volatility	pc_volatility	volume
company									
A	49.09	49.10	0.01	0.04	48.68	49.49	0.80	1.70	2191931.57
AAL	42.42	42.42	0.00	0.04	41.80	43.04	1.24	3.03	9751521.26
AAP	143.21	143.19	-0.02	-0.01	141.66	144.72	3.05	2.21	1137306.71
AAPL	116.81	116.84	0.03	0.03	115.86	117.75	1.89	1.67	45169571.17
ABBV	63.30	63.34	0.04	0.07	62.65	63.96	1.31	2.14	8408835.98
...	...	...	...	...	...	...	...	...	...
XYL	43.50	43.52	0.02	0.04	43.14	43.85	0.71	1.69	1159214.06
YUM	76.38	76.40	0.02	0.03	75.79	76.99	1.20	1.58	3225015.93
ZBH	110.54	110.59	0.04	0.04	109.64	111.45	1.82	1.68	1349760.22
ZION	32.67	32.67	0.00	0.03	32.30	33.02	0.72	2.29	2758079.00
ZTS	47.25	47.27	0.02	0.05	46.83	47.66	0.83	1.82	3485038.12

505 rows × 9 columns

In [13]:

```
#show dates with the highest and lowest volumes of trading
data_wip.groupby(["date"])[["volume"]].sum().sort_values(by = ["volume"], ascending = False)
```

Out[13]:

	volume
date	
2015-08-24	4607945196
2016-06-24	4367393052
2015-12-18	4124454411
2016-01-20	4087629753
2016-11-10	4060601612
...	...
2014-12-26	894908944
2015-11-27	791154818
2014-12-24	750895627
2015-12-24	736263173
2017-11-24	728261080

1007 rows × 1 columns

In [14]:

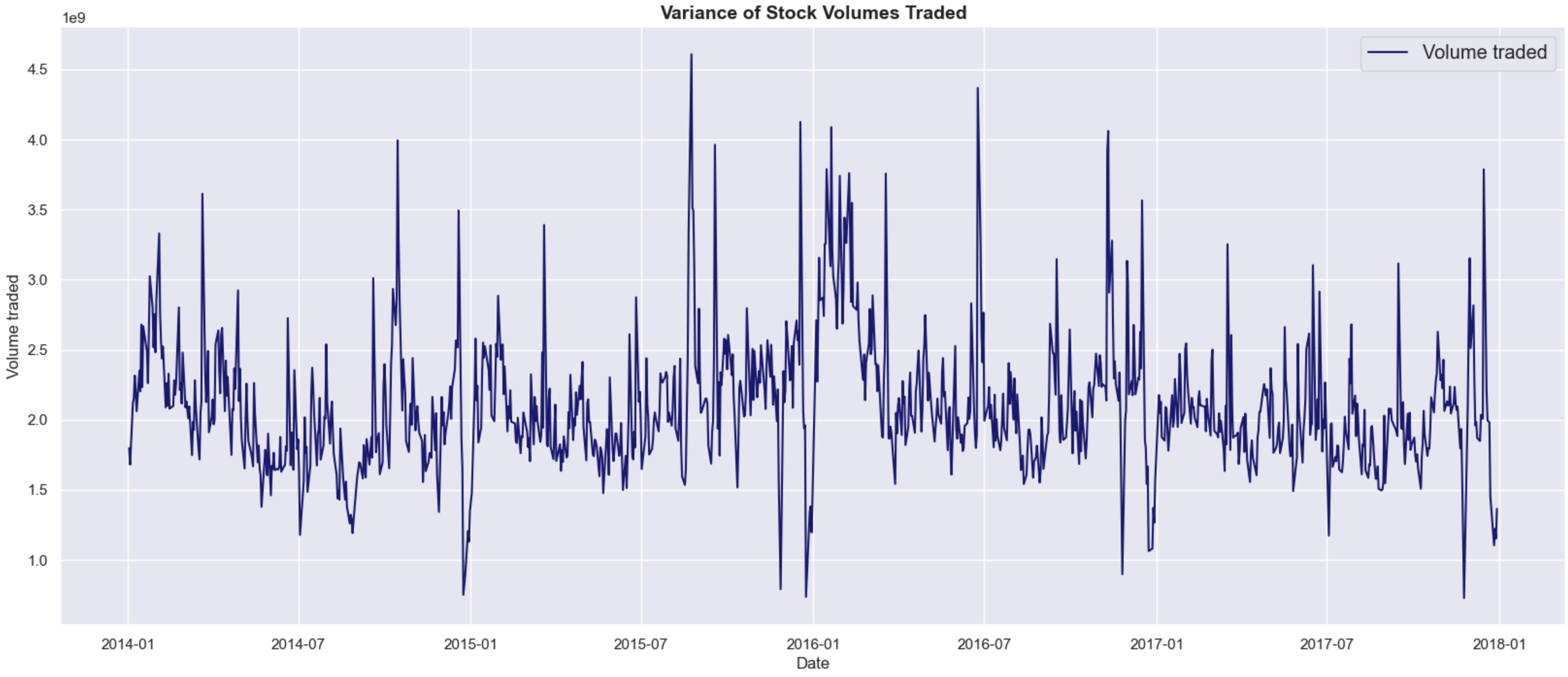
```
#show the bottom 20 records of the above query
data_wip.groupby(["date"])[["volume"]].sum().sort_values(by = ["volume"], ascending = False).iloc[-1:-13:-1]
```

Out[14]:

date	volume
2017-11-24	728261080
2015-12-24	736263173
2014-12-24	750895627
2015-11-27	791154818
2014-12-26	894908944
2016-11-25	898051561
2016-12-23	1063636294
2016-12-27	1080274822
2017-12-26	1104102103
2014-12-30	1130517899
2017-12-28	1152466365
2017-07-03	1173653906

In [15]:

```
#create a line chart to show the variance of trading volumes

plt.figure(figsize = (20,8))
plt.plot(data_wip.groupby(["date"])[["volume"]].sum(), color = "midnightblue")
plt.title("Variance of Stock Volumes Traded", fontsize = 14, fontweight = "bold")
plt.ylabel("Volume traded")
plt.xlabel("Date")
plt.legend(labels = ["Volume traded"], fontsize = "large")
plt.show()
```



In [16]:

```
#which two stocks were most highly traded on the day with the most trading.
data_wip.iloc[np.argwhere(data_wip["date"] == "2015-08-24").squeeze()].sort_values(by = "volume", ascending = False).iloc[0:2,[0,1,11]]
```

Out[16]:

	company	date	volume
201260	BAC	2015-08-24	214649482
201204	AAPL	2015-08-24	162206292

In [17]:

```
#show companies with the highest and lowest volumes of trading across the 4 years
data_wip.groupby(["company"])[["volume"]].sum().sort_values(by = ["volume"], ascending = False)
```

Out[17]:

	volume
company	
BAC	89988444028
AAPL	45485758169
GE	41734050117
AMD	33522535638
F	33144701045
...	...
HII	335472634
AZO	323425210
MTD	173123302
BHF	122667236
APTV	36306643

505 rows × 1 columns

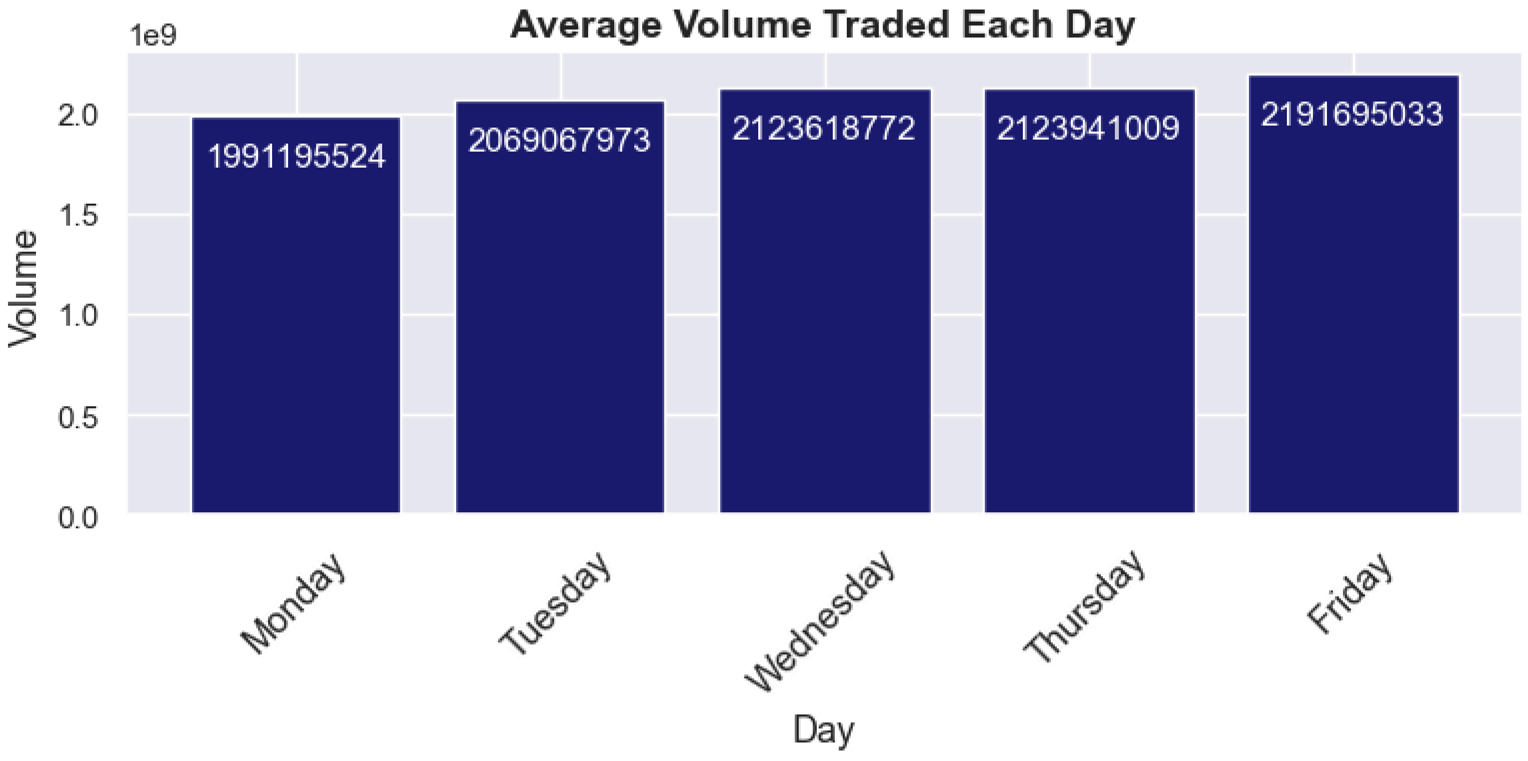
In [18]:

```
#calculate average volume by day of the week

#group by date, showing the sum of all volume traded on each date.  Then concat with the day column.
groupby_date_concat = pd.concat([(data_wip.groupby(["date"])[["volume"]].sum()), (data_wip.groupby(["date"])[["day"]].min())],\
                                axis = 1)

#now group by day, and show the mean volume traded on each day
avg_vol_by_day = groupby_date_concat.groupby(["day"])[["volume"]].mean().sort_values(by="volume", ascending = False)

#print
print(avg_vol_by_day)
```

```
                 volume
day
Friday    2191695033.16
Thursday  2123941009.38
Wednesday 2123618772.80
Tuesday   2069067973.93
Monday    1991195524.81
```

In [19]:

```
#visualise the above in a bar chart

avg_vol_by_day_ordered = avg_vol_by_day.iloc[[4,3,2,1,0]]

plt.figure(figsize= (9,3))
plt.bar(x = avg_vol_by_day_ordered.index, height = avg_vol_by_day_ordered["volume"], color = "midnightblue")

plt.xticks(rotation = 45, fontsize = 13)
plt.title("Average Volume Traded Each Day", fontsize = 14, fontweight = "bold")
plt.ylabel("Volume", fontsize = 13)
plt.xlabel("Day", fontsize = 13)
for index, value in enumerate(avg_vol_by_day_ordered["volume"]):
    plt.text(index, value-250000000, int(value), color = "white", ha="center")
plt.show()
```



In [20]:

```
#Show records with the biggest and smallest % day change.
data_wip.sort_values(by = "pc_day_change", ascending = False)
```

Out[20]:

	company	date	day	open	close	abs_day_change	pc_day_change	low	high	abs_volatility	pc_volatility	volume
45548	CHD	2014-05-19	Monday	18.77	33.90	15.13	80.58	33.70	33.95	0.26	0.76	1078888
250551	WMB	2016-01-14	Thursday	13.42	18.29	4.87	36.29	13.24	18.44	5.20	39.27	42552128
266038	CHK	2016-03-02	Wednesday	2.62	3.40	0.78	29.77	2.60	3.75	1.15	44.23	76288029
266534	CHK	2016-03-03	Thursday	3.37	4.27	0.90	26.71	3.32	4.72	1.40	42.17	138475442
283843	AMD	2016-04-22	Friday	3.19	3.99	0.80	25.08	3.18	3.99	0.81	25.47	143265305
...	...	...	...	...	...	...	...	...	...	...	...	...
220270	PWR	2015-10-16	Friday	23.06	18.74	-4.32	-18.73	18.51	23.06	4.55	24.58	24411177
248764	FCX	2016-01-11	Monday	5.40	4.31	-1.09	-20.19	4.23	5.42	1.19	28.13	117668158
258102	CHK	2016-02-08	Monday	2.56	2.04	-0.52	-20.31	1.50	2.59	1.09	72.67	121984560
258487	WMB	2016-02-08	Monday	14.93	11.16	-3.77	-25.25	10.22	15.00	4.78	46.77	62368018
293534	LNT	2016-05-19	Thursday	35.19	17.87	-17.32	-49.22	35.09	35.78	0.70	2.00	1231722

497460 rows × 12 columns

In [21]:

```
#show records where Amazon saw the most and least volatility - judged by percentage
data_wip.iloc[np.argwhere(data_wip["company"] == "AMZN").squeeze()].sort_values(by="pc_volatility", ascending = False)
```

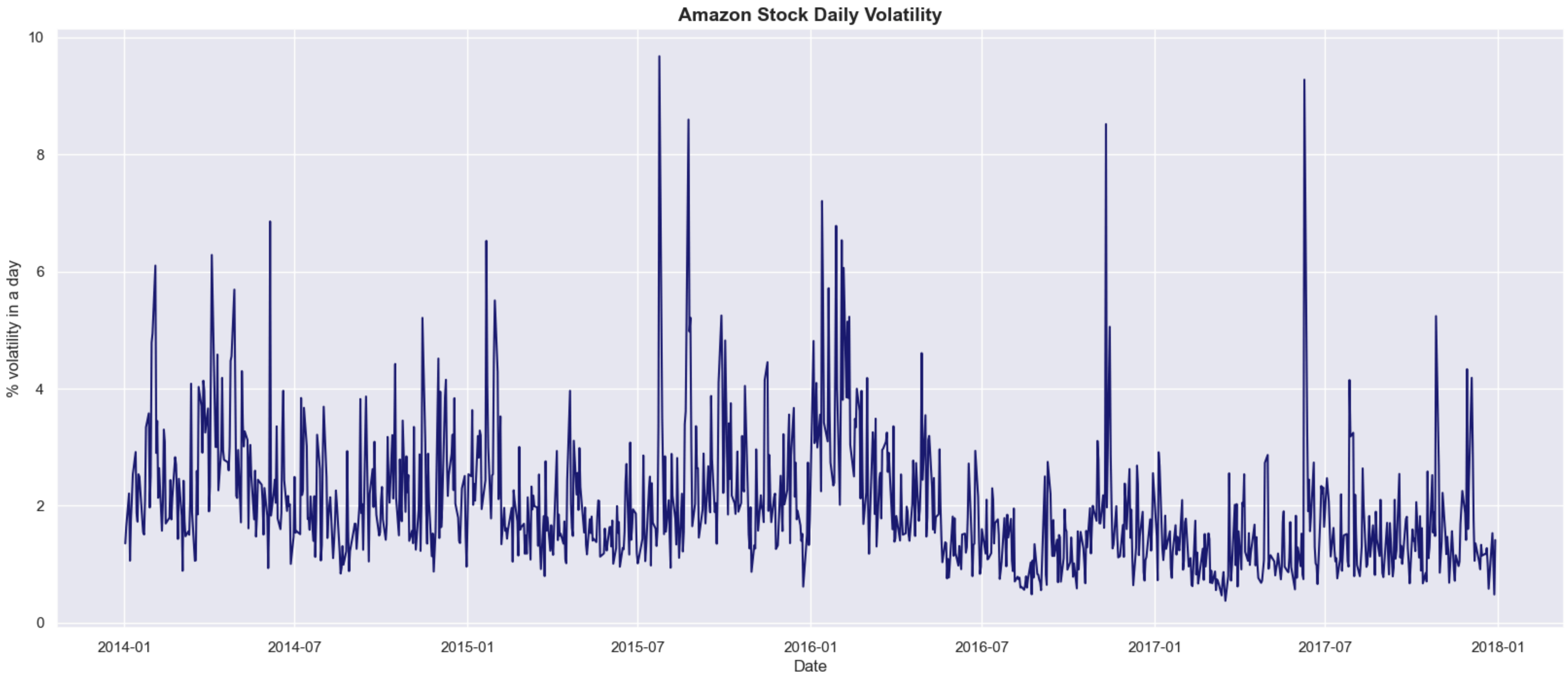
Out[21]:

	company	date	day	open	close	abs_day_change	pc_day_change	low	high	abs_volatility	pc_volatility	volume
190907	AMZN	2015-07-24	Friday	578.99	529.42	-49.57	-8.56	529.35	580.57	51.22	9.68	21909381
426013	AMZN	2017-06-09	Friday	1012.50	978.31	-34.19	-3.38	927.00	1012.99	85.99	9.28	7647692
201239	AMZN	2015-08-24	Monday	463.58	463.37	-0.21	-0.05	451.00	489.76	38.76	8.59	10097601
354017	AMZN	2016-11-10	Thursday	778.81	742.38	-36.43	-4.68	717.70	778.83	61.13	8.52	12746994
249616	AMZN	2016-01-13	Wednesday	620.88	581.81	-39.07	-6.29	579.16	620.88	41.72	7.20	7655239
...	...	...	...	...	...	...	...	...	...	...	...	...
330113	AMZN	2016-09-02	Friday	774.11	772.44	-1.67	-0.22	771.70	776.00	4.30	0.56	2181792
326129	AMZN	2016-08-23	Tuesday	763.31	762.45	-0.86	-0.11	761.00	764.70	3.70	0.49	1524131
496488	AMZN	2017-12-28	Thursday	1189.00	1186.10	-2.90	-0.24	1184.38	1190.10	5.72	0.48	1841676
394966	AMZN	2017-03-13	Monday	851.77	854.59	2.82	0.33	851.71	855.69	3.98	0.47	1909672
396966	AMZN	2017-03-17	Friday	853.49	852.31	-1.18	-0.14	850.64	853.83	3.19	0.38	3384403

1007 rows × 12 columns

In [22]:

```
#visualise Amazon percentage volatility over time on a line chart
amazon_volatility_by_date = data_wip.iloc[np.argwhere(data_wip["company"] == "AMZN").squeeze()]["pc_volatility"]
amazon_volatility_by_date.index = data_wip.iloc[np.argwhere(data_wip["company"] == "AMZN").squeeze()]["date"]

plt.figure(figsize = (20,8))
plt.plot(amazon_volatility_by_date, color = "midnightblue")
plt.title("Amazon Stock Daily Volatility", fontsize = 14, fontweight = "bold")
plt.ylabel("% volatility in a day")
plt.xlabel("Date")
plt.show()
```



In [23]:

```
#show records where Amazon saw the most and least volatility - judged by absolute value.
data_wip.iloc[np.argwhere(data_wip["company"] == "AMZN").squeeze()].sort_values(by="abs_volatility", ascending = False)
```

Out[23]:

	company	date	day	open	close	abs_day_change	pc_day_change	low	high	abs_volatility	pc_volatility	volume
426013	AMZN	2017-06-09	Friday	1012.50	978.31	-34.19	-3.38	927.00	1012.99	85.99	9.28	7647692
354017	AMZN	2016-11-10	Thursday	778.81	742.38	-36.43	-4.68	717.70	778.83	61.13	8.52	12746994
475304	AMZN	2017-10-27	Friday	1058.14	1100.95	42.81	4.05	1050.55	1105.58	55.03	5.24	16565021
190907	AMZN	2015-07-24	Friday	578.99	529.42	-49.57	-8.56	529.35	580.57	51.22	9.68	21909381
486392	AMZN	2017-11-29	Wednesday	1194.80	1161.27	-33.53	-2.81	1145.19	1194.80	49.61	4.33	9257512
...	...	...	...	...	...	...	...	...	...	...	...	...
149299	AMZN	2015-03-24	Tuesday	373.99	374.09	0.10	0.03	372.27	375.24	2.97	0.80	2228214
121946	AMZN	2014-12-31	Wednesday	311.55	310.35	-1.20	-0.39	310.01	312.98	2.97	0.96	2057766
110722	AMZN	2014-11-26	Wednesday	333.78	333.57	-0.21	-0.06	331.75	334.65	2.90	0.87	1985949
50346	AMZN	2014-06-03	Tuesday	305.75	307.19	1.44	0.47	305.07	307.92	2.85	0.93	2379273
76587	AMZN	2014-08-19	Tuesday	334.87	335.13	0.26	0.08	333.01	335.81	2.80	0.84	1714120

1007 rows × 12 columns

In [24]:

```
#show records where Amazon saw the highest day gains and worst day loses, by percentage
data_wip.iloc[np.argwhere(data_wip["company"] == "AMZN").squeeze()].sort_values(by="pc_day_change", ascending = False)
```

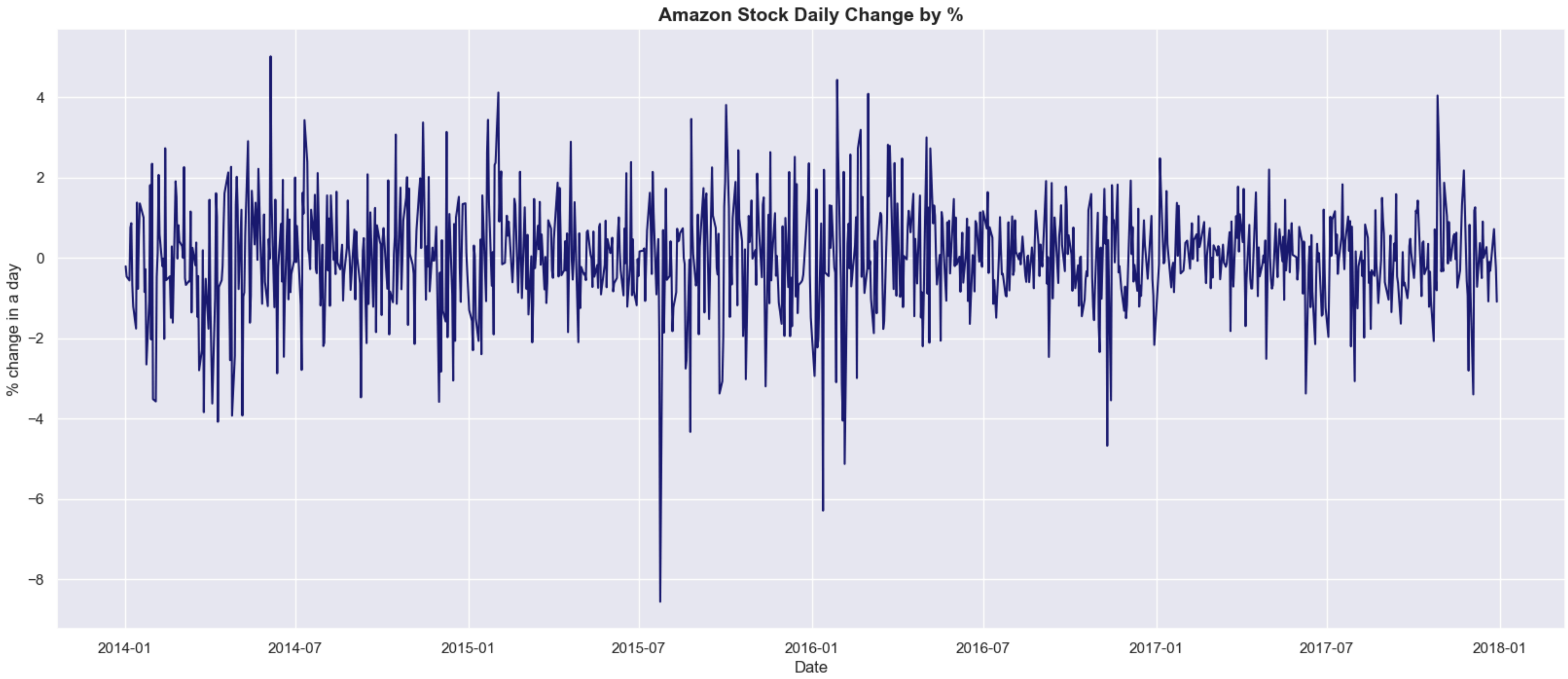
Out[24]:

	company	date	day	open	close	abs_day_change	pc_day_change	low	high	abs_volatility	pc_volatility	volume
51316	AMZN	2014-06-05	Thursday	308.10	323.57	15.47	5.02	306.90	327.94	21.04	6.86	7803760
254576	AMZN	2016-01-28	Thursday	608.37	635.35	26.98	4.43	597.55	638.06	40.51	6.78	14015171
132214	AMZN	2015-02-02	Monday	350.05	364.47	14.42	4.12	350.01	365.00	14.99	4.28	10231914
265488	AMZN	2016-03-01	Tuesday	556.29	579.04	22.75	4.09	556.00	579.25	23.25	4.18	5038452
475304	AMZN	2017-10-27	Friday	1058.14	1100.95	42.81	4.05	1050.55	1105.58	55.03	5.24	16565021
...	...	...	...	...	...	...	...	...	...	...	...	...
201731	AMZN	2015-08-25	Tuesday	487.49	466.37	-21.12	-4.33	466.25	489.44	23.19	4.97	5679329
354017	AMZN	2016-11-10	Thursday	778.81	742.38	-36.43	-4.68	717.70	778.83	61.13	8.52	12746994
257552	AMZN	2016-02-05	Friday	529.28	502.13	-27.15	-5.13	499.19	529.45	30.26	6.06	9708929
249616	AMZN	2016-01-13	Wednesday	620.88	581.81	-39.07	-6.29	579.16	620.88	41.72	7.20	7655239
190907	AMZN	2015-07-24	Friday	578.99	529.42	-49.57	-8.56	529.35	580.57	51.22	9.68	21909381

1007 rows × 12 columns

In [25]:

```
#visualise Amazon percentage day change over time on a line chart
amazon_day_change_by_date = data_wip.iloc[np.argwhere(data_wip["company"] == "AMZN").squeeze()]["pc_day_change"]
amazon_day_change_by_date.index = data_wip.iloc[np.argwhere(data_wip["company"] == "AMZN").squeeze()]["date"]

plt.figure(figsize = (20,8))
plt.plot(amazon_day_change_by_date, color = "midnightblue")
plt.title("Amazon Stock Daily Change by %", fontsize = 14, fontweight = "bold")
plt.ylabel("% change in a day")
plt.xlabel("Date")
plt.show()
```



In [26]:

```
#show records where Amazon saw the highest day gains and worst day loses, by absolute value.
data_wip.iloc[np.argwhere(data_wip["company"] == "AMZN").squeeze()].sort_values(by="abs_day_change", ascending = False)
```

Out[26]:

	company	date	day	open	close	abs_day_change	pc_day_change	low	high	abs_volatility	pc_volatility	volume
475304	AMZN	2017-10-27	Friday	1058.14	1100.95	42.81	4.05	1050.55	1105.58	55.03	5.24	16565021
254576	AMZN	2016-01-28	Thursday	608.37	635.35	26.98	4.43	597.55	638.06	40.51	6.78	14015171
484880	AMZN	2017-11-24	Friday	1160.70	1186.00	25.30	2.18	1160.70	1186.84	26.14	2.25	3528011
265488	AMZN	2016-03-01	Tuesday	556.29	579.04	22.75	4.09	556.00	579.25	23.25	4.18	5038452
477824	AMZN	2017-11-03	Friday	1091.15	1111.60	20.45	1.87	1088.52	1112.68	24.16	2.22	3751480
...	...	...	...	...	...	...	...	...	...	...	...	...
426013	AMZN	2017-06-09	Friday	1012.50	978.31	-34.19	-3.38	927.00	1012.99	85.99	9.28	7647692
354017	AMZN	2016-11-10	Thursday	778.81	742.38	-36.43	-4.68	717.70	778.83	61.13	8.52	12746994
249616	AMZN	2016-01-13	Wednesday	620.88	581.81	-39.07	-6.29	579.16	620.88	41.72	7.20	7655239
487904	AMZN	2017-12-04	Monday	1173.85	1133.95	-39.90	-3.40	1128.00	1175.20	47.20	4.18	5931915
190907	AMZN	2015-07-24	Friday	578.99	529.42	-49.57	-8.56	529.35	580.57	51.22	9.68	21909381

1007 rows × 12 columns

In [27]:

```
#FYI show a list of companies that are recorded on 2nd Jan 2014 list but not on 29th Dec 2017
jan_2014_companies = list(data_wip[(data_wip["date"] == "2014-01-02")]["company"])
dec_2017_companies = list(data_wip[(data_wip["date"] == "2017-12-29")]["company"])

count = 1

for x in range(len(dec_2017_companies)):
    if dec_2017_companies[x] not in jan_2014_companies:
        print (count, dec_2017_companies[x], "wasn't on list on 2nd Jan 2014")
        count += 1
#so naturally these won't be included in the next query.
```

```
1 APTV wasn't on list on 2nd Jan 2014
2 BHF wasn't on list on 2nd Jan 2014
3 BHGE wasn't on list on 2nd Jan 2014
4 CFG wasn't on list on 2nd Jan 2014
5 CSRA wasn't on list on 2nd Jan 2014
6 DWDP wasn't on list on 2nd Jan 2014
7 DXC wasn't on list on 2nd Jan 2014
8 EVHC wasn't on list on 2nd Jan 2014
9 FTV wasn't on list on 2nd Jan 2014
10 GOOG wasn't on list on 2nd Jan 2014
11 HLT wasn't on list on 2nd Jan 2014
12 HPE wasn't on list on 2nd Jan 2014
13 HPQ wasn't on list on 2nd Jan 2014
14 INFO wasn't on list on 2nd Jan 2014
15 KHC wasn't on list on 2nd Jan 2014
16 NAVI wasn't on list on 2nd Jan 2014
17 PYPL wasn't on list on 2nd Jan 2014
18 QRVO wasn't on list on 2nd Jan 2014
19 SYF wasn't on list on 2nd Jan 2014
20 UA wasn't on list on 2nd Jan 2014
21 WLTW wasn't on list on 2nd Jan 2014
22 WRK wasn't on list on 2nd Jan 2014
```

In [28]:

```
#In hindsight, which stock should you have invested in from 2nd Jan 2014 - 29th Dec 2017 to get best return.
jan_2014_open = data_wip[(data_wip["date"] == "2014-01-02")][["company", "date", "open"]]
dec_2017_close = data_wip[(data_wip["date"] == "2017-12-29")][["company", "date", "close"]]
jan_2014_dec_2017_merged = pd.merge(jan_2014_open, dec_2017_close, on="company")
jan_2014_dec_2017_merged["price_difference"] = jan_2014_dec_2017_merged["close"] - jan_2014_dec_2017_merged["open"]
jan_2014_dec_2017_merged["price_difference_pc"] = (jan_2014_dec_2017_merged["close"] - jan_2014_dec_2017_merged["open"]) \
/ jan_2014_dec_2017_merged["open"] * 100

jan_2014_dec_2017_merged.sort_values(by = ["price_difference_pc"], ascending = False)
#Nvidia is the clear winner, for % growth.
```

Out[28]:

	company	date_x	open	date_y	close	price_difference	price_difference_pc
330	NVDA	2014-01-02	15.92	2017-12-29	193.50	177.58	1115.45
52	AVGO	2014-01-02	52.85	2017-12-29	256.90	204.05	386.09
146	EA	2014-01-02	22.90	2017-12-29	105.06	82.16	358.78
26	ALGN	2014-01-02	57.06	2017-12-29	222.19	165.13	289.40
318	NFLX	2014-01-02	52.40	2017-12-29	191.96	139.56	266.33
...	...	...	...	...	...	...	...
285	MAT	2014-01-02	47.57	2017-12-29	15.38	-32.19	-67.67
134	DISCK	2014-01-02	83.22	2017-12-29	21.17	-62.05	-74.56
133	DISCA	2014-01-02	90.21	2017-12-29	22.38	-67.83	-75.19
385	RRC	2014-01-02	83.13	2017-12-29	17.06	-66.07	-79.48
91	CHK	2014-01-02	27.07	2017-12-29	3.96	-23.11	-85.37

483 rows × 7 columns

In [29]:

```
#same question as above, but specifically looking at highest increase in absolute share value.
jan_2014_dec_2017_merged.sort_values(by = ["price_difference"], ascending = False)
#Amazon is the winner on absolute share value increase.
```

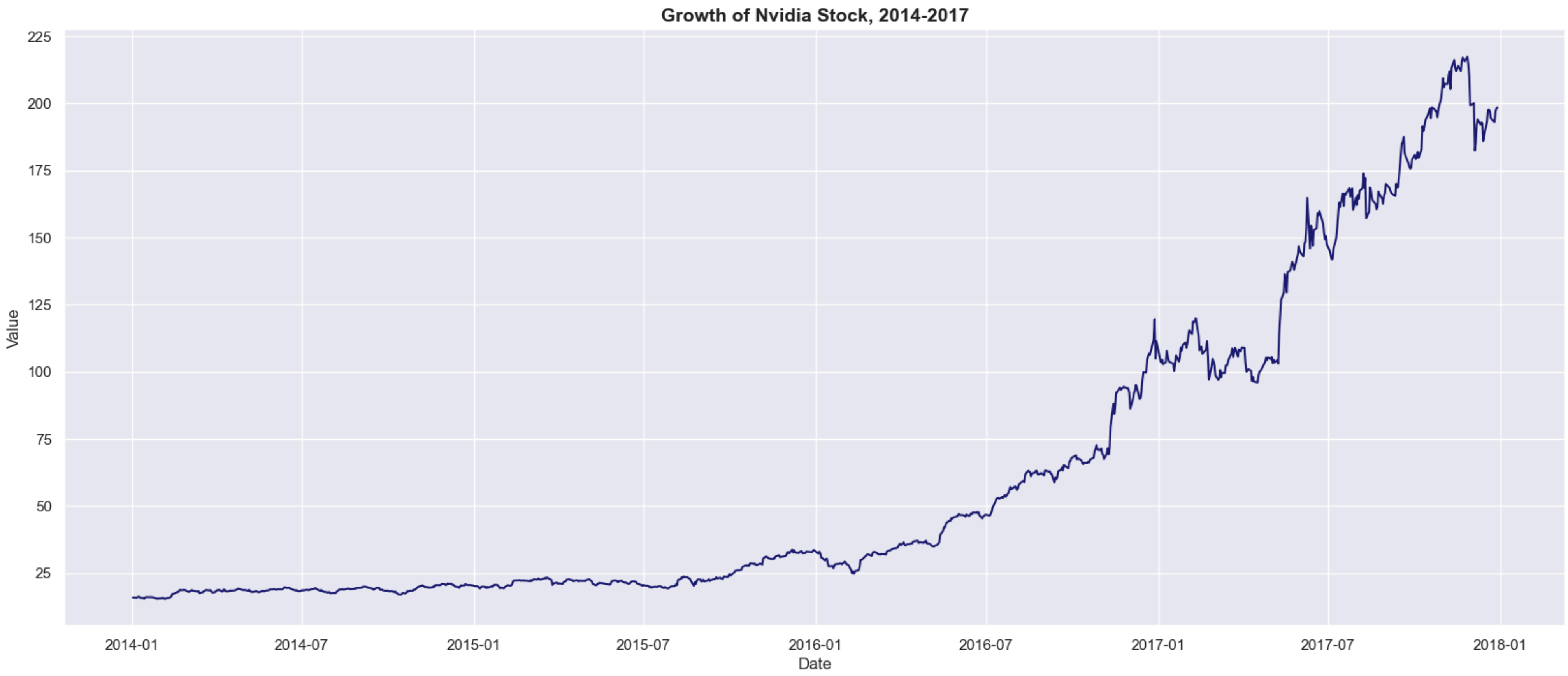
Out[29]:

	company	date_x	open	date_y	close	price_difference	price_difference_pc
38	AMZN	2014-01-02	398.80	2017-12-29	1169.47	770.67	193.25
344	PCLN	2014-01-02	1159.97	2017-12-29	1737.74	577.77	49.81
197	GOOGL	2014-01-02	558.29	2017-12-29	1053.40	495.11	88.68
310	MTD	2014-01-02	241.09	2017-12-29	619.52	378.43	156.97
156	EQIX	2014-01-02	177.63	2017-12-29	453.22	275.59	155.15
...	...	...	...	...	...	...	...
363	PRGO	2014-01-02	152.45	2017-12-29	87.16	-65.29	-42.83
385	RRC	2014-01-02	83.13	2017-12-29	17.06	-66.07	-79.48
133	DISCA	2014-01-02	90.21	2017-12-29	22.38	-67.83	-75.19
380	RL	2014-01-02	175.44	2017-12-29	103.69	-71.75	-40.90
101	CMG	2014-01-02	530.00	2017-12-29	289.03	-240.97	-45.47

483 rows × 7 columns

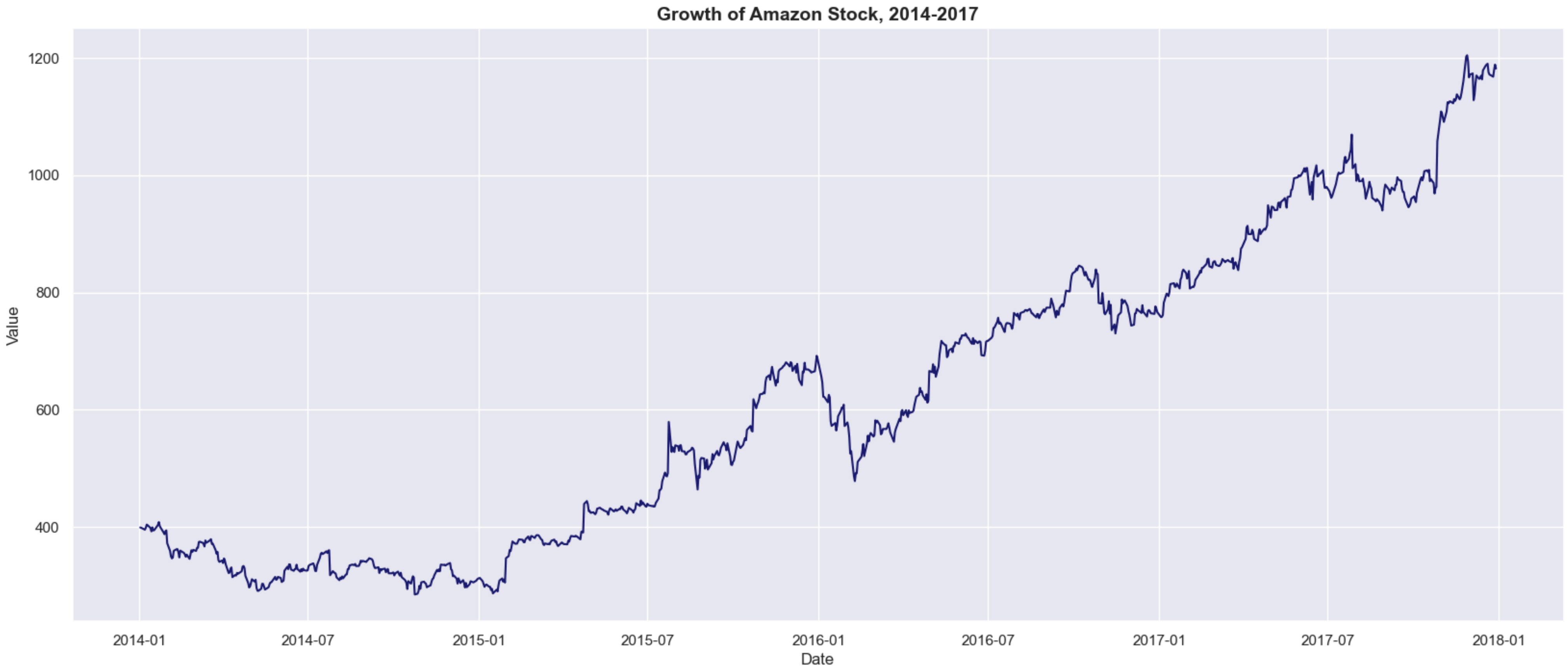
In [30]:

```
#create a line chart of Nvidia stock growth 2014 - 2017
nvidia_growth = data_wip[(data_wip["company"] == "NVDA")]["open"]
nvidia_growth.index = data_wip[(data_wip["company"] == "NVDA")]["date"]

plt.figure(figsize = (20,8))
plt.plot(nvidia_growth, color = "midnightblue")
plt.title("Growth of Nvidia Stock, 2014-2017", fontsize = 14, fontweight = "bold")
plt.ylabel("Value")
plt.xlabel("Date")
plt.show()
```



In [31]:

```
#create a chart of Amazon stock growth 2014 - 2017
amazon_growth = data_wip[(data_wip["company"] == "AMZN")]["open"]
amazon_growth.index = data_wip[(data_wip["company"] == "AMZN")]["date"]

plt.figure(figsize = (20,8))
plt.plot(amazon_growth, color = "midnightblue")
plt.title("Growth of Amazon Stock, 2014-2017", fontsize = 14, fontweight = "bold")
plt.ylabel("Value")
plt.xlabel("Date")
plt.show()
```



In [32]:

```
#calculate average total growth from 2nd Jan 2014 - 29th Dec 2017.
round(jan_2014_dec_2017_merged.sort_values(by = ["price_difference_pc"], ascending = False)["price_difference_pc"].mean(),2)
```

Out[32]:

```
53.32
```