FRA (MILESTONE-2) MARKER RISK

2.1 Draw Stock Price Graph(Stock Price vs Time) for any 2 given stocks with inference

1. Importing the libraries of numpy, pandas, matplotlib, seaborn sklearn metrics.
2. Read the dataset 'Market+Risk+Dataset.csv'.
3. Fixing messy column names for easy using.
4. Glimpse on the head of the dataset.
5. Shape of the dataset.
6. Information on datatypes.
7. Checking basic measures of descriptive statistics.
8. To plot & see price trend over time for i.e., Stock Price garph for any two companies viz., Infosys and Sun-Pharma.

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns # for making plots with seaborn
color = sns.color_palette()
import sklearn.metrics as metrics


import warnings
warnings.filterwarnings("ignore")#### Importing the libraries
```

Importing the dataset

In [2]:

```
stock_prices = pd.read_csv('Market+Risk+Dataset.csv')

#Glimpse of Data
#### Importing the dataset
```

Fixing messy column names (containing spaces) for ease of use

In [3]:

```
stock_prices.columns = stock_prices.columns.str.replace(' ', '_')
```

Checking top 5 rows again

In [4]:

```
stock_prices.head()
```

Out[4]:

	Date	Infosys	Indian_Hotel	Mahindra_&_Mahindra	Axis_Bank	SAIL	Shree_Cement	Sun_Pharma	Jindal_Steel	Idea_Vodafc
0	31-03-2014	264	69	455	263	68	5543	555	298	

	Date	Infosys	Indian_Hotel	Mahindra_&_Mahindra	Axis_Bank	SAIL	Shree_Cement	Sun_Pharma	Jindal_Steel	Idea_Vodafo
1	07-04-2014	257	68	458	276	70	5728	610	279	
2	14-04-2014	254	68	454	270	68	5649	607	279	
3	21-04-2014	253	68	488	283	68	5692	604	274	
4	28-04-2014	256	65	482	282	63	5582	611	238	

First, let us check the number of rows (observations) and the number of columns (variables)

In [5]:

```
print('The number of rows (observations) is',stock_prices.shape[0],'\n''The number of col
umns (variables) is',stock_prices.shape[1])
```

```
The number of rows (observations) is 314
The number of columns (variables) is 11
```

Checking data types of all columns

In [6]:

```
stock_prices.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 314 entries, 0 to 313
Data columns (total 11 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Date                 314 non-null    object
 1   Infosys              314 non-null    int64
 2   Indian_Hotel         314 non-null    int64
 3   Mahindra_&_Mahindra  314 non-null    int64
 4   Axis_Bank            314 non-null    int64
 5   SAIL                 314 non-null    int64
 6   Shree_Cement         314 non-null    int64
 7   Sun_Pharma           314 non-null    int64
 8   Jindal_Steel         314 non-null    int64
 9   Idea_Vodafone        314 non-null    int64
 10  Jet_Airways          314 non-null    int64
dtypes: int64(10), object(1)
memory usage: 27.1+ KB
```

Now, let us check the basic measures of descriptive statistics for the continuous variables

In [7]:

```
stock_prices.describe().T
```

Out[7]:

	count	mean	std	min	25%	50%	75%	max
Infosys	314.0	511.340764	135.952051	234.0	424.00	466.5	630.75	810.0
Indian_Hotel	314.0	114.560510	22.509732	64.0	96.00	115.0	134.00	157.0
Mahindra_&_Mahindra	314.0	636.678344	102.879975	284.0	572.00	625.0	678.00	956.0
Axis_Bank	314.0	540.742038	115.835569	263.0	470.50	528.0	605.25	808.0
SAIL	314.0	59.095541	15.810493	21.0	47.00	57.0	71.75	104.0

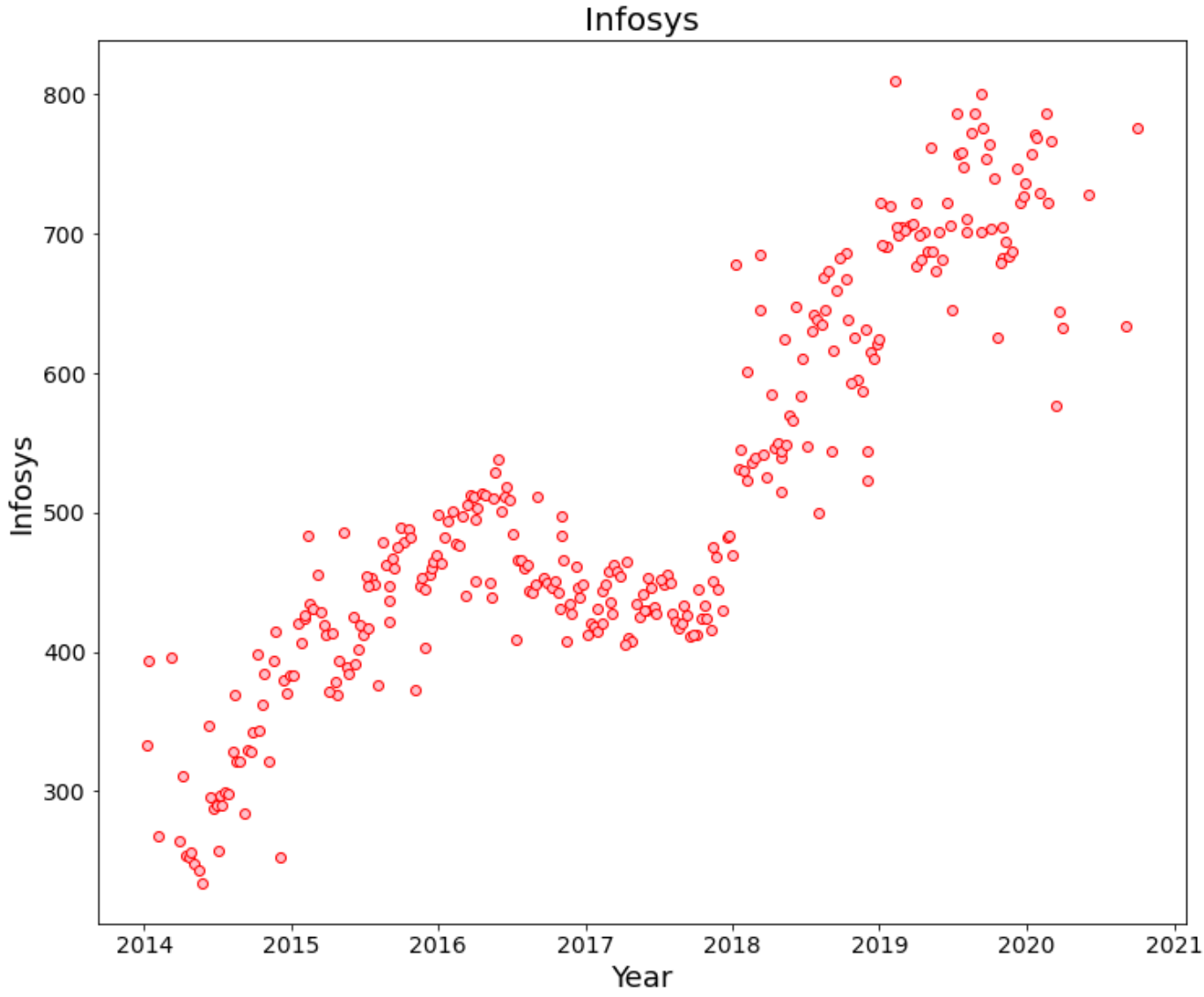
	count	mean	std	min	25%	50%	75%	max
Shree_Cement	314.0	14806.410828	4288.275085	5543.0	10952.25	16018.5	17773.25	24806.0
Sun_Pharma	314.0	633.468153	171.855893	338.0	478.50	614.0	785.00	1089.0
Jindal_Steel	314.0	147.627389	65.879195	53.0	88.25	142.5	182.75	338.0
Idea_Vodafone	314.0	53.713376	31.248985	3.0	25.25	53.0	82.00	117.0
Jet_Airways	314.0	372.659236	202.262668	14.0	243.25	376.0	534.00	871.0

In [8]:

```
#### To plot & see price trend over time for different companies
```

In [9]:

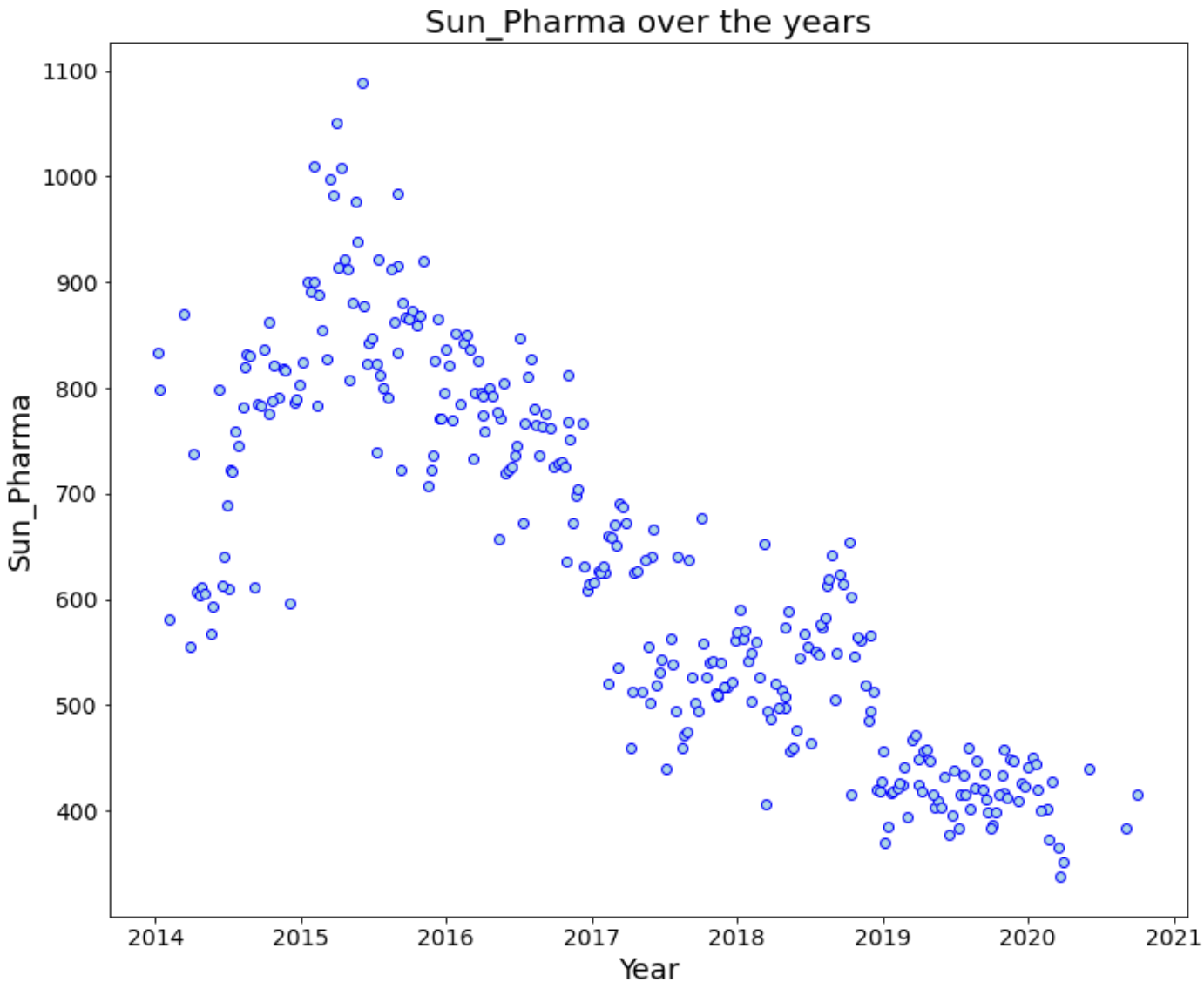
```
plt.figure(figsize = (12, 10))
stock_prices['dates'] = [pd.to_datetime(d) for d in stock_prices['Date']]
plt.scatter(stock_prices['dates'], stock_prices['Infosys'], edgecolors='r', color = 'pin
k')
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.xlabel('Year', fontsize=18)
plt.ylabel('Infosys', fontsize=18)
plt.title('Infosys', fontsize=20)
plt.show()
```



In [10]:

```
plt.figure(figsize = (12, 10))
stock_prices['dates'] = [pd.to_datetime(d) for d in stock_prices['Date']]
plt.scatter(stock_prices['dates'], stock_prices['Sun_Pharma'], edgecolors='b', color = '
lightblue')
plt.xticks(fontsize=14)
```

```
plt.yticks(fontsize=14)
plt.xlabel('Year', fontsize=18)
plt.ylabel('Sun_Pharma', fontsize=18)
plt.title('Sun_Pharma over the years', fontsize=20)
plt.show()
```



2.2 Calculate Returns for all stocks with inference

The logarithmic returns are a difference between two consecutive week prices.

Analyzing returns

Steps for calculating returns from prices:

- **Take logarithms**
- **Take differences**

In [11]:

```
stock_returns = np.log(stock_prices.drop(['Date','dates'],axis=1)).diff(axis = 0, period
s = 1)
```

Checking the rows & columns of dataset

In [12]:

```
stock_returns.shape
```

Out[12]:

(314, 10)

Checking top 5 rows

In [13]:

```
stock_returns.head()
```

Out[13]:

	Infosys	Indian_Hotel	Mahindra_&_Mahindra	Axis_Bank	SAIL	Shree_Cement	Sun_Pharma	Jindal_Steel	Idea_Vodafo
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1	-0.026873	-0.014599	0.006572	0.048247	0.028988	0.032831	0.094491	-0.065882	0.0119
2	-0.011742	0.000000	-0.008772	-0.021979	-0.028988	-0.013888	-0.004930	0.000000	-0.0119
3	-0.003945	0.000000	0.072218	0.047025	0.000000	0.007583	-0.004955	-0.018084	0.0000
4	0.011788	-0.045120	-0.012371	-0.003540	-0.076373	-0.019515	0.011523	-0.140857	-0.0493

2.3 Calculate Stock Means and Standard Deviation for all stocks with inference

In [ ]:

```

```

We now look at Means & Standard Deviations of these returns

- **Stock Means: Average returns that the stock is making on a week to week basis**
- **Stock Standard Deviation : It is a measure of volatility meaning the more a stock's returns vary from the stock's average return, the more volatile the stock**

Calculating stock means

In [ ]:

```

```

In [14]:

```
stock_means = stock_returns.mean(axis = 0)
stock_means
```

Out[14]:

```
Infosys                0.002794
Indian_Hotel           0.000266
Mahindra_&_Mahindra   -0.001506
Axis_Bank              0.001167
SAIL                  -0.003463
Shree_Cement           0.003681
Sun_Pharma            -0.001455
Jindal_Steel          -0.004123
Idea_Vodafone         -0.010608
Jet_Airways           -0.009548
dtype: float64
```

Calculating stock standard deviation

In [15]:

```
stock_sd = stock_returns.std(axis = 0)
stock_sd
```

Out[15]:

```
Infosys                 0.035070
Indian_Hotel            0.047131
Mahindra_&_Mahindra     0.040169
Axis_Bank               0.045828
SAIL                    0.062188
Shree_Cement            0.039917
Sun_Pharma              0.045033
Jindal_Steel            0.075108
Idea_Vodafone           0.104315
Jet_Airways             0.097972
dtype: float64
```

In [16]:

```
df = pd.DataFrame({'Average':stock_means, 'Volatility': stock_sd})
df
```

Out[16]:

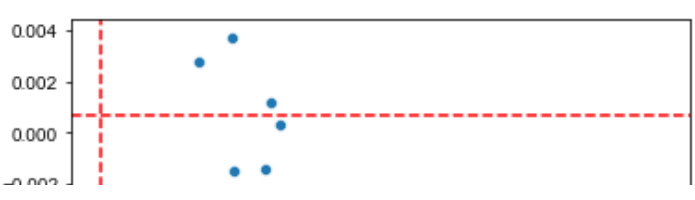
	Average	Volatility
Infosys	0.002794	0.035070
Indian_Hotel	0.000266	0.047131
Mahindra_&_Mahindra	-0.001506	0.040169
Axis_Bank	0.001167	0.045828
SAIL	-0.003463	0.062188
Shree_Cement	0.003681	0.039917
Sun_Pharma	-0.001455	0.045033
Jindal_Steel	-0.004123	0.075108
Idea_Vodafone	-0.010608	0.104315
Jet_Airways	-0.009548	0.097972

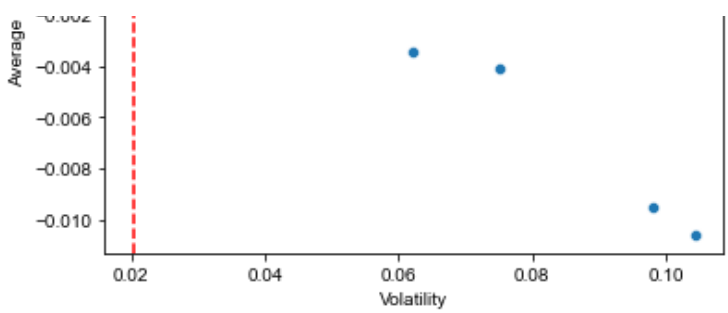
2.4 Draw a plot of Stock Means vs Standard Deviation and state your inference

Let us plot & see what they say about stock prices

In [17]:

```
#plt.scatter(stock_sd, stock_means, edgecolors='r')
plot = sns.scatterplot(df['Volatility'], df['Average'])
plot.axvline(x=0.020257,linestyle='--', color = "red")
plot.axhline(y=0.000683,linestyle='--', color = "red")
sns.set(font_scale = 1)
plt.show()
```





2.5 Conclusion and Recommendations

Based on the average and volatility as well as evident from plot Shree_Cement, Infosys and Axis_bank have high returns on positive side. Among these Shree_Cement is having low risk followed by Infosys. CONCLUSION: Low risk and high returns are considered as best stocks and for the data provided, Shree_Cement is best followed by Infosys, Axix_Bank.

In [ ]: