# A Whale off the Port(folio)

---

In this assignment, you'll get to use what you've learned this week to evaluate the performance among various algorithmic, hedge, and mutual fund portfolios and compare them against the S&P 500 Index.

In [2]:
```
# Initial imports
import pandas as pd
import numpy as np
import datetime as dt
from pathlib import Path

%matplotlib inline
```

# Data Cleaning

In this section, you will need to read the CSV files into DataFrames and perform any necessary data cleaning steps. After cleaning, combine all DataFrames into a single DataFrame.

Files:

- `whale_returns.csv` : Contains returns of some famous "whale" investors' portfolios.
- `algo_returns.csv` : Contains returns from the in-house trading algorithms from Harold's company.
- `sp500_history.csv` : Contains historical closing prices of the S&P 500 Index.

In [3]:
```
#file_path = Path="C:/Users/CS_Knit_tinK_SC/Documents/My Data Sources/loans.c
#csv_path = Path="C:/Users/CS_Knit_tinK_SC/Documents/My Data Sources/goog_goo
csv_path_w = "C:/Users/CS_Knit_tinK_SC/Documents/My Data Sources/Whale/whale_
whale_returns = pd.read_csv(
    csv_path_w, parse_dates=True, index_col="Date", infer_datetime_format=True
)
whale_returns.head()
```

Out[3]:

| Date | SOROS FUND MANAGEMENT LLC | PAULSON & CO.INC. | TIGER GLOBAL MANAGEMENT LLC | BERKSHIRE HATHAWAY INC |
|---|---|---|---|---|
| **2015-03-02** | NaN | NaN | NaN | NaN |
| **2015-03-03** | -0.001266 | -0.004981 | -0.000496 | -0.006569 |
| **2015-03-04** | 0.002230 | 0.003241 | -0.002534 | 0.004213 |
| **2015-03-05** | 0.004016 | 0.004076 | 0.002355 | 0.006726 |

| | SOROS FUND MANAGEMENT LLC | PAULSON & CO.INC. | TIGER GLOBAL MANAGEMENT LLC | BERKSHIRE HATHAWAY INC |
|---|---|---|---|---|

# Whale Returns

Read the Whale Portfolio daily returns and clean the data

In [4]:
```
# Reading whale returns
whale_returns.head()
```

Out[4]:

| Date | SOROS FUND MANAGEMENT LLC | PAULSON & CO.INC. | TIGER GLOBAL MANAGEMENT LLC | BERKSHIRE HATHAWAY INC |
|---|---|---|---|---|
| **2015-03-02** | NaN | NaN | NaN | NaN |
| **2015-03-03** | -0.001266 | -0.004981 | -0.000496 | -0.006569 |
| **2015-03-04** | 0.002230 | 0.003241 | -0.002534 | 0.004213 |
| **2015-03-05** | 0.004016 | 0.004076 | 0.002355 | 0.006726 |
| **2015-03-06** | -0.007905 | -0.003574 | -0.008481 | -0.013098 |

In [5]:
```
# Count nulls
whale_returns.isnull().mean() * 100
```

Out[5]:
```
SOROS FUND MANAGEMENT LLC      0.09434
PAULSON & CO.INC.              0.09434
TIGER GLOBAL MANAGEMENT LLC    0.09434
BERKSHIRE HATHAWAY INC         0.09434
dtype: float64
```

In [6]:
```
# Drop nulls
whale_returns=whale_returns.dropna()
```

In [7]:
```
#file_path = Path="C:/Users/CS_Knit_tinK_SC/Documents/My Data Sources/loans.c
#csv_path = Path="C:/Users/CS_Knit_tinK_SC/Documents/My Data Sources/goog_goo
csv_path_a = "C:/Users/CS_Knit_tinK_SC/Documents/My Data Sources/Whale/algo_re
algo_returns = pd.read_csv(
    csv_path_a, parse_dates=True, index_col="Date", infer_datetime_format=True
)
algo_returns.head()
```

Out[7]:

| Date | Algo 1 | Algo 2 |
|---|---|---|
| **2014-05-28** | 0.001745 | NaN |
| **2014-05-29** | 0.003978 | NaN |
| **2014-05-30** | 0.004464 | NaN |

| Date | Algo 1 | Algo 2 |
|---|---|---|

# Algorithmic Daily Returns

Read the algorithmic daily returns and clean the data

In [8]:
```
# Reading algorithmic returns
algo_returns.head()
```

Out[8]:

| Date | Algo 1 | Algo 2 |
|---|---|---|
| **2014-05-28** | 0.001745 | NaN |
| **2014-05-29** | 0.003978 | NaN |
| **2014-05-30** | 0.004464 | NaN |
| **2014-06-02** | 0.005692 | NaN |
| **2014-06-03** | 0.005292 | NaN |

In [9]:
```
# Count nulls
algo_returns.isnull().mean() * 100
```

Out[9]:
```
Algo 1    0.000000
Algo 2    0.483481
dtype: float64
```

In [10]:
```
# Drop nulls
algo_returns=algo_returns.dropna()
```

In [11]:
```
#file_path = Path="C:/Users/CS_Knit_tinK_SC/Documents/My Data Sources/loans.c
#csv_path = Path="C:/Users/CS_Knit_tinK_SC/Documents/My Data Sources/goog_goo
csv_path_s = "C:/Users/CS_Knit_tinK_SC/Documents/My Data Sources/Whale/sp500_
sp500_history = pd.read_csv(csv_path_s)
sp500_history.head()
```

Out[11]:

| | Date | Close |
|---|---|---|
| **0** | 23-Apr-19 | $2933.68 |
| **1** | 22-Apr-19 | $2907.97 |
| **2** | 18-Apr-19 | $2905.03 |
| **3** | 17-Apr-19 | $2900.45 |
| **4** | 16-Apr-19 | $2907.06 |

# S&P 500 Returns

Read the S&P 500 historic closing prices and create a new daily returns DataFrame from the data.

In [12]:
```
# Reading S&P 500 Closing Prices
sp500_history.head()
```

Out[12]:

| | Date | Close |
|---|---|---|
| **0** | 23-Apr-19 | $2933.68 |
| **1** | 22-Apr-19 | $2907.97 |
| **2** | 18-Apr-19 | $2905.03 |
| **3** | 17-Apr-19 | $2900.45 |
| **4** | 16-Apr-19 | $2907.06 |

In [13]:
```
# look at shape of data
sp500_history.shape
```

Out[13]:
```
(1649, 2)
```

In [14]:
```
# Check Data Types
sp500_history.dtypes
```

Out[14]:
```
Date     object
Close    object
dtype: object
```

In [15]:
```
# Fix Data Types
sp500_history.set_index(pd.to_datetime(sp500_history['Date'], infer_datetime_
sp500_history['Close'] = sp500_history['Close'].str.replace('$','')
sp500_history['Close']=sp500_history['Close'].astype('float')

# Drop the extra date column
sp500_history.drop(columns=['Date'], inplace=True)
sp500_history.head()
```

```
C:\Users\CS_Knit_tinK_SC\anaconda3\envs\dev\lib\site-packages\ipykernel_launch
er.py:3: FutureWarning: The default value of regex will change from True to Fa
lse in a future version. In addition, single character regular expressions wil
l *not* be treated as literal strings when regex=True.
  This is separate from the ipykernel package so we can avoid doing imports un
til
```

Out[15]:

| | Close |
|---|---|
| **Date** | |
| **2019-04-23** | 2933.68 |
| **2019-04-22** | 2907.97 |

| | Close |
|---|---|
| Date | |
| 2019-04-18 | 2905.03 |

In [16]:
```
# Calculate Daily Returns


sp500_returns = sp500_history.pct_change()
```

In [17]:
```
# Drop nulls
sp500_history=sp500_history.dropna()
```

In [18]:
```
# Rename `Close` Column to be specific to this portfolio.
sp500_returns.columns = ["SP500"]
```

## Combine Whale, Algorithmic, and S&P 500 Returns

In [19]:
```
# Join Whale Returns, Algorithmic Returns, and the S&P 500 Returns into a sin
all_returns = pd.concat([algo_returns, whale_returns, sp500_returns], axis='c
```

---

# Conduct Quantitative Analysis

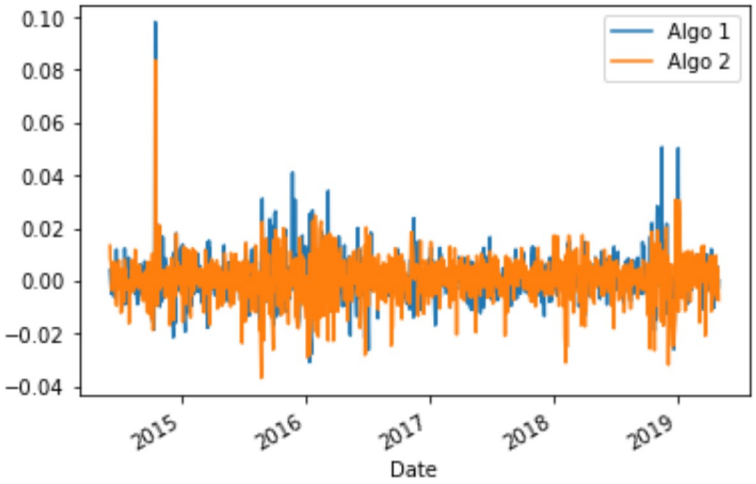
In this section, you will calculate and visualize performance and risk metrics for the portfolios.

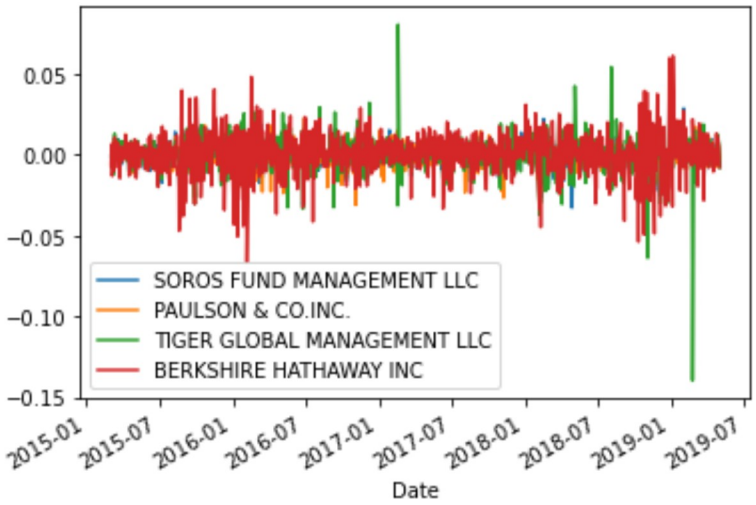
## Performance Anlysis

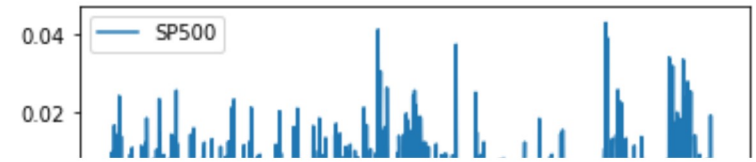
### Calculate and Plot the daily returns.

In [20]:
```
# Plot daily returns of all portfolios
algo_returns.plot()
whale_returns.plot()
sp500_returns.plot()
```

Out[20]:
```
<AxesSubplot:xlabel='Date'>
```



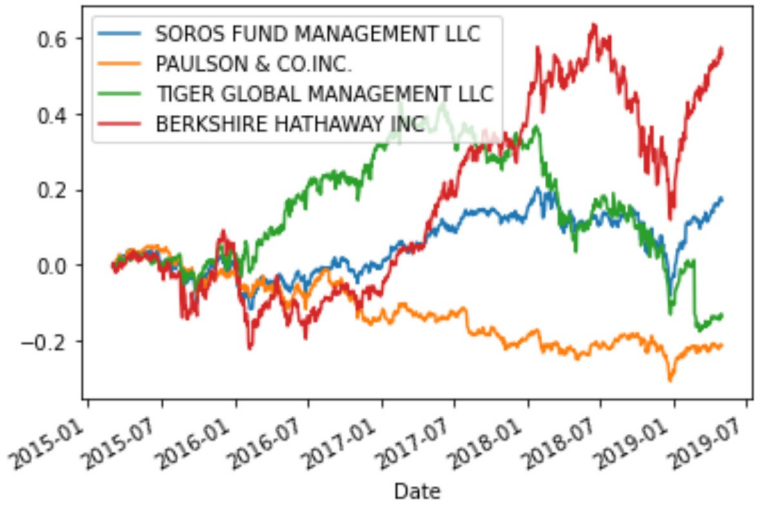

## Calculate and Plot cumulative returns.

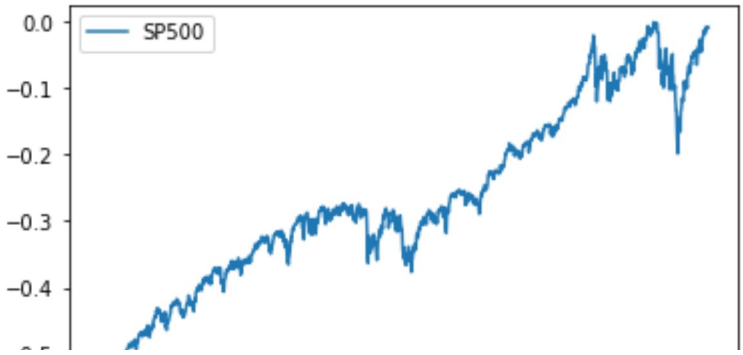
In [21]:
```
# Calculate cumulative returns of all portfolios
cumulative_algo_returns = (1 + algo_returns).cumprod() - 1
cumulative_whale_returns = (1 + whale_returns).cumprod() - 1
cumulative_sp500_returns = (1 + sp500_returns).cumprod() - 1


# Plot cumulative returns

cumulative_algo_returns.plot()
cumulative_whale_returns.plot()
cumulative_sp500_returns.plot()
```

Out[21]: <AxesSubplot:xlabel='Date'>


---

# Risk Analysis

Determine the *risk* of each portfolio:

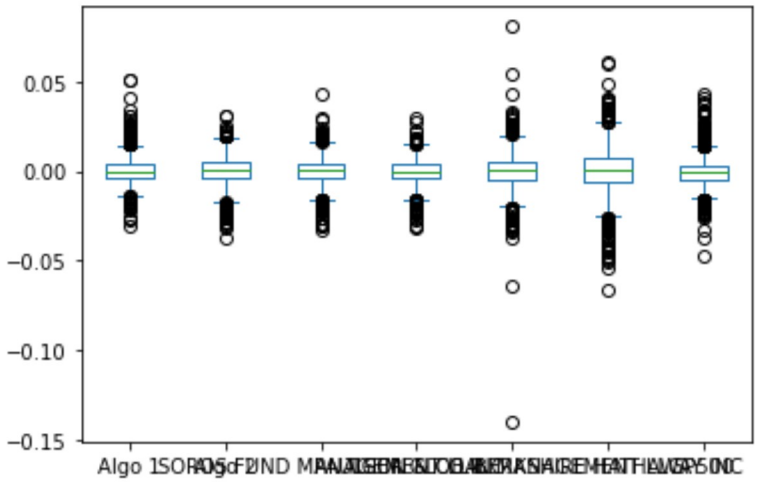
1. Create a box plot for each portfolio.
2. Calculate the standard deviation for all portfolios
3. Determine which portfolios are riskier than the S&P 500
4. Calculate the Annualized Standard Deviation

## Create a box plot for each portfolio

In [22]:
```
# Box plot to visually show risk

all_returns.plot.box()
```

Out[22]:
```
<AxesSubplot:>
```




## Calculate Standard Deviations

In [23]:
```
# Calculate the daily standard deviations of all portfolios

daily_std = all_returns.std()
print(f' Daily Standard Returns are: \n{daily_std}')
```

```
 Daily Standard Returns are:
Algo 1                         0.007620
Algo 2                         0.008342
SOROS FUND MANAGEMENT LLC      0.007895
PAULSON & CO.INC.              0.007023
TIGER GLOBAL MANAGEMENT LLC    0.010894
BERKSHIRE HATHAWAY INC         0.012919
SP500                          0.008587
dtype: float64
```

## Determine which portfolios are riskier than the S&P 500

In [ ]:
```
# Calculate  the daily standard deviation of S&P 500


# Determine which portfolios are riskier than the S&P 500
```

## Calculate the Annualized Standard Deviation

In [ ]:
```
# Calculate the annualized standard deviation (252 trading days)
```

---

# Rolling Statistics

Risk changes over time. Analyze the rolling statistics for Risk and Beta.

1. Calculate and plot the rolling standard deviation for all portfolios using a 21-day window
2. Calculate the correlation between each stock to determine which portfolios may mimick the S&P 500
3. Choose one portfolio, then calculate and plot the 60-day rolling beta between it and the S&P 500

## Calculate and plot rolling `std` for all portfolios with 21-day window

In [ ]:
```
# Calculate the rolling standard deviation for all portfolios using a 21-day w

# Plot the rolling standard deviation
```

## Calculate and plot the correlation

In [ ]:
```
# Calculate the correlation

# Display de correlation matrix
```

## Calculate and Plot Beta for a chosen portfolio and the S&P 500

In [ ]:
```
# Calculate covariance of a single portfolio

# Calculate variance of S&P 500

# Computing beta

# Plot beta trend
```

# Rolling Statistics Challenge: Exponentially Weighted Average

An alternative way to calculate a rolling window is to take the exponentially weighted moving average. This is like a moving window average, but it assigns greater importance to more recent observations. Try calculating the `ewm` with a 21-day half-life.

In [ ]:
```
# Use `ewm` to calculate the rolling window
```

---

# Sharpe Ratios

In reality, investment managers and thier institutional investors look at the ratio of return-to-risk, and not just returns alone. After all, if you could invest in one of two portfolios, and each offered the same 10% return, yet one offered lower risk, you'd take that one, right?

## Using the daily returns, calculate and visualize the Sharpe ratios using a bar plot

In [ ]:
```
# Annualized Sharpe Ratios
```

In [ ]:
```
# Visualize the sharpe ratios as a bar plot
```

## Determine whether the algorithmic strategies outperform both the market (S&P 500) and the whales portfolios.

Write your answer here!

---

# Create Custom Portfolio

In this section, you will build your own portfolio of stocks, calculate the returns, and compare the results to the Whale Portfolios and the S&P 500.

1. Choose 3-5 custom stocks with at last 1 year's worth of historic prices and create a DataFrame of the closing prices and dates for each stock.
2. Calculate the weighted returns for the portfolio assuming an equal number of shares for each stock
3. Join your portfolio returns to the DataFrame that contains all of the portfolio returns
4. Re-run the performance and risk analysis with your portfolio to see how it compares to the others
5. Include correlation analysis to determine which stocks (if any) are correlated

## Choose 3-5 custom stocks with at last 1 year's worth of historic prices and create a DataFrame of the closing prices and dates for each stock.

For this demo solution, we fetch data from three companies listes in the S&P 500 index.

- `GOOG` - Google, LLC
- `AAPL` - Apple Inc.

- COST - Costco Wholesale Corporation

In [ ]:
```
# Reading data from 1st stock
```

In [ ]:
```
# Reading data from 2nd stock
```

In [ ]:
```
# Reading data from 3rd stock
```

In [ ]:
```
# Combine all stocks in a single DataFrame
```

In [ ]:
```
# Reset Date index
```

In [ ]:
```
# Reorganize portfolio data by having a column per symbol
```

In [ ]:
```
# Calculate daily returns

# Drop NAs

# Display sample data
```

## Calculate the weighted returns for the portfolio assuming an equal number of shares for each stock

In [ ]:
```
# Set weights
weights = [1/3, 1/3, 1/3]

# Calculate portfolio return

# Display sample data
```

## Join your portfolio returns to the DataFrame that contains all of the portfolio returns

In [ ]:
```
# Join your returns DataFrame to the original returns DataFrame
```

In [ ]:
```
# Only compare dates where return data exists for all the stocks (drop NaNs)
```

## Re-run the risk analysis with your portfolio to see how it compares to the others

## Calculate the Annualized Standard Deviation

In [ ]:
```
# Calculate the annualized `std`
```

## Calculate and plot rolling `std` with 21-day window

In [ ]:
```
# Calculate rolling standard deviation

# Plot rolling standard deviation
```

## Calculate and plot the correlation

In [ ]:
```
# Calculate and plot the correlation
```

## Calculate and Plot Rolling 60-day Beta for Your Portfolio compared to the S&P 500

In [ ]:
```
# Calculate and plot Beta
```

## Using the daily returns, calculate and visualize the Sharpe ratios using a bar plot

In [ ]:
```
# Calculate Annualzied Sharpe Ratios
```

In [ ]:
```
# Visualize the sharpe ratios as a bar plot
```

## How does your portfolio do?

Write your answer here!

In [ ]: