In [2]:
```
import numpy as np
import pandas as pd
import datetime


%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
```

In [15]:
```
df = pd.read_csv("sap.csv")
df.head(20)
```

Out[15]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2016-04-27	80.379997	81.190002	80.199997	81.019997	75.324928	1350900
1	2016-04-28	79.519997	80.190002	79.199997	79.370003	73.790909	982700
2	2016-04-29	78.739998	78.830002	78.169998	78.620003	73.093628	878700
3	2016-05-02	79.680000	79.860001	79.150002	79.750000	74.144196	893600
4	2016-05-03	78.820000	79.010002	78.180000	78.519997	73.000656	710300
5	2016-05-04	77.620003	77.680000	77.169998	77.470001	72.024467	1121700
6	2016-05-05	76.800003	77.750000	76.669998	77.070000	71.652580	1288100
7	2016-05-06	76.989998	77.440002	76.790001	77.330002	71.894310	467300
8	2016-05-09	78.410004	78.699997	78.150002	78.309998	72.805420	933500
9	2016-05-10	76.860001	77.629997	76.849998	77.589996	73.363281	620900
10	2016-05-11	77.000000	77.489998	76.849998	76.860001	72.673042	974100
11	2016-05-12	77.889999	78.040001	76.870003	77.250000	73.041801	559100
12	2016-05-13	77.089996	77.419998	76.739998	77.029999	72.833786	1193300
13	2016-05-16	77.230003	78.120003	77.150002	77.830002	73.590210	595600
14	2016-05-17	77.489998	77.669998	76.699997	76.959999	72.767601	956100
15	2016-05-18	77.000000	78.169998	76.860001	77.519997	73.297089	660900
16	2016-05-19	77.110001	77.320000	76.550003	76.800003	72.616318	801000
17	2016-05-20	77.660004	77.959999	77.470001	77.720001	73.486198	883400
18	2016-05-23	78.269997	78.580002	77.900002	77.949997	73.703659	730600
19	2016-05-24	79.110001	79.730003	79.080002	79.550003	75.216515	855500

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1258 entries, 0 to 1257
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       1258 non-null   object 
 1   Open       1258 non-null   float64
 2   High       1258 non-null   float64
 3   Low        1258 non-null   float64
 4   Close      1258 non-null   float64
 5   Adj Close  1258 non-null   float64
 6   Volume     1258 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 68.9+ KB
```

In [5]:
```
df.describe()
```

Out[5]:

	Open	High	Low	Close	Adj Close	Volume
count	1258.000000	1258.000000	1258.000000	1258.000000	1258.000000	1.258000e+03
mean	113.877051	114.605533	113.128331	113.908013	110.992397	8.590167e+05
std	19.456975	19.617020	19.234336	19.433727	20.830699	6.328945e+05
min	72.349998	72.360001	71.389999	72.019997	68.096703	1.179000e+05
25%	103.049999	103.922499	102.412503	103.314999	99.348825	5.167750e+05
50%	113.420002	113.990002	112.645001	113.279999	109.620148	7.136500e+05
75%	125.545002	126.559998	124.590001	125.750000	124.164991	9.943500e+05
max	168.089996	169.300003	166.289993	169.020004	169.020004	1.129020e+07

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

Out[6]:
```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')
```

In [7]:

```
import matplotlib.dates as mdates

years = mdates.YearLocator() # Get every year
yearsFmt = mdates.DateFormatter('%Y') # Set year format

# Create subplots to plot graph and control axes
fig, ax = plt.subplots()
ax.plot(df['Date'], df['Close'])

# Format the ticks
ax.xaxis.set_major_locator(years)
ax.xaxis.set_major_formatter(yearsFmt)

# Set figure title
plt.title('Close Stock Price History [2009 - 2019]', fontsize=16)
# Set x label
plt.xlabel('Date', fontsize=14)
# Set y label
plt.ylabel('Closing Stock Price in $', fontsize=14)

# Rotate and align the x labels
fig.autofmt_xdate()

# Show plot
plt.show()
```


Close Stock Price History [2009 - 2019]
Closing Stock Price in $
160
140
120
100
80
1970
1971
1972
1973
Date

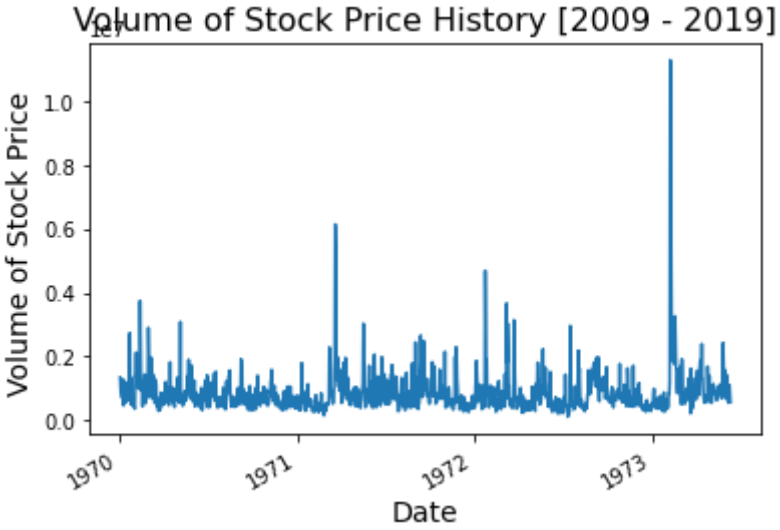
In [20]:

```
import matplotlib.dates as mdates

years = mdates.YearLocator() # Get every year
yearsFmt = mdates.DateFormatter('%Y') # Set year format

# Create subplots to plot graph and control axes
fig, ax = plt.subplots()
ax.plot(df['Date'], df['Open'])

# Format the ticks
ax.xaxis.set_major_locator(years)
ax.xaxis.set_major_formatter(yearsFmt)

# Set figure title
plt.title('Open Stock Price History [2009 - 2019]', fontsize=16)
# Set x label
plt.xlabel('Date', fontsize=14)
# Set y label
plt.ylabel('Opening Stock Price in $', fontsize=14)

# Rotate and align the x labels
fig.autofmt_xdate()

# Show plot
plt.show()
```


Open Stock Price History [2009 - 2019]
Opening Stock Price in $
160
140
120
100
80
1970
1971
1972
1973
Date

In [21]:

```
import matplotlib.dates as mdates

years = mdates.YearLocator() # Get every year
yearsFmt = mdates.DateFormatter('%Y') # Set year format

# Create subplots to plot graph and control axes
fig, ax = plt.subplots()
ax.plot(df['Date'], df['Volume'])

# Format the ticks
ax.xaxis.set_major_locator(years)
ax.xaxis.set_major_formatter(yearsFmt)

# Set figure title
plt.title('Volume of Stock Price History [2009 - 2019]', fontsize=16)
# Set x label
plt.xlabel('Date', fontsize=14)
# Set y label
plt.ylabel('Volume of Stock Price ', fontsize=14)

# Rotate and align the x labels
fig.autofmt_xdate()

# Show plot
plt.show()
```


Volume of Stock Price History [2009 - 2019]
1e7
1.0
0.8
0.6
0.4
0.2
0.0
Volume of Stock Price
1970
1971
1972
1973
Date

In [22]:

```
import matplotlib.dates as mdates

years = mdates.YearLocator() # Get every year
yearsFmt = mdates.DateFormatter('%Y') # Set year format

# Create subplots to plot graph and control axes
fig, ax = plt.subplots()
ax.plot(df['Date'], df['High'])

# Format the ticks
ax.xaxis.set_major_locator(years)
ax.xaxis.set_major_formatter(yearsFmt)

# Set figure title
plt.title('Highest Stock Price History [2009 - 2019]', fontsize=16)
# Set x label
plt.xlabel('Date', fontsize=14)
# Set y label
plt.ylabel('Highest Stock Price in $', fontsize=14)

# Rotate and align the x labels
fig.autofmt_xdate()

# Show plot
plt.show()
```


Highest Stock Price History [2009 - 2019]
Highest Stock Price in $
160
140
120
100
80
1970
1971
1972
1973
Date

In [23]:

```
import matplotlib.dates as mdates

years = mdates.YearLocator() # Get every year
yearsFmt = mdates.DateFormatter('%Y') # Set year format

# Create subplots to plot graph and control axes
fig, ax = plt.subplots()
ax.plot(df['Date'], df['Low'])

# Format the ticks
ax.xaxis.set_major_locator(years)
ax.xaxis.set_major_formatter(yearsFmt)

# Set figure title
plt.title(' Lowest Price History [2009 - 2019]', fontsize=16)
# Set x label
plt.xlabel('Date', fontsize=14)
# Set y label
plt.ylabel('LOwest Stock Price in $', fontsize=14)

# Rotate and align the x labels
fig.autofmt_xdate()

# Show plot
plt.show()
```

In [18]:
```
from sklearn.model_selection import train_test_split
# Split data into train and test set: 80% / 20%
train, test = train_test_split(df, test_size=0.20)
```

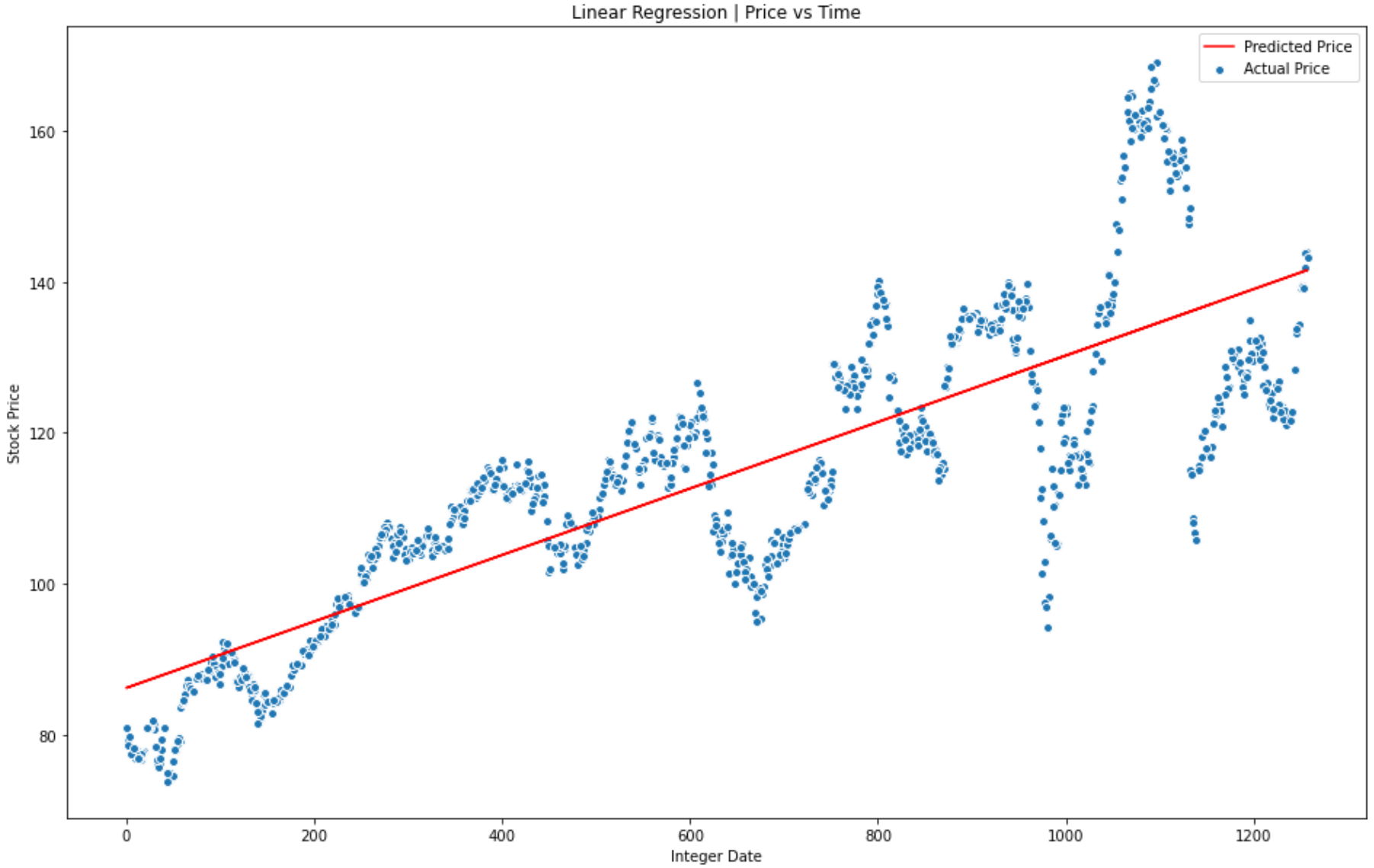
In [9]:
```
from sklearn.linear_model import LinearRegression
# Reshape index column to 2D array for .fit() method
X_train = np.array(train.index).reshape(-1, 1)
y_train = train['Close']
# Create LinearRegression Object
model = LinearRegression()
# Fit linear model using the train data set
model.fit(X_train, y_train)
```

Out[9]:
```
LinearRegression()
```

In [24]:
```
# The coefficient
print('Slope: ', np.asscalar(np.squeeze(model.coef_)))
# The Intercept
print('Intercept: ', model.intercept_)
```

```
Slope:  0.043958127564923774
Intercept:  86.26105402827619

C:\Users\SATHIS~1\AppData\Local\Temp/ipykernel_18732/4094384673.py:2: DeprecationWarning: np.asscalar(a) is deprecated
since NumPy v1.16, use a.item() instead
  print('Slope: ', np.asscalar(np.squeeze(model.coef_)))
```

In [11]:
```
# Train set graph
plt.figure(1, figsize=(16,10))
plt.title('Linear Regression | Price vs Time')
plt.scatter(X_train, y_train, edgecolor='w', label='Actual Price')
plt.plot(X_train, model.predict(X_train), color='r', label='Predicted Price')
plt.xlabel('Integer Date')
plt.ylabel('Stock Price')
plt.legend()
plt.show()
```


Linear Regression | Price vs Time
Predicted Price
Actual Price
Stock Price
Integer Date
160
140
120
100
80
0
200
400
600
800
1000
1200

In [ ]: