1. What was the change in price of the stock overtime?

```
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]:

	Open	High	Low	Close	Adj Close	Volume	company_name
Date							
2023-01-17 00:00:00-05:00	98.680000	98.889999	95.730003	96.050003	96.050003	72755000	AMAZON
2023-01-18 00:00:00-05:00	97.250000	99.320000	95.379997	95.459999	95.459999	79570400	AMAZON
2023-01-19 00:00:00-05:00	94.739998	95.440002	92.860001	93.680000	93.680000	69002700	AMAZON
2023-01-20	93.860001	97.349998	93.199997	97.250000	97.250000	67307100	AMAZON

	Open	High	Low	Close	Adj Close	Volume	company_name
Date							
00:00:00-05:00							
2023-01-23 00:00:00-05:00	97.559998	97.779999	95.860001	97.519997	97.519997	76501100	AMAZON
2023-01-24 00:00:00-05:00	96.930000	98.089996	96.000000	96.320000	96.320000	66929500	AMAZON
2023-01-25 00:00:00-05:00	92.559998	97.239998	91.519997	97.180000	97.180000	94261600	AMAZON
2023-01-26 00:00:00-05:00	98.239998	99.489998	96.919998	99.220001	99.220001	68523600	AMAZON
2023-01-27 00:00:00-05:00	99.529999	103.489998	99.529999	102.239998	102.239998	87678100	AMAZON
2023-01-30 00:00:00-05:00	101.089996	101.739998	99.010002	100.550003	100.550003	70566100	AMAZON

Reviewing the content of our data, we can see that the data is numeric and the date is the index of the data. Notice also that weekends are missing from the records.

Quick note: Using globals() is a sloppy way of setting the DataFrame names, but it's simple. Now we have our data, let's perform some basic data analysis and check our data.

Descriptive Statistics about the Data

.describe() generates descriptive statistics. Descriptive statistics include those that summarize the central tendency, dispersion, and shape of a dataset’s distribution, excluding NaN values.

Analyzes both numeric and object series, as well as DataFrame column sets of mixed data types. The output will vary depending on what is provided. Refer to the notes below for more detail.

In [3]:

 # Summary Stats

AAPL.describe()

Out[3]:

	Open	High	Low	Close	Adj Close	Volume
count	251.000000	251.000000	251.000000	251.000000	251.000000	2.510000e+02
mean	152.117251	154.227052	150.098406	152.240797	151.861737	8.545738e+07
std	13.239204	13.124055	13.268053	13.255593	13.057870	2.257398e+07
min	126.010002	127.769997	124.169998	125.019997	125.019997	3.519590e+07
25%	142.110001	143.854996	139.949997	142.464996	142.190201	7.027710e+07
50%	150.089996	151.990005	148.199997	150.649994	150.400497	8.100050e+07
75%	163.434998	165.835007	160.879997	163.629997	163.200417	9.374540e+07
max	178.550003	179.610001	176.699997	178.960007	178.154037	1.826020e+08

We have only 255 records in one year because weekends are not included in the data.

Information About the Data

.info() method prints information about a DataFrame including the index dtype and columns, non-null values, and memory usage.

In [4]:

 # General info

AAPL.info()

```
<class 'pandas.core.frame.DataFrame'>

DatetimeIndex: 251 entries, 2022-01-31 00:00:00-05:00 to 2023-01-30 00:00:00-05:00

Data columns (total 7 columns):

 #   Column     Non-Null Count  Dtype

---  ------     --------------  -----

 0   Open       251 non-null    float64

 1   High       251 non-null    float64
```

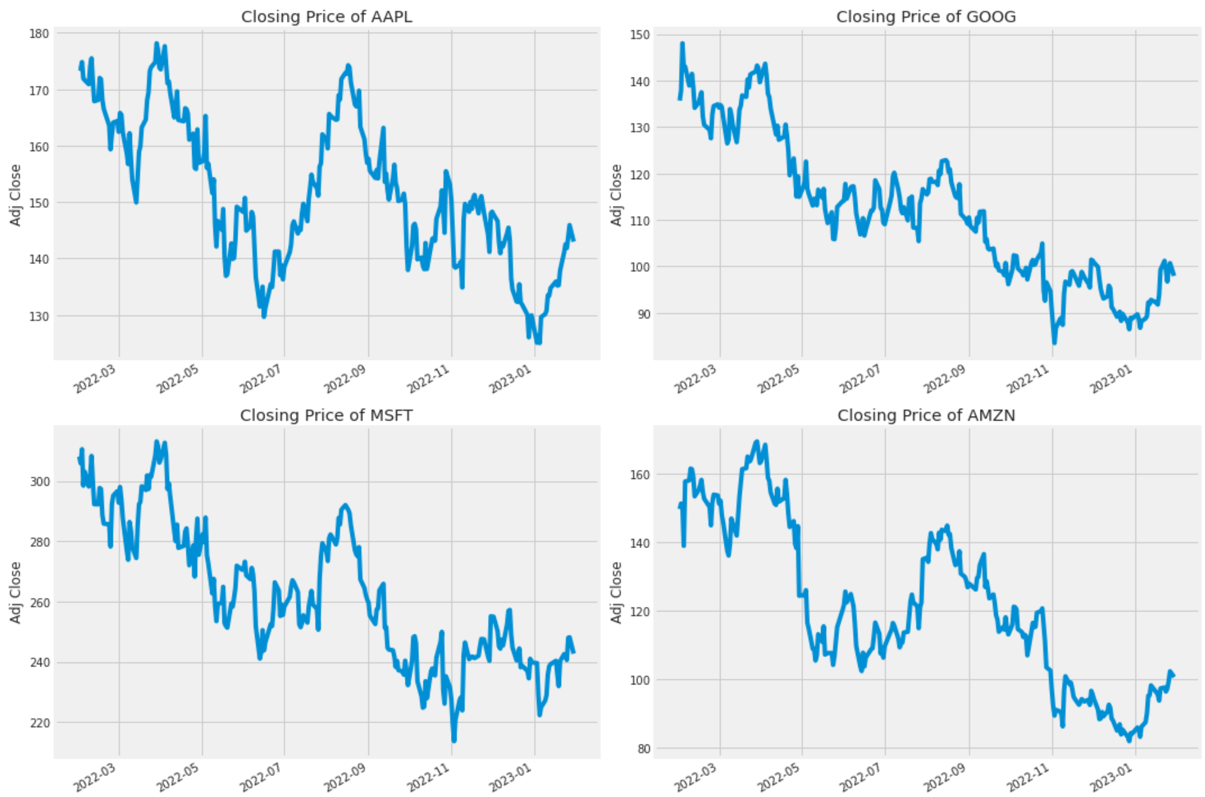
```
 2   Low          251 non-null    float64
 3   Close        251 non-null    float64
 4   Adj Close    251 non-null    float64
 5   Volume       251 non-null    int64
 6   company_name  251 non-null    object
dtypes: float64(5), int64(1), object(1)
memory usage: 23.8+ KB
```

Closing Price

The closing price is the last price at which the stock is traded during the regular trading day. A stock's closing price is the standard benchmark used by investors to track its performance over time.

In [5]:

```python
# 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()
```

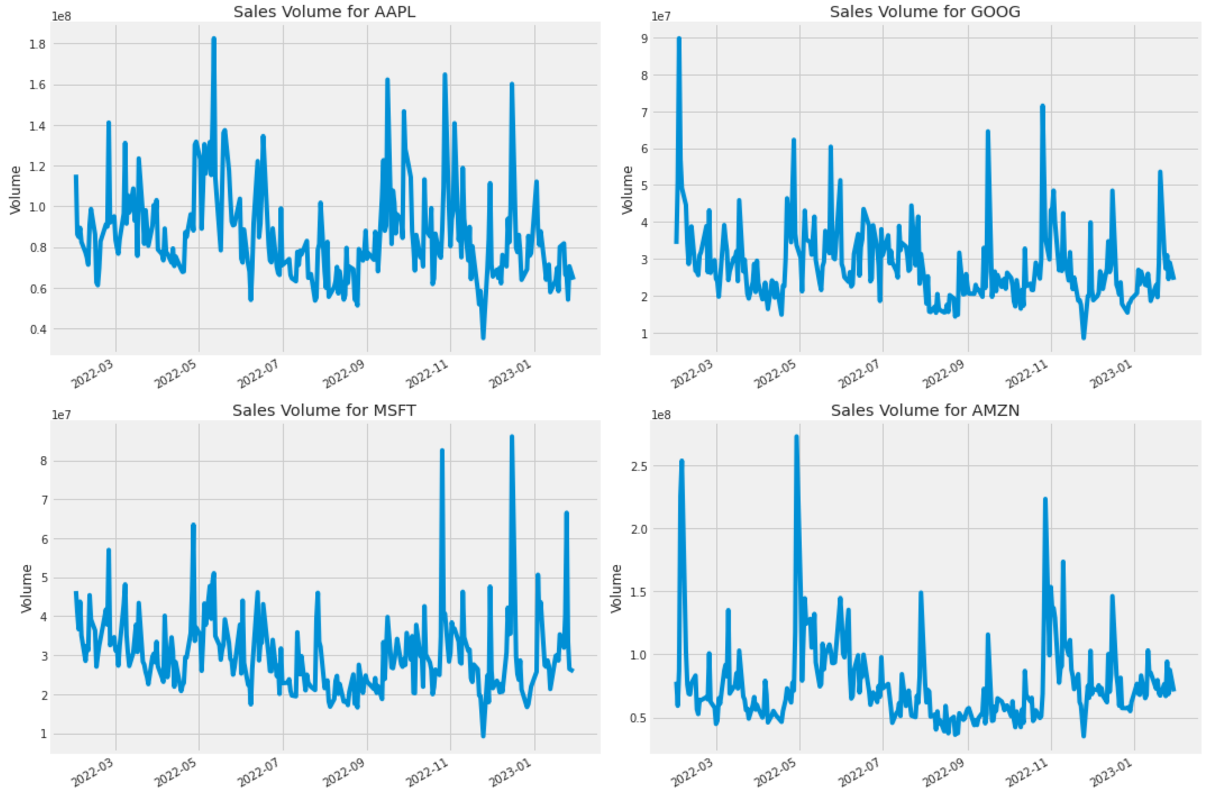


Volume of Sales

Volume is the amount of an asset or security that changes hands over some period of time, often over the course of a day. For instance, the stock trading volume would refer to the number of shares of security traded between its daily open and close. Trading volume, and changes to volume over the course of time, are important inputs for technical traders.

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()
```



Now that we've seen the visualizations for the closing price and the volume traded each day, let's go ahead and caculate the moving average for the stock.

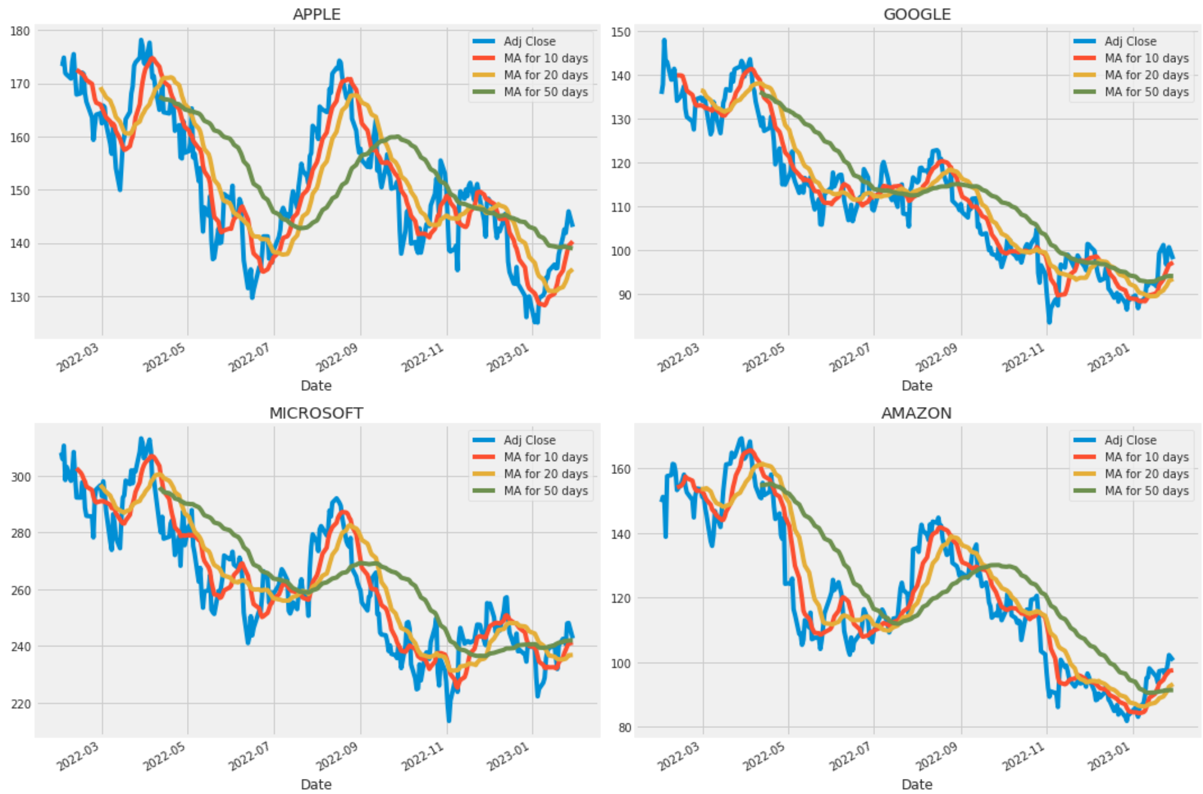
2. What was the moving average of the various stocks?

The moving average (MA) is a simple technical analysis tool that smooths out price data by creating a constantly updated average price. The average is taken over a specific period of time, like 10 days, 20 minutes, 30 weeks, or any time period the trader chooses.

In [7]:

```
ma_day = [10, 20, 50]


for ma in ma_day:
    for company in company_list:
        column_name = f"MA for {ma} days"
        company[column_name] = company['Adj Close'].rolling(ma).mean()



fig, axes = plt.subplots(nrows=2, ncols=2)
fig.set_figheight(10)
fig.set_figwidth(15)
```

```
AAPL[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[0,0])
axes[0,0].set_title('APPLE')

GOOG[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[0,1])
axes[0,1].set_title('GOOGLE')

MSFT[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[1,0])
axes[1,0].set_title('MICROSOFT')

AMZN[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[1,1])
axes[1,1].set_title('AMAZON')

fig.tight_layout()
```



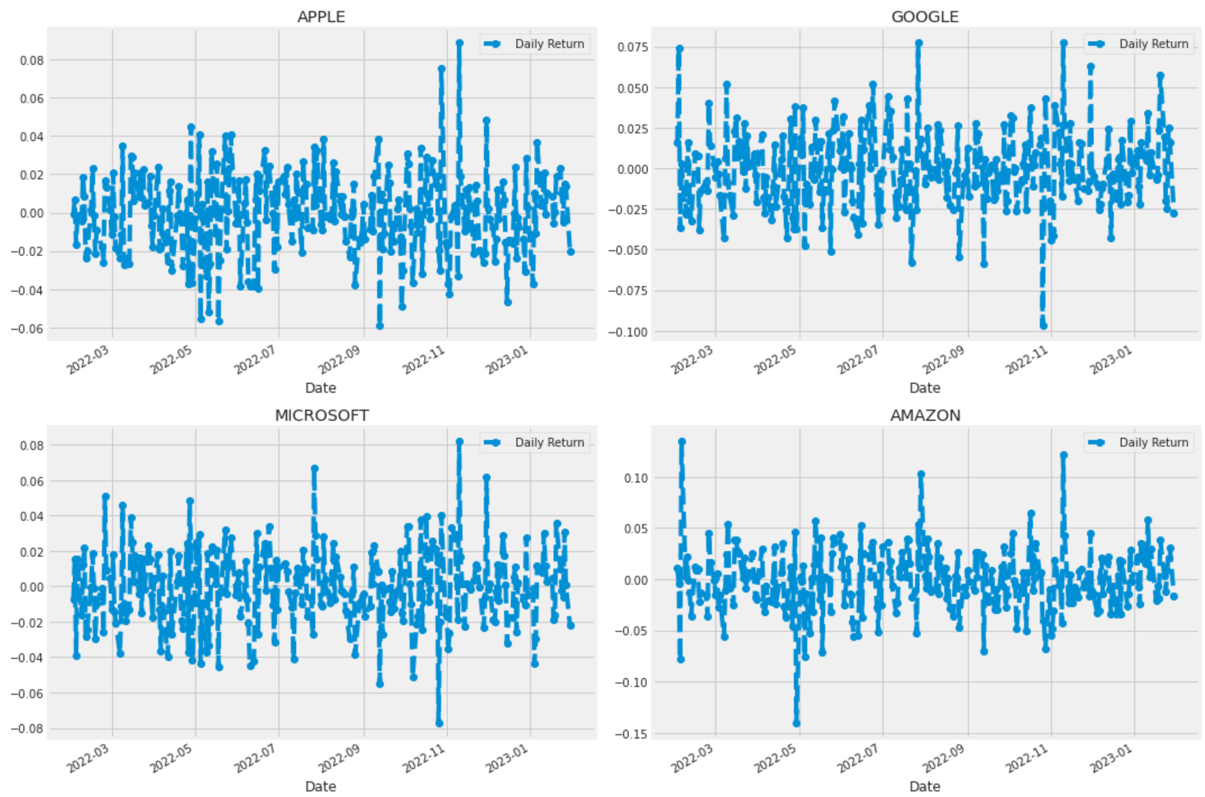
We see in the graph that the best values to measure the moving average are 10 and 20 days because we still capture trends in the data without noise.

3. What was the daily return of the stock on average?

Now that we've done some baseline analysis, let's go ahead and dive a little deeper. We're now going to analyze the risk of the stock. In order to do so we'll need to take a closer look at the daily changes of the stock, and not just its absolute value. Let's go ahead and use pandas to retrieve teh daily returns for the Apple stock.

In [8]:

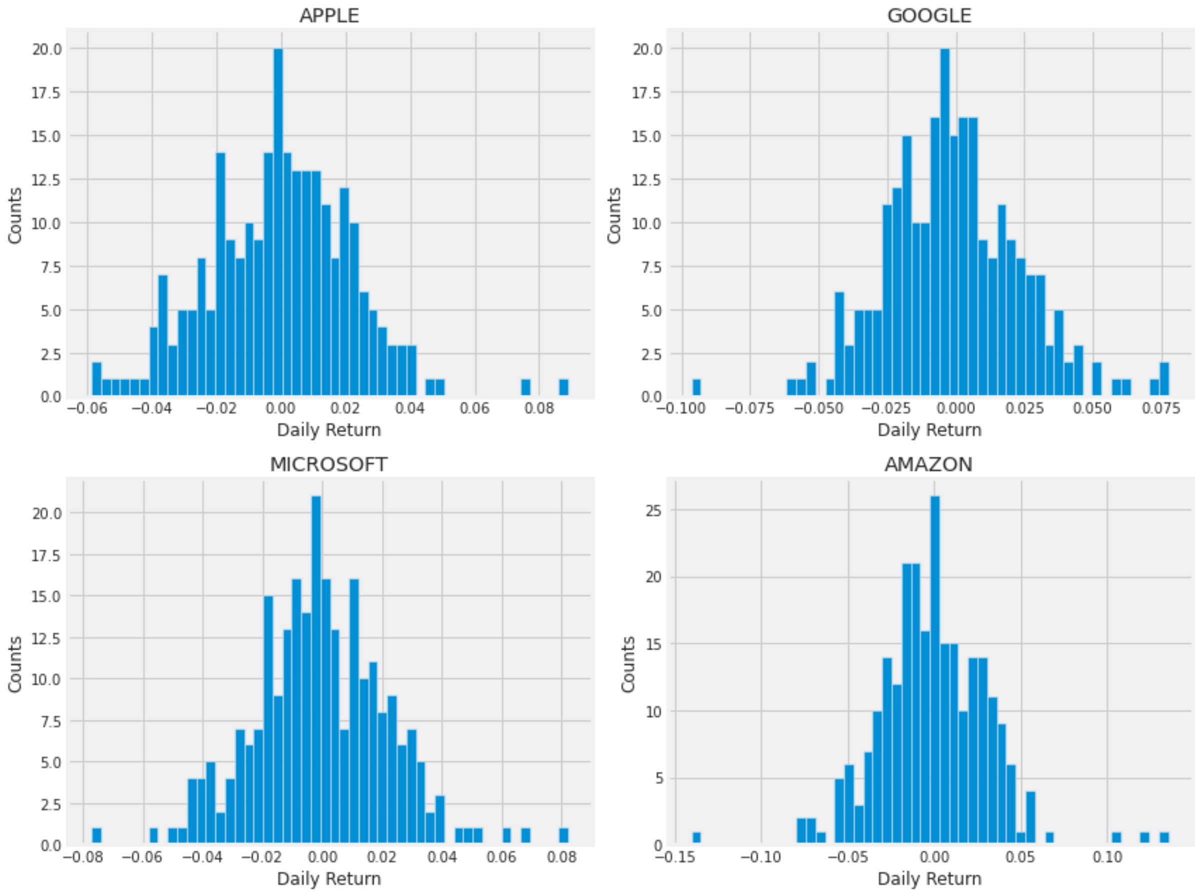
```
# We'll use pct_change to find the percent change for each day
for company in company_list:
    company['Daily Return'] = company['Adj Close'].pct_change()

# Then we'll plot the daily return percentage
fig, axes = plt.subplots(nrows=2, ncols=2)
fig.set_figheight(10)
fig.set_figwidth(15)

AAPL['Daily Return'].plot(ax=axes[0,0], legend=True, linestyle='--', marker='o')
axes[0,0].set_title('APPLE')

GOOG['Daily Return'].plot(ax=axes[0,1], legend=True, linestyle='--', marker='o')
axes[0,1].set_title('GOOGLE')

MSFT['Daily Return'].plot(ax=axes[1,0], legend=True, linestyle='--', marker='o')
axes[1,0].set_title('MICROSOFT')

AMZN['Daily Return'].plot(ax=axes[1,1], legend=True, linestyle='--', marker='o')
axes[1,1].set_title('AMAZON')

fig.tight_layout()
```



Great, now let's get an overall look at the average daily return using a histogram. We'll use seaborn to create both a histogram and kde plot on the same figure.

In [9]:

```
plt.figure(figsize=(12, 9))


for i, company in enumerate(company_list, 1):
    plt.subplot(2, 2, i)
    company['Daily Return'].hist(bins=50)
    plt.xlabel('Daily Return')
    plt.ylabel('Counts')
    plt.title(f'{company_name[i - 1]}')


plt.tight_layout()
```



4. What was the correlation between different stocks closing prices?

Correlation is a statistic that measures the degree to which two variables move in relation to each other which has a value that must fall between -1.0 and +1.0. Correlation measures association, but doesn't show if x causes y or vice versa — or if the association is caused by a third factor[1].

Now what if we wanted to analyze the returns of all the stocks in our list? Let's go ahead and build a DataFrame with all the ['Close'] columns for each of the stocks dataframes.

In [10]:

```
# Grab all the closing prices for the tech stock list into one DataFrame


closing_df = pdr.get_data_yahoo(tech_list, start=start, end=end)['Adj Close']


# Make a new tech returns DataFrame

tech_rets = closing_df.pct_change()

tech_rets.head()
```

[*********************100%***********************] 4 of 4 completed

Out[10]:

	AAPL	AMZN	GOOG	MSFT
Date				
2022-01-31 00:00:00-05:00	NaN	NaN	NaN	NaN
2022-02-01 00:00:00-05:00	-0.000973	0.010831	0.016065	-0.007139
2022-02-02 00:00:00-05:00	0.007044	-0.003843	0.073674	0.015222
2022-02-03 00:00:00-05:00	-0.016720	-0.078128	-0.036383	-0.038952
2022-02-04 00:00:00-05:00	-0.001679	0.135359	0.002562	0.015568

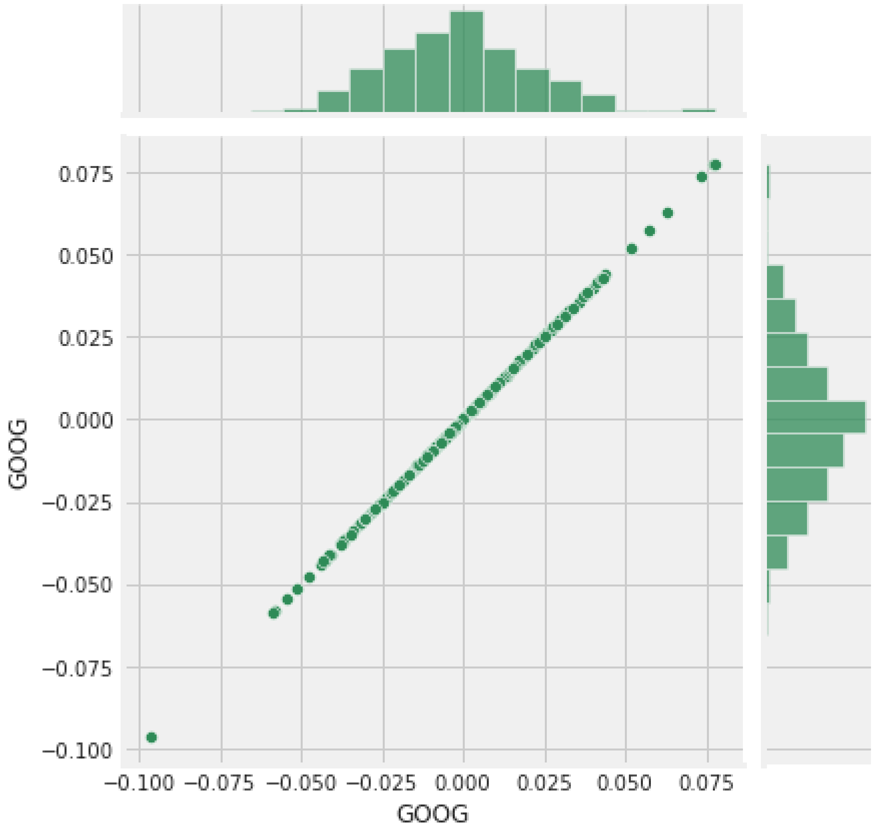
Now we can compare the daily percentage return of two stocks to check how correlated. First let's see a sotck compared to itself.

In [11]:

```
# Comparing Google to itself should show a perfectly linear relationship
sns.jointplot(x='GOOG', y='GOOG', data=tech_rets, kind='scatter', color='seagreen')
```

Out[11]:

<seaborn.axisgrid.JointGrid at 0x7f63e33d4990>



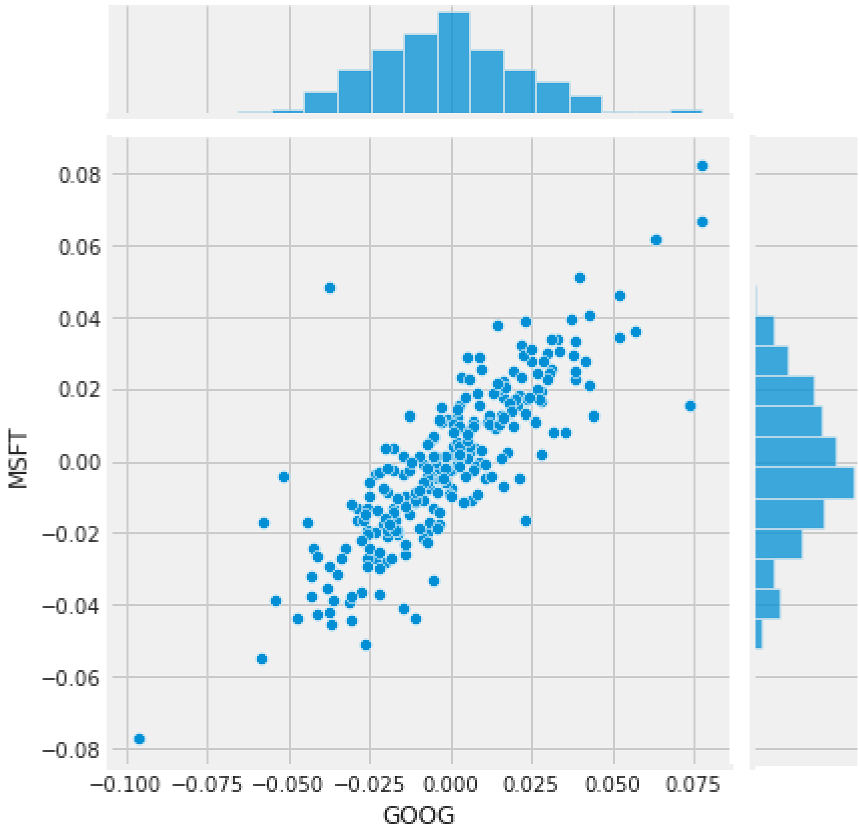
In [12]:

We'll use joinplot to compare the daily returns of Google and Microsoft

sns.jointplot(x='GOOG', y='MSFT', data=tech_rets, kind='scatter')

Out[12]:

<seaborn.axisgrid.JointGrid at 0x7f63dba49210>



So now we can see that if two stocks are perfectly (and positivley) correlated with each other a linear relationship bewteen its daily return values should occur.

Seaborn and pandas make it very easy to repeat this comparison analysis for every possible combination of stocks in our technology stock ticker list. We can use sns.pairplot() to automatically create this plot

In [13]:

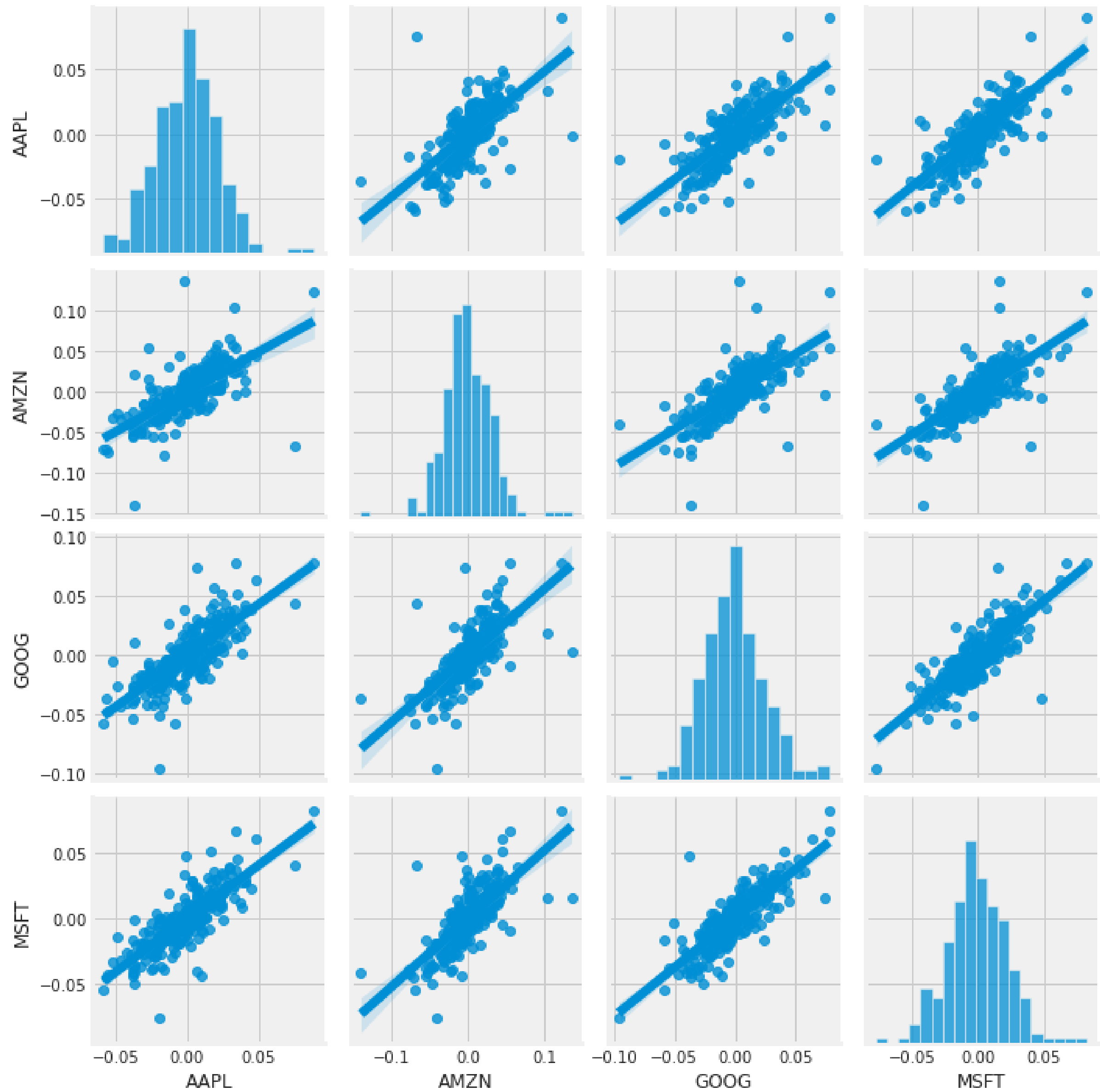
 # We can simply call pairplot on our DataFrame for an automatic visual analysis

# of all the comparisons

sns.pairplot(tech_rets, kind='reg')

Out[13]:

<seaborn.axisgrid.PairGrid at 0x7f63c3f952d0>



Above we can see all the relationships on daily returns between all the stocks. A quick glance shows an interesting correlation between Google and Amazon daily returns. It might be interesting to investigate that individual comaprison.

While the simplicity of just calling sns.pairplot() is fantastic we can also use sns.PairGrid() for full control of the figure, including what kind of plots go in the diagonal, the upper triangle, and the lower triangle. Below is an example of utilizing the full power of seaborn to achieve this result.

In [14]:

 # Set up our figure by naming it returns_fig, call PairPLot on the DataFrame

return_fig = sns.PairGrid(tech_rets.dropna())

 # Using map_upper we can specify what the upper triangle will look like.

return_fig.map_upper(plt.scatter, color='purple')

 # We can also define the lower triangle in the figure, inclufing the plot type (kde)

 # or the color map (BluePurple)

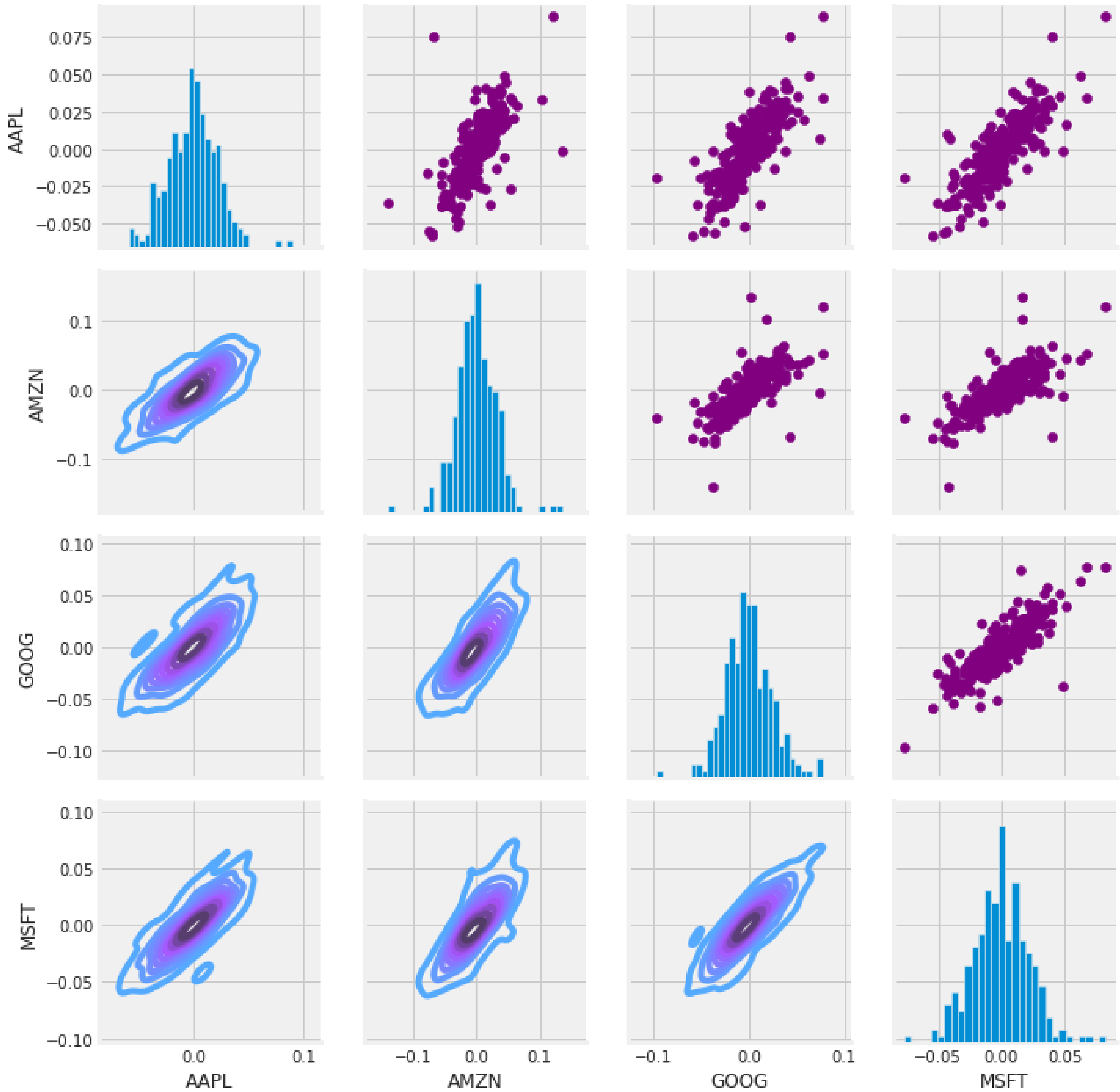
return_fig.map_lower(sns.kdeplot, cmap='cool_d')

 # Finally we'll define the diagonal as a series of histogram plots of the daily return

return_fig.map_diag(plt.hist, bins=30)

Out[14]:

<seaborn.axisgrid.PairGrid at 0x7f63dbee8c10>



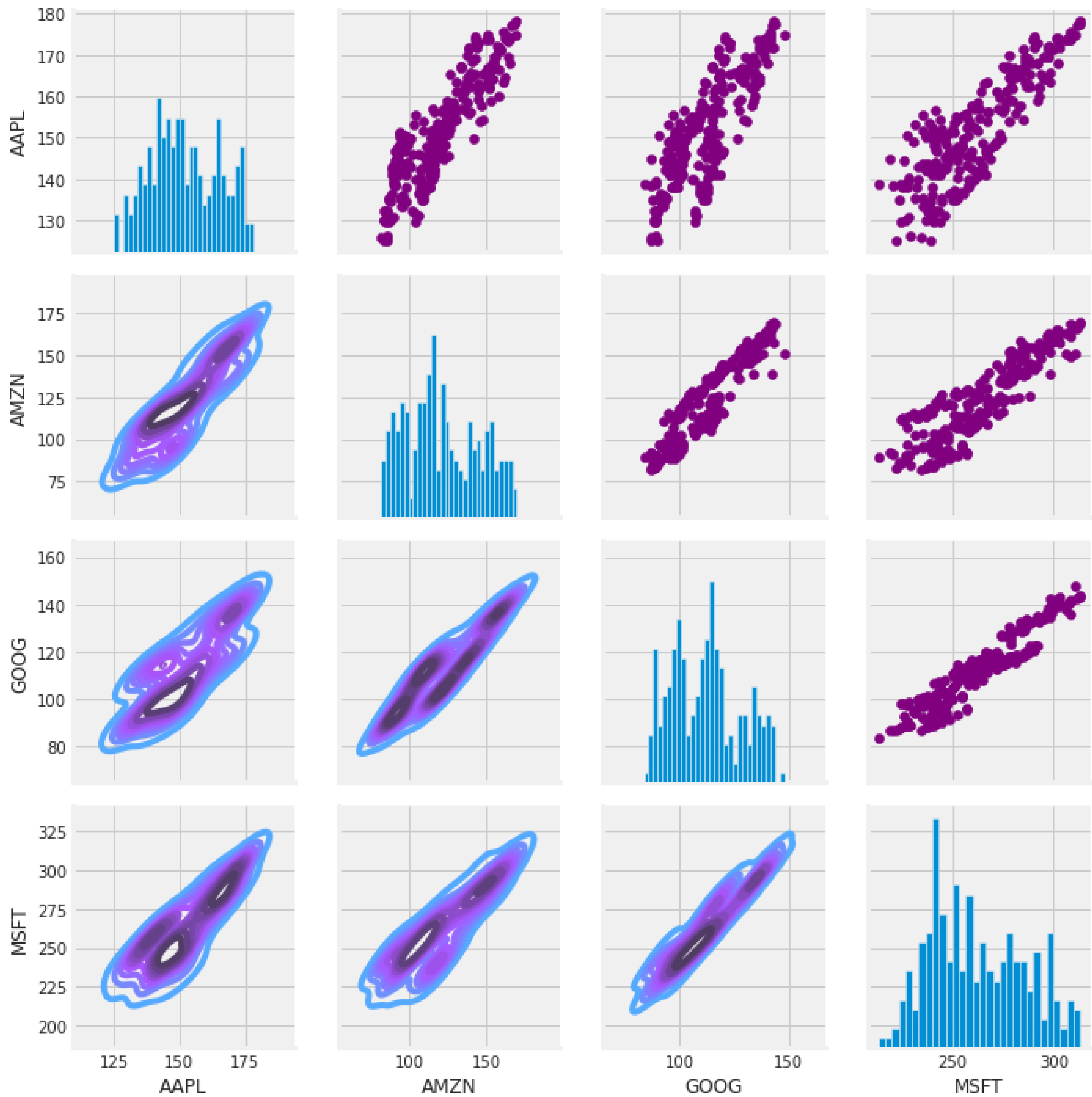
In [15]:

 # Set up our figure by naming it returns_fig, call PairPLot on the DataFrame

```python
returns_fig = sns.PairGrid(closing_df)

# Using map_upper we can specify what the upper triangle will look like.
returns_fig.map_upper(plt.scatter,color='purple')

# We can also define the lower triangle in the figure, inclufing the plot type (kde) or the color map (BluePurple)
returns_fig.map_lower(sns.kdeplot,cmap='cool_d')

# Finally we'll define the diagonal as a series of histogram plots of the daily return
returns_fig.map_diag(plt.hist,bins=30)
```

Out[15]:

```
<seaborn.axisgrid.PairGrid at 0x7f63bb2df7d0>
```



Finally, we could also do a correlation plot, to get actual numerical values for the correlation between the stocks' daily return values. By comparing the closing prices, we see an interesting relationship between Microsoft and Apple.

In [16]:

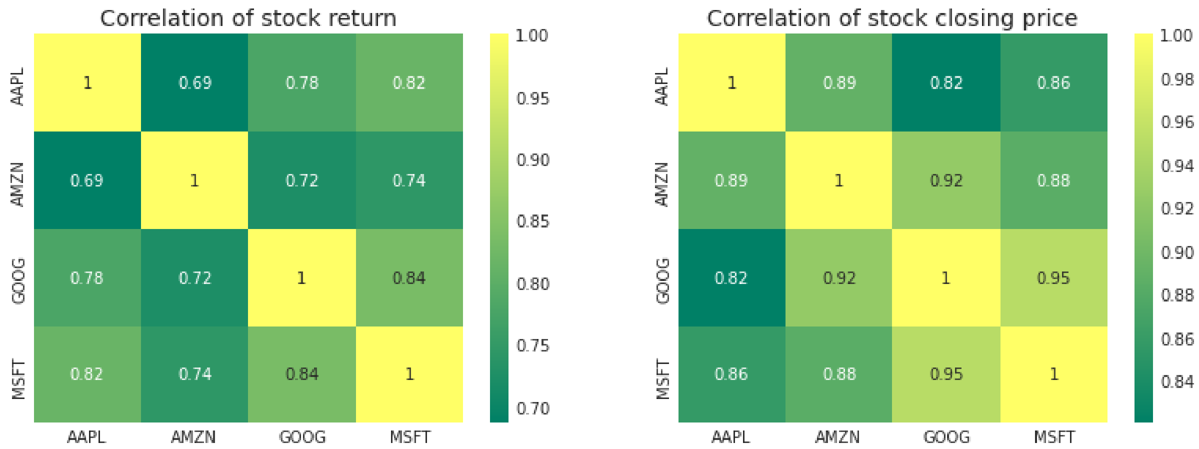
```
plt.figure(figsize=(12, 10))


plt.subplot(2, 2, 1)

sns.heatmap(tech_rets.corr(), annot=True, cmap='summer')

plt.title('Correlation of stock return')


plt.subplot(2, 2, 2)
```

sns.heatmap(closing_df.corr(), annot=True, cmap='summer')

plt.title('Correlation of stock closing price')

Out[16]:

Text(0.5, 1.0, 'Correlation of stock closing price')



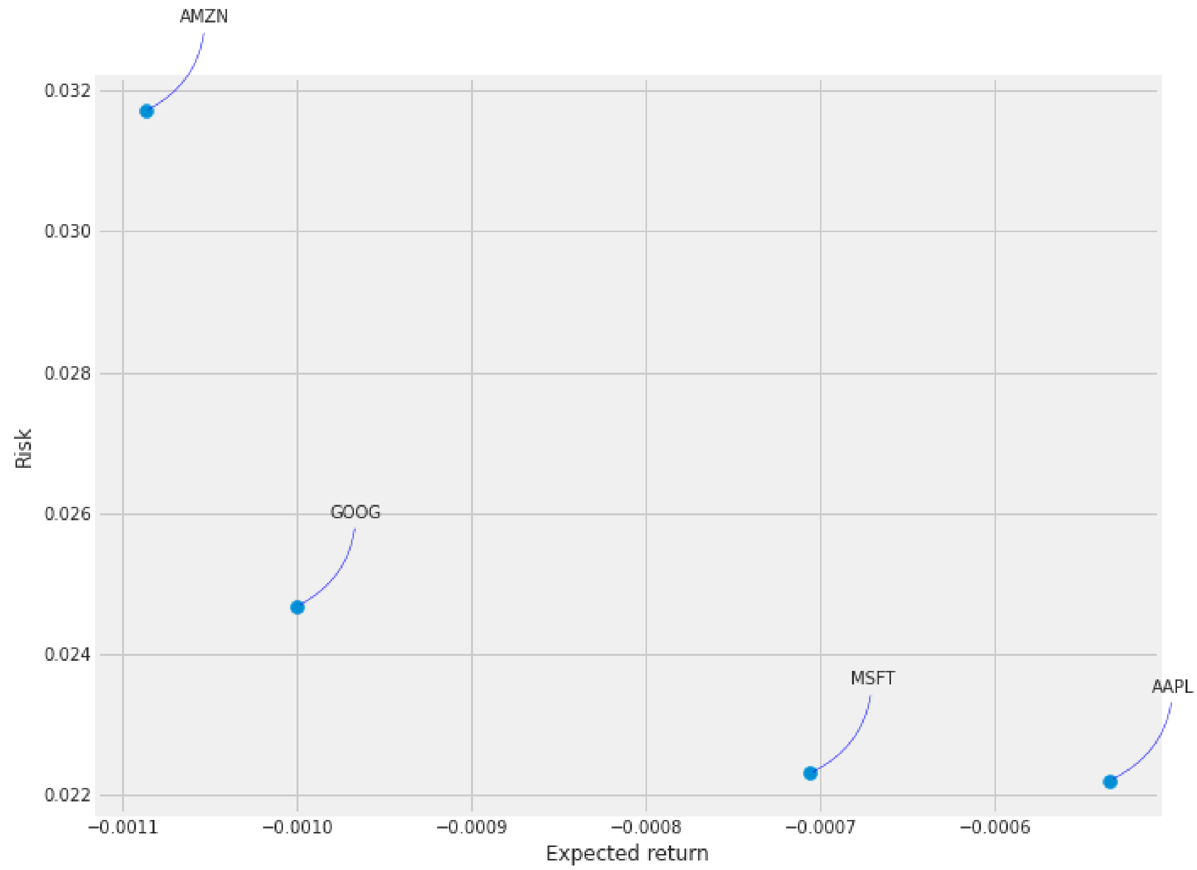
Just like we suspected in our PairPlot we see here numerically and visually that Microsoft and Amazon had the strongest correlation of daily stock return. It's also interesting to see that all the technology comapnies are positively correlated.

5. How much value do we put at risk by investing in a particular stock?

There are many ways we can quantify risk, one of the most basic ways using the information we've gathered on daily percentage returns is by comparing the expected return with the standard deviation of the daily returns.

In [17]:

```
rets = tech_rets.dropna()


area = np.pi * 20


plt.figure(figsize=(10, 8))

plt.scatter(rets.mean(), rets.std(), s=area)

plt.xlabel('Expected return')

plt.ylabel('Risk')


for label, x, y in zip(rets.columns, rets.mean(), rets.std()):

    plt.annotate(label, xy=(x, y), xytext=(50, 50), textcoords='offset points', ha='right', va='bottom',

            arrowprops=dict(arrowstyle='-', color='blue', connectionstyle='arc3,rad=-0.3'))
```



6. Predicting the closing price stock price of APPLE inc:

In [18]:

 # 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[18]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2012-01-03 00:00:00-05:00	14.621429	14.732143	14.607143	14.686786	12.519278	302220800
2012-01-04	14.642857	14.810000	14.617143	14.765714	12.586559	260022000

	Open	High	Low	Close	Adj Close	Volume
Date						
00:00:00-05:00						
2012-01-05 00:00:00-05:00	14.819643	14.948214	14.738214	14.929643	12.726295	271269600
2012-01-06 00:00:00-05:00	14.991786	15.098214	14.972143	15.085714	12.859331	318292800
2012-01-09 00:00:00-05:00	15.196429	15.276786	15.048214	15.061786	12.838936	394024400
...	...	...	...	...	...	...
2023-01-24 00:00:00-05:00	140.309998	143.160004	140.300003	142.529999	142.529999	66435100
2023-01-25 00:00:00-05:00	140.889999	142.429993	138.809998	141.860001	141.860001	65799300
2023-01-26 00:00:00-05:00	143.169998	144.250000	141.899994	143.960007	143.960007	54105100
2023-01-27 00:00:00-05:00	143.160004	147.229996	143.080002	145.929993	145.929993	70492800
2023-01-30	144.960007	145.550003	142.850006	143.000000	143.000000	63947600

	Open	High	Low	Close	Adj Close	Volume
Date						
00:00:00-05:00						

2787 rows × 6 columns

In [19]:

```
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 [20]:

```
# 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[20]:

2648

In [21]:

```
# Scale the data
from sklearn.preprocessing import MinMaxScaler

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

scaled_data
```

Out[21]:

```
array([[0.00439887],
       [0.00486851],
       [0.00584391],
       ...,
       [0.7735962 ],
       [0.78531794],
       [0.767884  ]])
```

In [22]:

```
# 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
```

unfold_moreShow hidden output

In [23]:

```
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)
```

```
2023-01-31 12:54:26.137995: I tensorflow/core/common_runtime/process_util.cc:146] Creating new thread pool with default inter op setting: 2. Tune using inter_op_parallelism_threads for best performance.

2023-01-31 12:54:26.831521: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR Optimization Passes are enabled (registered 2)

2588/2588 [==============================] - 98s 37ms/step - loss: 0.0013
```

Out[23]:

```
<keras.callbacks.History at 0x7f639802c810>
```

In [24]:

```
# 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
```

Out[24]:

4.982936594544208

In [25]:

```
# 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()

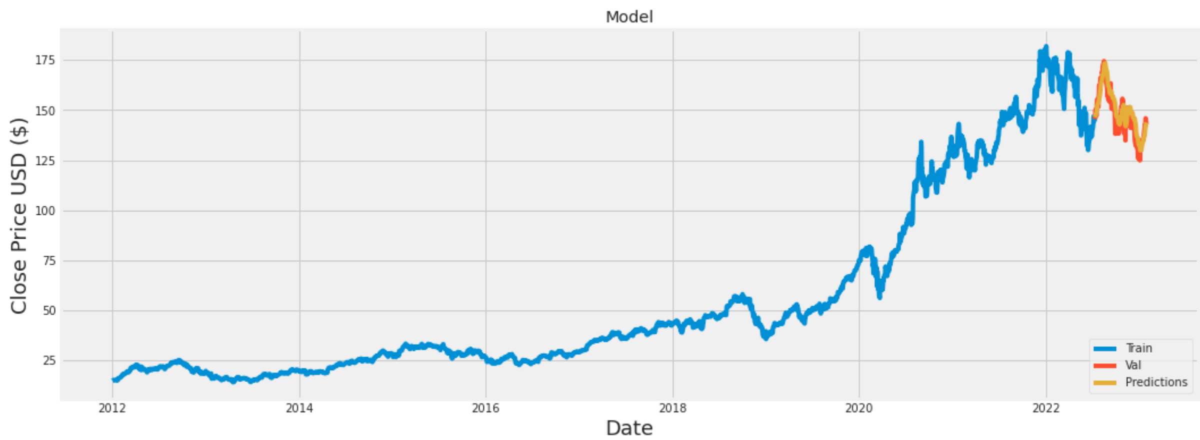
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:4: SettingWithCopyWarning:

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/user_guide/indexing.html#returning-a-view-versus-a-copy

after removing the cwd from sys.path.



In [26]:

 # Show the valid and predicted prices

valid

Out[26]:

	Close	Predictions
Date		
2022-07-13 00:00:00-04:00	145.490005	146.457565

	Close	Predictions
Date		
2022-07-14 00:00:00-04:00	148.470001	146.872879
2022-07-15 00:00:00-04:00	150.169998	147.586197
2022-07-18 00:00:00-04:00	147.070007	148.572937
2022-07-19 00:00:00-04:00	151.000000	148.995255
...	...	...
2023-01-24 00:00:00-05:00	142.529999	138.565536
2023-01-25 00:00:00-05:00	141.860001	140.022110
2023-01-26 00:00:00-05:00	143.960007	141.225128
2023-01-27 00:00:00-05:00	145.929993	142.469315
2023-01-30 00:00:00-05:00	143.000000	143.833130

139 rows × 2 columns

linkcode

Summary

In this notebook, you discovered and explored stock data.

Specifically, you learned:

- How to load stock market data from the YAHOO Finance website using yfinance.
- How to explore and visualize time-series data using Pandas, Matplotlib, and Seaborn.
- How to measure the correlation between stocks.
- How to measure the risk of investing in a particular stock.