In [1]:
```
!pip install -q yfinance
```

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

import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
plt.style.use("fivethirtyeight")
%matplotlib inline

# For reading stock data from yahoo
from pandas_datareader.data import DataReader
import yfinance as yf
from pandas_datareader import data as pdr

yf.pdr_override()

# For time stamps
from datetime import datetime


# The tech stocks we'll use for this analysis
tech_list = ['AAPL', 'GOOG', 'MSFT', 'AMZN']

# Set up End and Start times for data grab
tech_list = ['AAPL', 'GOOG', 'MSFT', 'AMZN']

end = datetime.now()
start = datetime(end.year - 1, end.month, end.day)

for stock in tech_list:
    globals()[stock] = yf.download(stock, start, end)


company_list = [AAPL, GOOG, MSFT, AMZN]
company_name = ["APPLE", "GOOGLE", "MICROSOFT", "AMAZON"]

for company, com_name in zip(company_list, company_name):
    company["company_name"] = com_name

df = pd.concat(company_list, axis=0)
df.tail(10)
```

```
[**********100%%**********]  1 of 1 completed
[**********100%%**********]  1 of 1 completed
[**********100%%**********]  1 of 1 completed
[**********100%%**********]  1 of 1 completed
```

Out[2]:

Date	Open	High	Low	Close	Adj Close	Volume	company_name
2023-08-25	132.470001	133.869995	130.580002	133.259995	133.259995	44147500	AMAZON
2023-08-28	133.779999	133.949997	131.850006	133.139999	133.139999	34108400	AMAZON
2023-08-29	133.380005	135.139999	133.250000	134.910004	134.910004	38646100	AMAZON

Date	Open	High	Low	Close	Adj Close	Volume	company_name
2023-08-30	134.929993	135.679993	133.919998	135.070007	135.070007	36137000	AMAZON
2023-08-31	135.059998	138.789993	135.000000	138.009995	138.009995	58781300	AMAZON
2023-09-01	139.460007	139.960007	136.880005	138.119995	138.119995	40948300	AMAZON
2023-09-05	137.729996	137.800003	135.820007	137.270004	137.270004	40636700	AMAZON
2023-09-06	136.320007	137.449997	134.610001	135.360001	135.360001	41785500	AMAZON
2023-09-07	133.899994	138.029999	133.160004	137.850006	137.850006	48498900	AMAZON
2023-09-08	136.860001	138.850006	136.750000	138.229996	138.229996	38348200	AMAZON

In [3]:
```
AAPL.describe()
```

Out[3]:

	Open	High	Low	Close	Adj Close	Volume
count	250.000000	250.000000	250.000000	250.000000	250.000000	2.500000e+02
mean	160.466081	162.256080	158.889280	160.652120	160.186888	6.948214e+07
std	19.014822	18.772899	19.275469	19.010060	19.188538	2.411895e+07
min	126.010002	127.769997	124.169998	125.019997	124.488876	3.145820e+07
25%	145.812500	147.320004	143.957497	145.915001	145.256302	5.126040e+07
50%	154.930000	157.440002	153.415001	155.320000	154.699356	6.462460e+07
75%	177.327503	179.109997	176.537502	177.412498	177.394997	8.116722e+07
max	196.240005	198.229996	195.279999	196.449997	196.185074	1.647624e+08

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

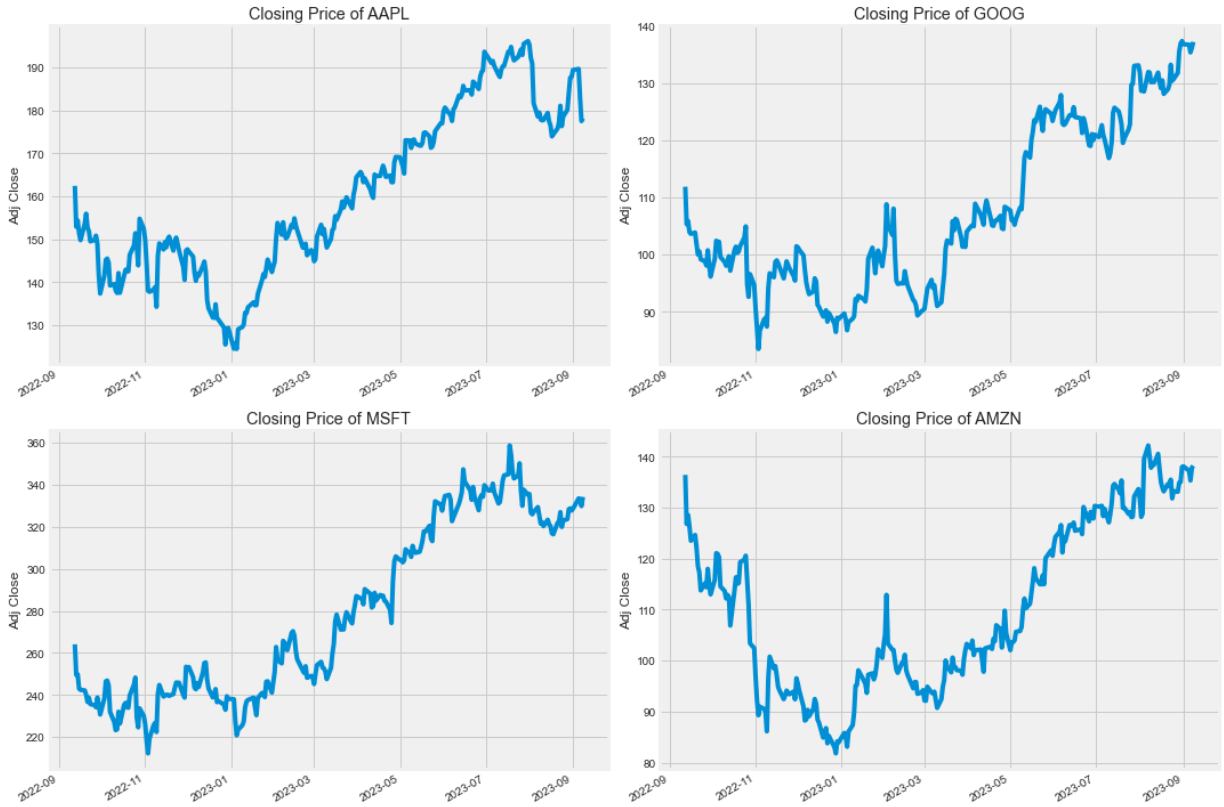
```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 250 entries, 2022-09-12 to 2023-09-08
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Open          250 non-null    float64
 1   High          250 non-null    float64
 2   Low           250 non-null    float64
 3   Close         250 non-null    float64
 4   Adj Close     250 non-null    float64
 5   Volume        250 non-null    int64  
 6   company_name  250 non-null    object 
dtypes: float64(5), int64(1), object(1)
memory usage: 15.6+ KB
```

In [5]:
```
# Let's see a historical view of the closing price
plt.figure(figsize=(15, 10))
```

```
plt.subplots_adjust(top=1.25, bottom=1.2)

for i, company in enumerate(company_list, 1):
    plt.subplot(2, 2, i)
    company['Adj Close'].plot()
    plt.ylabel('Adj Close')
    plt.xlabel(None)
    plt.title(f"Closing Price of {tech_list[i - 1]}")

plt.tight_layout()
```

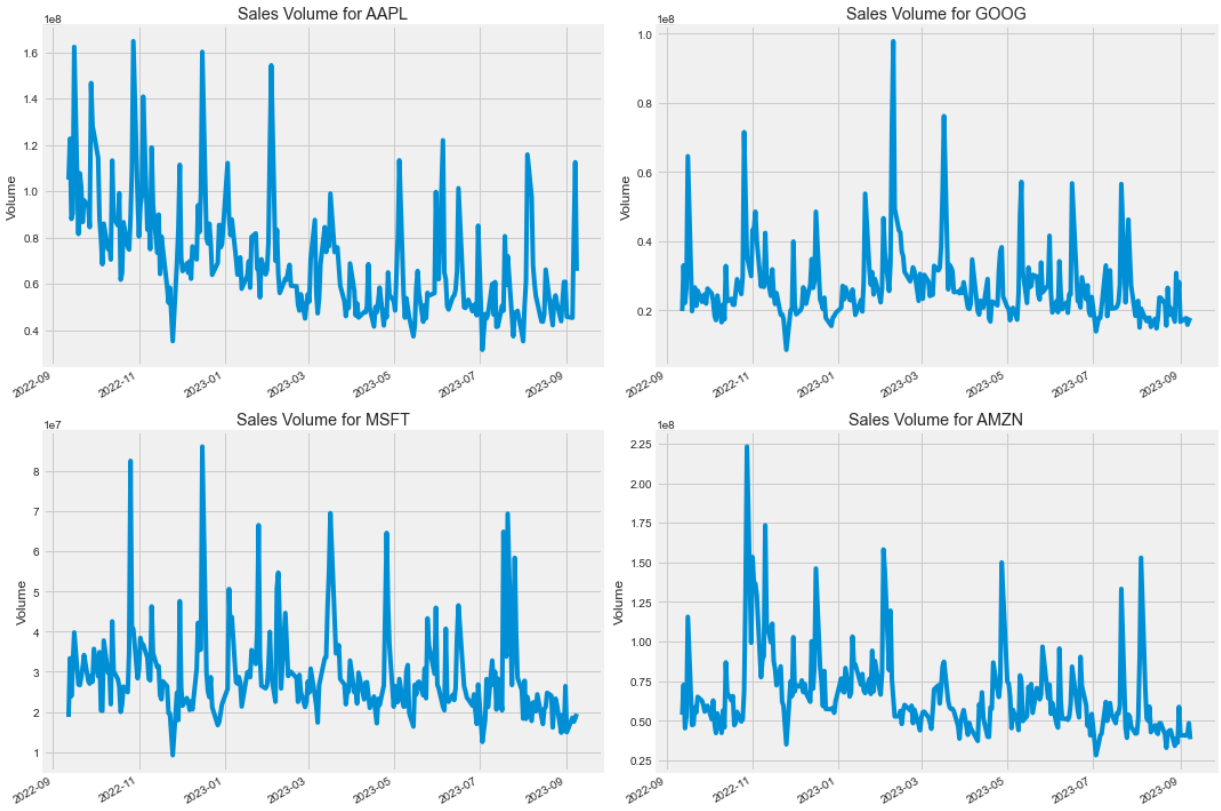


In [6]:

```
# Now let's plot the total volume of stock being traded each day
plt.figure(figsize=(15, 10))
plt.subplots_adjust(top=1.25, bottom=1.2)

for i, company in enumerate(company_list, 1):
    plt.subplot(2, 2, i)
    company['Volume'].plot()
    plt.ylabel('Volume')
    plt.xlabel(None)
    plt.title(f"Sales Volume for {tech_list[i - 1]}")

plt.tight_layout()
```



In [7]:
```
# Get the stock quote
df = pdr.get_data_yahoo('AAPL', start='2012-01-01', end=datetime.now())
# Show teh data
df
```

```
[**********100%%**********]  1 of 1 completed
```

Out[7]:

Date	Open	High	Low	Close	Adj Close	Volume
2012-01-03	14.621429	14.732143	14.607143	14.686786	12.466094	302220800
2012-01-04	14.642857	14.810000	14.617143	14.765714	12.533086	260022000
2012-01-05	14.819643	14.948214	14.738214	14.929643	12.672228	271269600
2012-01-06	14.991786	15.098214	14.972143	15.085714	12.804701	318292800
2012-01-09	15.196429	15.276786	15.048214	15.061786	12.784392	394024400
...	...	...	...	...	...	...
2023-09-01	189.490005	189.919998	188.279999	189.460007	189.460007	45732600
2023-09-05	188.279999	189.979996	187.610001	189.699997	189.699997	45280000
2023-09-06	188.399994	188.850006	181.470001	182.910004	182.910004	81755800
2023-09-07	175.179993	178.210007	173.539993	177.559998	177.559998	112488800
2023-09-08	178.350006	180.240005	177.789993	178.179993	178.179993	65551300

2940 rows × 6 columns

In [8]:
```
plt.figure(figsize=(16,6))
plt.title('Close Price History')
plt.plot(df['Close'])
plt.xlabel('Date', fontsize=18)
```

```
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.show()
```

In [9]:
```
# Create a new dataframe with only the 'Close column
data = df.filter(['Close'])
# Convert the dataframe to a numpy array
dataset = data.values
# Get the number of rows to train the model on
training_data_len = int(np.ceil( len(dataset) * .95 ))

training_data_len
```

Out[9]:
```
2793
```

In [10]:
```
# Scale the data
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(dataset)

scaled_data
```

Out[10]:
```
array([[0.00405082],
       [0.0044833 ],
       [0.00538153],
       ...,
       [0.92580927],
       [0.89649457],
       [0.89989176]])
```

In [11]:
```
# Create the training data set
# Create the scaled training data set
train_data = scaled_data[0:int(training_data_len), :]
# Split the data into x_train and y_train data sets
x_train = []
y_train = []

for i in range(60, len(train_data)):
    x_train.append(train_data[i-60:i, 0])
    y_train.append(train_data[i, 0])
    if i<= 61:
        print(x_train)
        print(y_train)
        print()

# Convert the x_train and y_train to numpy arrays
```

```
x_train, y_train = np.array(x_train), np.array(y_train)

# Reshape the data
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
# x_train.shape
```

```
[array([0.00405082, 0.0044833 , 0.00538153, 0.0062367 , 0.00610559,
       0.00640108, 0.00626606, 0.00603905, 0.00572986, 0.0066868 ,
       0.0075498 , 0.00728366, 0.00582575, 0.00721712, 0.00584728,
       0.01098419, 0.01058694, 0.01110552, 0.01222684, 0.01290588,
       0.01284914, 0.01263975, 0.0135321 , 0.01437162, 0.01532269,
       0.01685887, 0.02008583, 0.02013475, 0.02193121, 0.02327365,
       0.02096645, 0.02185489, 0.02183728, 0.02432844, 0.02397423,
       0.02462979, 0.02580786, 0.02646344, 0.02835186, 0.02972757,
       0.03012483, 0.03026377, 0.02791156, 0.02734404, 0.0274282 ,
       0.02963952, 0.03026182, 0.0315984 , 0.03474903, 0.0389525 ,
       0.03816582, 0.03816777, 0.04120687, 0.04215794, 0.04148084,
       0.04086246, 0.04021863, 0.04235754, 0.04382523, 0.04443971])]
[0.04292113229660477]

[array([0.00405082, 0.0044833 , 0.00538153, 0.0062367 , 0.00610559,
       0.00640108, 0.00626606, 0.00603905, 0.00572986, 0.0066868 ,
       0.0075498 , 0.00728366, 0.00582575, 0.00721712, 0.00584728,
       0.01098419, 0.01058694, 0.01110552, 0.01222684, 0.01290588,
       0.01284914, 0.01263975, 0.0135321 , 0.01437162, 0.01532269,
       0.01685887, 0.02008583, 0.02013475, 0.02193121, 0.02327365,
       0.02096645, 0.02185489, 0.02183728, 0.02432844, 0.02397423,
       0.02462979, 0.02580786, 0.02646344, 0.02835186, 0.02972757,
       0.03012483, 0.03026377, 0.02791156, 0.02734404, 0.0274282 ,
       0.02963952, 0.03026182, 0.0315984 , 0.03474903, 0.0389525 ,
       0.03816582, 0.03816777, 0.04120687, 0.04215794, 0.04148084,
       0.04086246, 0.04021863, 0.04235754, 0.04382523, 0.04443971]), array([0.004483
3 , 0.00538153, 0.0062367 , 0.00610559, 0.00640108,
       0.00626606, 0.00603905, 0.00572986, 0.0066868 , 0.0075498 ,
       0.00728366, 0.00582575, 0.00721712, 0.00584728, 0.01098419,
       0.01058694, 0.01110552, 0.01222684, 0.01290588, 0.01284914,
       0.01263975, 0.0135321 , 0.01437162, 0.01532269, 0.01685887,
       0.02008583, 0.02013475, 0.02193121, 0.02327365, 0.02096645,
       0.02185489, 0.02183728, 0.02432844, 0.02397423, 0.02462979,
       0.02580786, 0.02646344, 0.02835186, 0.02972757, 0.03012483,
       0.03026377, 0.02791156, 0.02734404, 0.0274282 , 0.02963952,
       0.03026182, 0.0315984 , 0.03474903, 0.0389525 , 0.03816582,
       0.03816777, 0.04120687, 0.04215794, 0.04148084, 0.04086246,
       0.04021863, 0.04235754, 0.04382523, 0.04443971, 0.04292113])]
[0.04292113229660477, 0.04090355083781154]
```

conda install "numpy>=1.16.5,<1.23.0"

In [12]:
```
pip install "numpy>=1.16.5,<1.23.0"
```

```
Requirement already satisfied: numpy<1.23.0,>=1.16.5 in c:\users\dell\anaconda3\lib
\site-packages (1.22.4)
Note: you may need to restart the kernel to use updated packages.
```

In [13]:
```
from keras.models import Sequential
from keras.layers import Dense, LSTM

# Build the LSTM model
model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape= (x_train.shape[1], 1)))
model.add(LSTM(64, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
```

```
# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(x_train, y_train, batch_size=1, epochs=1)
```

```
2733/2733 [==============================] - 142s 49ms/step - loss: 0.0010
```

Out[13]:
```
<keras.src.callbacks.History at 0x1c2038d90a0>
```

In [14]:
```
# Create the testing data set
# Create a new array containing scaled values from index 1543 to 2002
test_data = scaled_data[training_data_len - 60: , :]
# Create the data sets x_test and y_test
x_test = []
y_test = dataset[training_data_len:, :]
for i in range(60, len(test_data)):
    x_test.append(test_data[i-60:i, 0])

# Convert the data to a numpy array
x_test = np.array(x_test)

# Reshape the data
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1 ))

# Get the models predicted price values
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)

# Get the root mean squared error (RMSE)
rmse = np.sqrt(np.mean(((predictions - y_test) ** 2)))
rmse
```

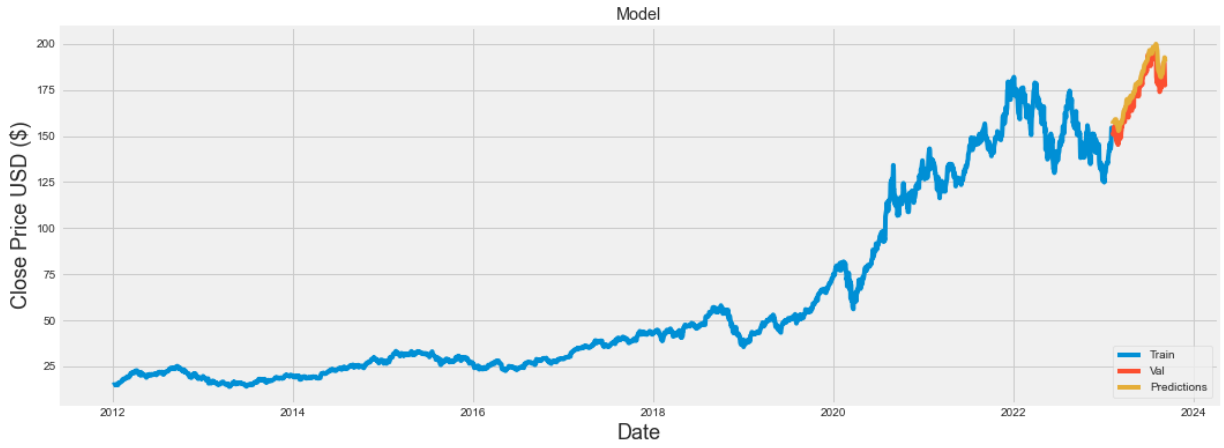
```
5/5 [==============================] - 2s 51ms/step
```

Out[14]:
```
5.881766016648926
```

In [15]:
```
# Plot the data
train = data[:training_data_len]
valid = data[training_data_len:]
valid['Predictions'] = predictions
# Visualize the data
plt.figure(figsize=(16,6))
plt.title('Model')
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
plt.legend(['Train', 'Val', 'Predictions'], loc='lower right')
plt.show()
```

```
C:\Users\DELL\AppData\Local\Temp/ipykernel_18712/2388977846.py:4: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  valid['Predictions'] = predictions
```



In [16]:
```
# Show the valid and predicted prices
valid
```

Out[16]:

Date	Close	Predictions
2023-02-08	151.919998	156.669418
2023-02-09	150.869995	157.516235
2023-02-10	151.009995	157.660507
2023-02-13	153.850006	157.484482
2023-02-14	153.199997	157.714539
...	...	...
2023-09-01	189.460007	189.393814
2023-09-05	189.699997	191.195389
2023-09-06	182.910004	192.622147
2023-09-07	177.559998	192.315704
2023-09-08	178.179993	190.360458

147 rows × 2 columns

In [ ]: