Before you turn this problem in, make sure everything runs as expected. First, **restart the kernel** (in the menu bar, select Kernel$\rightarrow$Restart) and then **run all cells** (in the menu bar, select Cell$\rightarrow$Run All).

Below, please fill in your name and collaborators:

In [1]:
```
NAME = "Jordan Vercillo"
COLLABORATORS = "Jordan Vercillo"
```

Assignment 3 - Time Series Analysis

(15 points total)

Assignment tasks:

In this assignment you will conduct time series analysis of the financial data.

1. Setup your environment to access and download latest stock data. Please see instructions below for different tools you can use to get the data. You can use any of the options provided, either Quandl or Yahoo Finance. If you know of any other service to download the data, please use that service, provide an explanation in the comments.
2. *(2 points)* Download the **adjusted** close prices for FB, MMM, IBM and AMZN for the last 60 months. If you run into any issues downloading the data from online sources, you can use `.csv` files provided. This will not affect your grade for the assignment.
3. *(3 points)* Resample the data to get prices for the end of the **business** month. Select the **Adjusted Close** for each stock.
4. *(3 points)* Use the pandas `autocorrelation_plot()` function to plot the autocorrelation of the adjusted month-end close prices for each of the stocks.
   - Are they autocorrelated?
   - Provide short explanation.
5. *(4 points)*
   - Calculate the monthly returns for each stock using the "shift trick" explained in the lecture, using `shift()` function.
   - Use pandas `autotocorrelation_plot()` to plot the autocorrelation of the monthly returns.
   - Are the returns autocorrelated? Provide short explanation.
6. *(3 points)*
   - Combine all 4 time series (returns) into a single DataFrame,

- Visualize the correlation between the returns of all pairs of stocks using a scatter plot matrix (use `scatter_matrix()` function from `pandas.plotting`).
- Explain the results. Is there any correlation?

NOTES:

1. In this assignment, please make sure the DataFrame(s) do not contain any NAs before you plot autocorrelations or scatter matrix.
2. Both options explained below use `pandas-datareader` package for remote data access. To install it, type the following in a command window: `conda install pandas-datareader`. You will also need to install one or more of the following packages `fix_yahoo_finance`, `quandl`. See below.

Downloading Stock Prices

Option 1 - Using QUANDL

To use QUANDL service, you need to create an account and get an API Key. Here is the short description of steps:

- Go to https://www.quandl.com/
- Click either `sign up` at the top right corner of the home page, or scroll all the way down and click `Create Free Account` button at the bottom of the page.
- Create an account.
- You will receive an email to the email address you have used during the registration. Confirm your email.

You are all set.

Now, as you login into your account, click the avatar icon at the top right corner of the page, select `"Account Settings."` On the next page, you will see `Your API Key` field with a long string of numbers and characters underneath. You need this API key for your call to Quandl from the notebook. In the code below, replace `YOUR_API_KEY` with the actual API key from your account.

NOTE: You can remove this key before submitting the assignment.

Question_1/2

Set up code for both options, I orginally downloaded stock prices using QUANDL but it looked like the libray only contained data up till 2018. I kept the code here for reference.

In [2]:
```
# all imports and env variables
import pandas as pd
pd.core.common.is_list_like = pd.api.types.is_list_like
```

```python
import datetime
import pandas_datareader.data as web
import os

os.environ['QUANDL_API_KEY'] = ""
```

In [3]:
```python
# Make sure you adjust the start and end date accordingly
# so that the start date = today date

start = datetime.datetime(2013, 11, 12)
end = datetime.datetime.now()

amzn_ql = web.DataReader('WIKI/AMZN', 'quandl', start, end)
fb_ql = web.DataReader('WIKI/FB', 'quandl', start, end)
ibm_ql = web.DataReader('WIKI/IBM', 'quandl', start, end)
mmm_ql = web.DataReader('WIKI/MMM', 'quandl', start, end)
```

In [4]:
```python
fb_ql
```

Out[4]:

Date	Open	High	Low	Close	Volume	ExDividend	SplitRatio	AdjOpen	Adjŀ
2018-03-27	156.31	162.85	150.75	152.190	76787884.0	0.0	1.0	156.31	16
2018-03-26	160.82	161.10	149.02	160.060	125438294.0	0.0	1.0	160.82	16
2018-03-23	165.44	167.10	159.02	159.390	52306891.0	0.0	1.0	165.44	16
2018-03-22	166.13	170.27	163.72	164.890	73389988.0	0.0	1.0	166.13	17
2018-03-21	164.80	173.40	163.30	169.390	105350867.0	0.0	1.0	164.80	17
...	...	...	...	...	...	...	...	...	
2013-11-18	48.47	48.84	45.80	45.830	85910000.0	0.0	1.0	48.47	4
2013-11-15	49.11	49.48	48.71	49.010	42453000.0	0.0	1.0	49.11	4
2013-11-14	48.70	49.57	48.03	48.990	75117000.0	0.0	1.0	48.70	4
2013-11-13	46.23	48.74	46.06	48.710	79245000.0	0.0	1.0	46.23	4
2013-11-12	46.00	47.37	45.83	46.605	68196000.0	0.0	1.0	46.00	4

1099 rows × 12 columns

In [5]:
```
fb_ql
```

Out[5]:

Date	Open	High	Low	Close	Volume	ExDividend	SplitRatio	AdjOpen	AdjH
2018-03-27	156.31	162.85	150.75	152.190	76787884.0	0.0	1.0	156.31	16
2018-03-26	160.82	161.10	149.02	160.060	125438294.0	0.0	1.0	160.82	16
2018-03-23	165.44	167.10	159.02	159.390	52306891.0	0.0	1.0	165.44	16
2018-03-22	166.13	170.27	163.72	164.890	73389988.0	0.0	1.0	166.13	17
2018-03-21	164.80	173.40	163.30	169.390	105350867.0	0.0	1.0	164.80	17
...	...	...	...	...	...	...	...	...	
2013-11-18	48.47	48.84	45.80	45.830	85910000.0	0.0	1.0	48.47	4
2013-11-15	49.11	49.48	48.71	49.010	42453000.0	0.0	1.0	49.11	4
2013-11-14	48.70	49.57	48.03	48.990	75117000.0	0.0	1.0	48.70	4
2013-11-13	46.23	48.74	46.06	48.710	79245000.0	0.0	1.0	46.23	4
2013-11-12	46.00	47.37	45.83	46.605	68196000.0	0.0	1.0	46.00	4

1099 rows × 12 columns

In [6]:
```
ibm_ql
```

Out[6]:

Date	Open	High	Low	Close	Volume	ExDividend	SplitRatio	AdjOpen	
2018-03-27	153.95	154.8697	151.160	151.91	3810994.0	0.0	1.0	153.950000	15
2018-03-26	151.21	153.6570	150.280	153.37	4038586.0	0.0	1.0	151.210000	15
2018-03-23	152.25	152.5800	148.541	148.89	4389015.0	0.0	1.0	152.250000	15
2018-03-22	155.00	155.2499	152.000	152.09	4617371.0	0.0	1.0	155.000000	15
2018-03-21	156.57	158.2000	155.920	156.69	3240695.0	0.0	1.0	156.570000	15
...	...	...	...	...	...	...	...	...	
2013-11-18	183.52	184.9900	183.270	184.47	5344900.0	0.0	1.0	160.908548	16
2013-11-15	182.38	183.2800	181.160	183.19	5176100.0	0.0	1.0	159.909007	16
2013-11-14	180.48	183.2000	179.660	182.21	6321500.0	0.0	1.0	158.243105	16
2013-11-13	182.27	183.5500	181.590	183.55	4704400.0	0.0	1.0	159.812560	16
2013-11-12	182.53	184.0487	182.260	183.07	4258500.0	0.0	1.0	160.040525	16

1099 rows × 12 columns

In [7]:

```
mmm_ql
```

Out[7]:

Date	Open	High	Low	Close	Volume	ExDividend	SplitRatio	AdjOpen	A
2018-03-07	231.22	236.22	230.590	235.57	2213792.0	0.0	1.0	231.220000	236.2
2018-03-06	234.05	235.92	230.800	233.66	2089047.0	0.0	1.0	234.050000	235.9
2018-03-05	230.00	233.71	228.530	232.81	2235348.0	0.0	1.0	230.000000	233.7
2018-03-02	229.75	231.27	226.330	230.37	2912828.0	0.0	1.0	229.750000	231.2
2018-03-01	236.15	236.83	229.530	231.34	3487126.0	0.0	1.0	236.150000	236.8
...	...	...	...	...	...	...	...	...	
2013-11-18	129.91	130.50	129.775	130.13	2148200.0	0.0	1.0	117.607560	118.1
2013-11-15	129.15	130.00	128.980	129.85	2360400.0	0.0	1.0	116.919532	117.6
2013-11-14	128.97	130.12	128.800	129.79	2569800.0	0.0	1.0	116.756578	117.7
2013-11-13	127.86	128.66	127.430	128.59	2426300.0	0.0	1.0	115.751695	116.4
2013-11-12	128.15	128.59	127.550	128.36	2428600.0	0.0	1.0	116.014232	116.4

1018 rows × 12 columns

Question1/2

Set up code and downloading of stock prices for AMZN, FB, IBM, and MMM from Yahoo Finance

In [8]:

```
#importing data science fundamental programs
import pandas as pd # For computations
import numpy as np # For indexing our data
# Our temporal data types
from datetime import datetime
from datetime import timedelta
#Used for importing and reading stock data
import yfinance as yf
from pandas_datareader import data as pdr

# 1) Using pandas datareader and Yahoo Finance
yf.pdr_override()
```

```
start = datetime(2019, 4, 27)
end = datetime(2024, 3, 27)


amzn = pdr.get_data_yahoo('AMZN', start = start)
fb = pdr.get_data_yahoo('META', start = start)
ibm = pdr.get_data_yahoo('IBM', start = start)
mmm = pdr.get_data_yahoo('MMM', start = start)
```

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

In [9]:
```
#Creating an all stocks list to validate data of all 4 stocks at once
all_stocks_list = ['AMZN', 'META','IBM','MMM']
all_stocks = yf.download(all_stocks_list, start = start)
```

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

In [10]:
```
all_stocks
```

Out[10]:

Price	Adj Close						
Ticker	AMZN	IBM	META	MMM	AMZN	IBM	META
Date							
2019-04-29	96.921501	104.422127	194.573547	154.065033	96.921501	132.934998	194.779999
2019-04-30	96.325996	105.338303	193.195007	153.498047	96.325996	134.101334	193.399994
2019-05-01	95.575996	105.556099	192.825409	150.711731	95.575996	134.378586	193.029999
2019-05-02	95.041000	104.827667	192.325943	149.642563	95.041000	133.451248	192.529999
2019-05-03	98.123001	105.323303	195.262817	150.023239	98.123001	134.082214	195.470001
...	...	...	...	...	...	...	...
2024-03-22	178.869995	190.839996	509.579987	106.779999	178.869995	190.839996	509.579987
2024-03-25	179.710007	188.789993	503.019989	104.839996	179.710007	188.789993	503.019989
2024-03-26	178.300003	188.500000	495.890015	102.629997	178.300003	188.500000	495.890015
2024-03-27	179.830002	190.800003	493.859985	104.589996	179.830002	190.800003	493.859985
2024-03-28	180.380005	190.960007	485.579987	106.070000	180.380005	190.960007	485.579987

1239 rows × 24 columns

In [11]:

```
#No NaNs we are good to go
all_stocks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1239 entries, 2019-04-29 to 2024-03-28
Data columns (total 24 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   (Adj Close, AMZN)  1239 non-null   float64
 1   (Adj Close, IBM)   1239 non-null   float64
 2   (Adj Close, META)  1239 non-null   float64
 3   (Adj Close, MMM)   1239 non-null   float64
 4   (Close, AMZN)      1239 non-null   float64
 5   (Close, IBM)       1239 non-null   float64
 6   (Close, META)      1239 non-null   float64
 7   (Close, MMM)       1239 non-null   float64
 8   (High, AMZN)       1239 non-null   float64
 9   (High, IBM)        1239 non-null   float64
 10  (High, META)       1239 non-null   float64
 11  (High, MMM)        1239 non-null   float64
 12  (Low, AMZN)        1239 non-null   float64
 13  (Low, IBM)         1239 non-null   float64
 14  (Low, META)        1239 non-null   float64
 15  (Low, MMM)         1239 non-null   float64
 16  (Open, AMZN)       1239 non-null   float64
 17  (Open, IBM)        1239 non-null   float64
 18  (Open, META)       1239 non-null   float64
 19  (Open, MMM)        1239 non-null   float64
 20  (Volume, AMZN)     1239 non-null   int64  
 21  (Volume, IBM)      1239 non-null   int64  
 22  (Volume, META)     1239 non-null   int64  
 23  (Volume, MMM)      1239 non-null   int64  
dtypes: float64(20), int64(4)
memory usage: 242.0 KB
```

In [12]:

```
# Using the Naive method, All forecasts are simply set to be the value of the last
#https://www.geeksforgeeks.org/python-pandas-dataframe-resample/

amzn_month_end = amzn['Adj Close'].resample('BM').last()
fb_month_end = fb['Adj Close'].resample('BM').last()
ibm_month_end = ibm['Adj Close'].resample('BM').last()
mmm_month_end = mmm['Adj Close'].resample('BM').last()
all_stocks_month_end = all_stocks['Adj Close'].resample('BM').last()
```

In [13]:

```
all_stocks_month_end
```

Out[13]:

Ticker	AMZN	IBM	META	MMM
Date				
2019-04-30	96.325996	105.338303	193.195007	153.498047
2019-05-31	88.753502	96.498276	177.281906	130.500351
2019-06-28	94.681503	104.788643	192.795441	141.602051
2019-07-31	93.338997	112.645905	194.024139	142.729370
2019-08-30	88.814499	104.201180	185.473206	133.318512
2019-09-30	86.795502	111.805038	177.891251	135.527847
2019-10-31	88.833000	102.817261	191.446869	136.014206
2019-11-29	90.040001	104.591766	201.426285	141.166412
2019-12-31	92.391998	104.272804	205.032455	146.695984
2020-01-31	100.435997	111.810875	201.695999	131.928299
2020-02-28	94.187500	102.303902	192.266006	125.219162
2020-03-31	97.486000	87.196106	166.623215	114.538071
2020-04-30	123.699997	98.695969	204.493042	127.467766
2020-05-29	122.118500	99.493866	224.851425	132.562149
2020-06-30	137.940994	96.203941	226.829346	132.180847
2020-07-31	158.233994	97.932541	253.401138	127.503380
2020-08-31	172.548004	99.513390	292.889252	139.395218
2020-09-30	157.436493	98.189888	261.622406	136.966766
2020-10-30	151.807495	90.111641	262.831116	136.778641
2020-11-30	158.401993	101.128319	276.676453	148.975723
2020-12-31	162.846497	103.060509	272.870483	150.752380
2021-01-29	160.309998	97.517784	258.056183	151.502777
2021-02-26	154.646500	98.671555	257.346954	152.229462
2021-03-31	154.703995	110.560600	294.217834	167.551544
2021-04-30	173.371002	117.712250	324.735443	171.429886
2021-05-31	161.153503	120.587883	328.381592	177.859741
2021-06-30	172.007996	122.978836	347.341461	173.996643
2021-07-30	166.379501	118.255676	355.922363	173.392212
2021-08-31	173.539505	119.091003	378.977905	171.891266

Ticker	AMZN	IBM	META	MMM
Date				
2021-09-30	164.251999	117.894501	339.030304	154.838058
2021-10-29	168.621506	106.158508	323.227051	157.715561
2021-11-30	175.353500	105.327812	324.116089	151.310730
2021-12-31	166.716995	120.223022	335.993500	158.064743
2022-01-31	149.573502	120.142052	312.927979	147.733551
2022-02-28	153.563004	111.521286	210.806335	133.555420
2022-03-31	162.997498	118.357651	222.124329	133.762070
2022-04-29	124.281502	120.351227	200.257523	129.575287
2022-05-31	120.209503	127.919662	193.434769	135.485931
2022-06-30	106.209999	130.084824	161.079086	117.444138
2022-07-29	134.949997	120.502831	158.931381	129.995361
2022-08-31	126.769997	119.837967	162.757309	114.010147
2022-09-30	113.000000	110.844299	135.536194	101.311798
2022-10-31	102.440002	129.018250	93.061264	115.330406
2022-11-30	96.540001	140.573166	117.974823	116.847809
2022-12-30	84.000000	133.011108	120.212448	111.235924
2023-01-31	103.129997	127.195595	148.812103	106.746422
2023-02-28	94.230003	123.568649	174.754593	101.261154
2023-03-31	103.290001	125.279297	211.715363	98.789314
2023-04-28	105.449997	120.806755	240.065292	99.832565
2023-05-31	120.580002	124.565796	264.439423	89.031967
2023-06-30	130.360001	129.622437	286.675842	95.501122
2023-07-31	133.679993	139.667908	318.262329	106.388008
2023-08-31	138.009995	143.871796	295.576416	103.301575
2023-09-29	127.120003	137.473358	299.891815	90.663666
2023-10-31	133.089996	141.725906	300.950684	88.077980
2023-11-30	146.089996	157.127487	326.803253	97.451347
2023-12-29	151.940002	162.072403	353.584839	107.533882
2024-01-31	155.199997	182.000717	389.726501	92.808464

Ticker	AMZN	IBM	META	MMM
Date				
2024-02-29	176.759995	185.029999	490.130005	92.120003
2024-03-29	180.380005	190.960007	485.579987	106.070000

In [14]:
```
all_stocks_month_end.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 60 entries, 2019-04-30 to 2024-03-29
Freq: BM
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   AMZN    60 non-null     float64
 1   IBM     60 non-null     float64
 2   META    60 non-null     float64
 3   MMM     60 non-null     float64
dtypes: float64(4)
memory usage: 2.3 KB
```

In [15]:
```
all_stocks_month_end.describe()
```

Out[15]:

Ticker	AMZN	IBM	META	MMM
count	60.000000	60.000000	60.000000	60.000000
mean	132.033657	119.345200	251.650583	129.816152
std	30.596246	22.021977	84.010040	24.412706
min	84.000000	87.196106	93.061264	88.077980
25%	101.939001	103.916012	192.663082	107.337017
50%	133.384995	118.075089	246.733215	132.371498
75%	158.878994	127.376612	314.261566	148.044094
max	180.380005	190.960007	490.130005	177.859741

In [16]:
```
#Importing for autocorrelation plot
import matplotlib.pyplot as plt
```

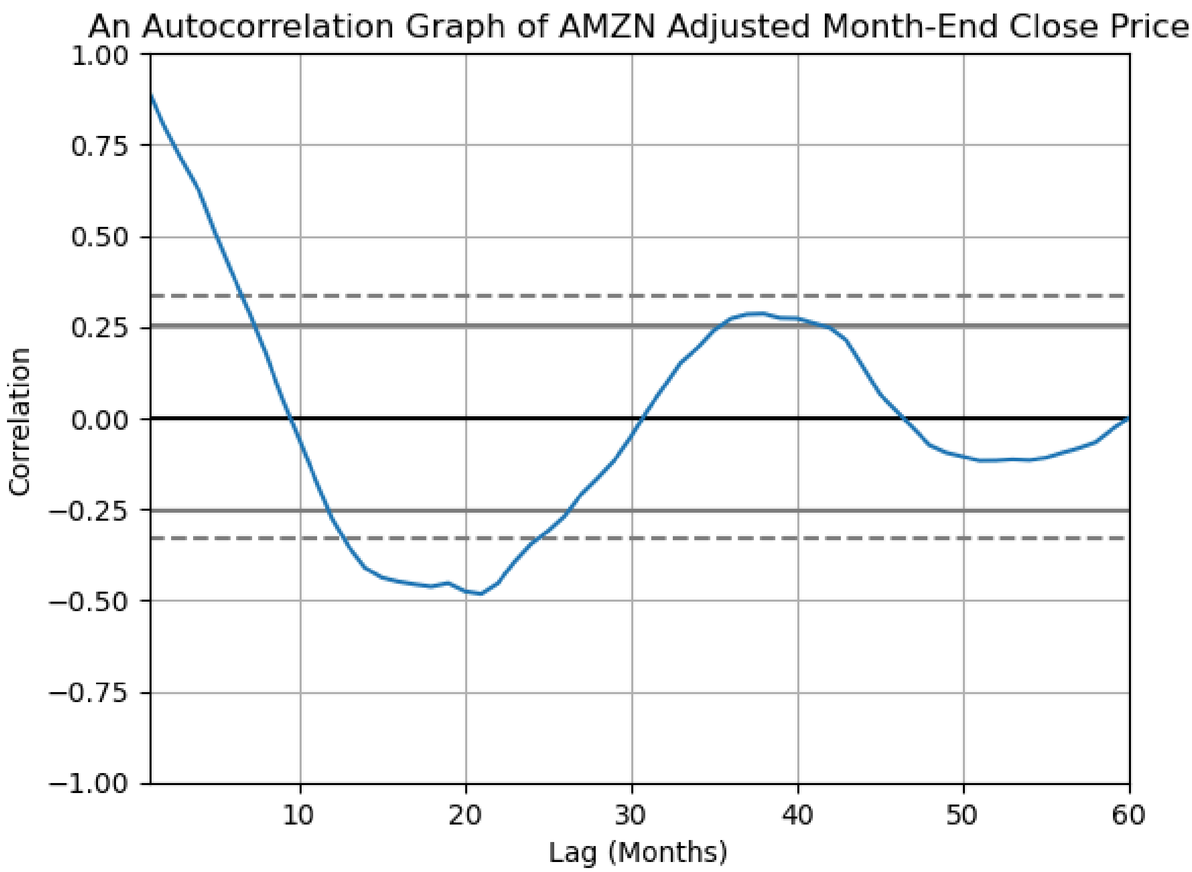
AMZN_Auto_Correlation

This graph shows positive autocorrelation that becomes statisically insignificant around 6/7 months lag (shown by the dashed lines). This suggests that while there is positive autocorrelation in the short term, it becomes much less predictable over a longer period of time.

The autocorrelation fluctuates around zero, occasionally dipping into negative territory between lags 12 and 24. This doesn't necessarily indicate an inverse relationship but rather a lack of consistent correlation, making predictions based on past returns unreliable in this period.

Overall this indicates that it is difficult to predicit the pattern due to the cyclical variation of the autocorrelation.

In [17]:
```
AMZN_cor_plot = pd.plotting.autocorrelation_plot(amzn_month_end)
AMZN_cor_plot.set_title("An Autocorrelation Graph of AMZN Adjusted Month-End Close
AMZN_cor_plot.set_ylabel("Correlation")
AMZN_cor_plot.set_xlabel("Lag (Months)")
plt.show()
```



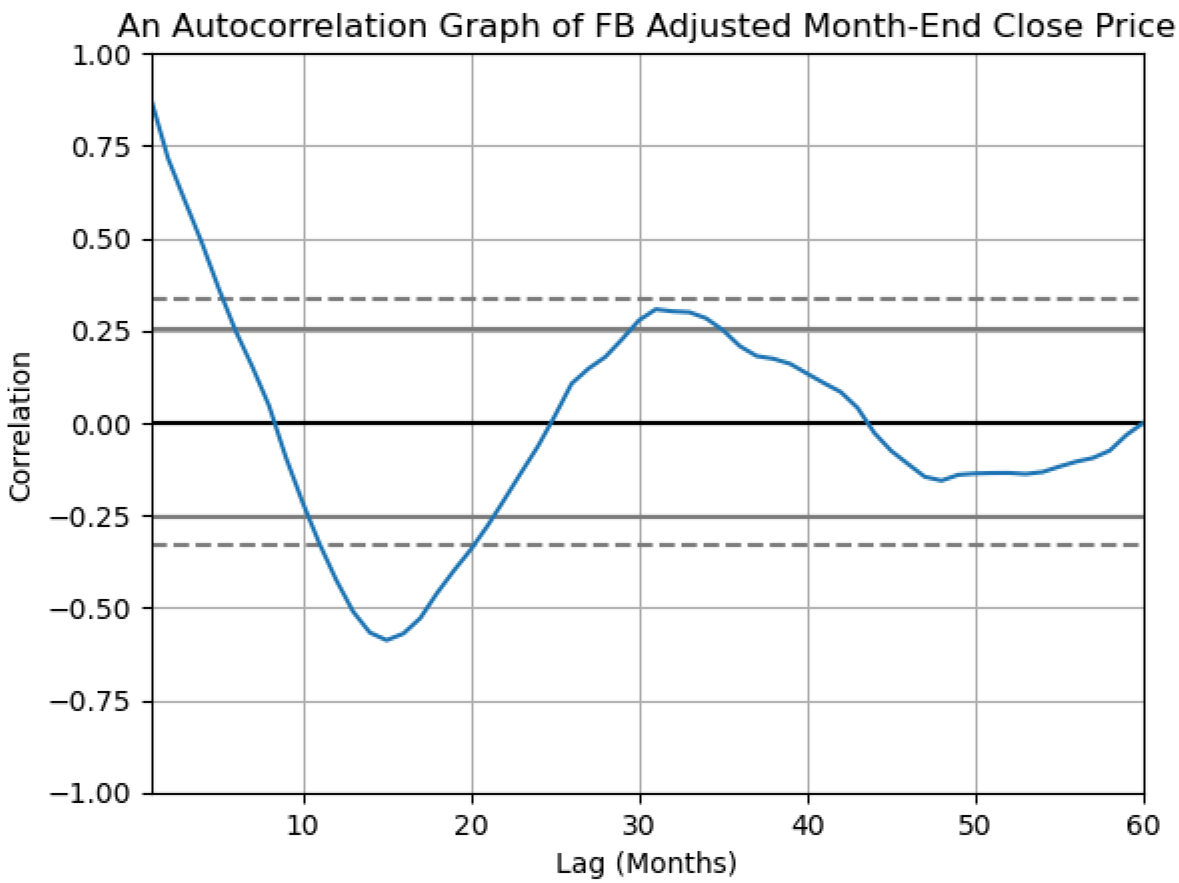
FB_Auto_Correlation

Similarly to AMZN, this graph shows positive autocorrelation that becomes statisically insignificant around 5 months lag (shown by the dashed lines). This suggests that while there is positive autocorrelation in the short term, it becomes much less predictable over a longer period of time.

The autocorrelation fluctuates around zero, occasionally dipping into negative territory between lags 11 and 20. This doesn't necessarily indicate an inverse relationship but rather a lack of consistent correlation, making predictions based on past returns unreliable in this period.

Overall this indicates that it is difficult to predicit the pattern due to the cyclical variation of the autocorrelation.

In [18]:
```
FB_cor_plot = pd.plotting.autocorrelation_plot(fb_month_end)
FB_cor_plot.set_title("An Autocorrelation Graph of FB Adjusted Month-End Close Pric
FB_cor_plot.set_ylabel("Correlation")
FB_cor_plot.set_xlabel("Lag (Months)")
plt.show()
```

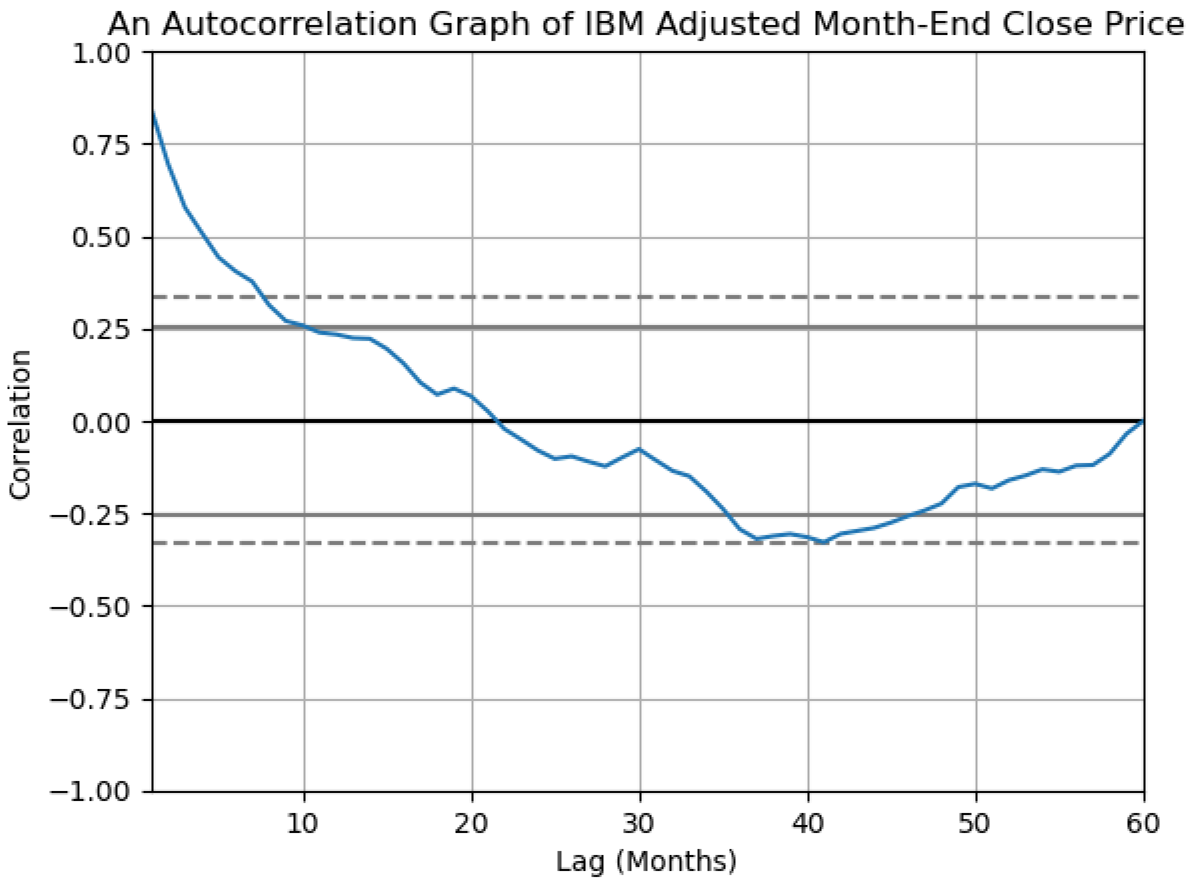


IBM_Auto_Correlation

This graph shows positive autocorrelation that becomes statisically insignificant around 8 months lag (shown by the dashed lines). This suggests that while there is positive autocorrelation in the short term, it becomes much less predictable over a longer period of time.

This doesn't have as much variation as AMZN or FB but is still not a great indicator of how the stock will preform 60 months from now.

In [19]:
```
IBM_cor_plot = pd.plotting.autocorrelation_plot(ibm_month_end)
IBM_cor_plot.set_title("An Autocorrelation Graph of IBM Adjusted Month-End Close Pr
IBM_cor_plot.set_ylabel("Correlation")
IBM_cor_plot.set_xlabel("Lag (Months)")
plt.show()
```

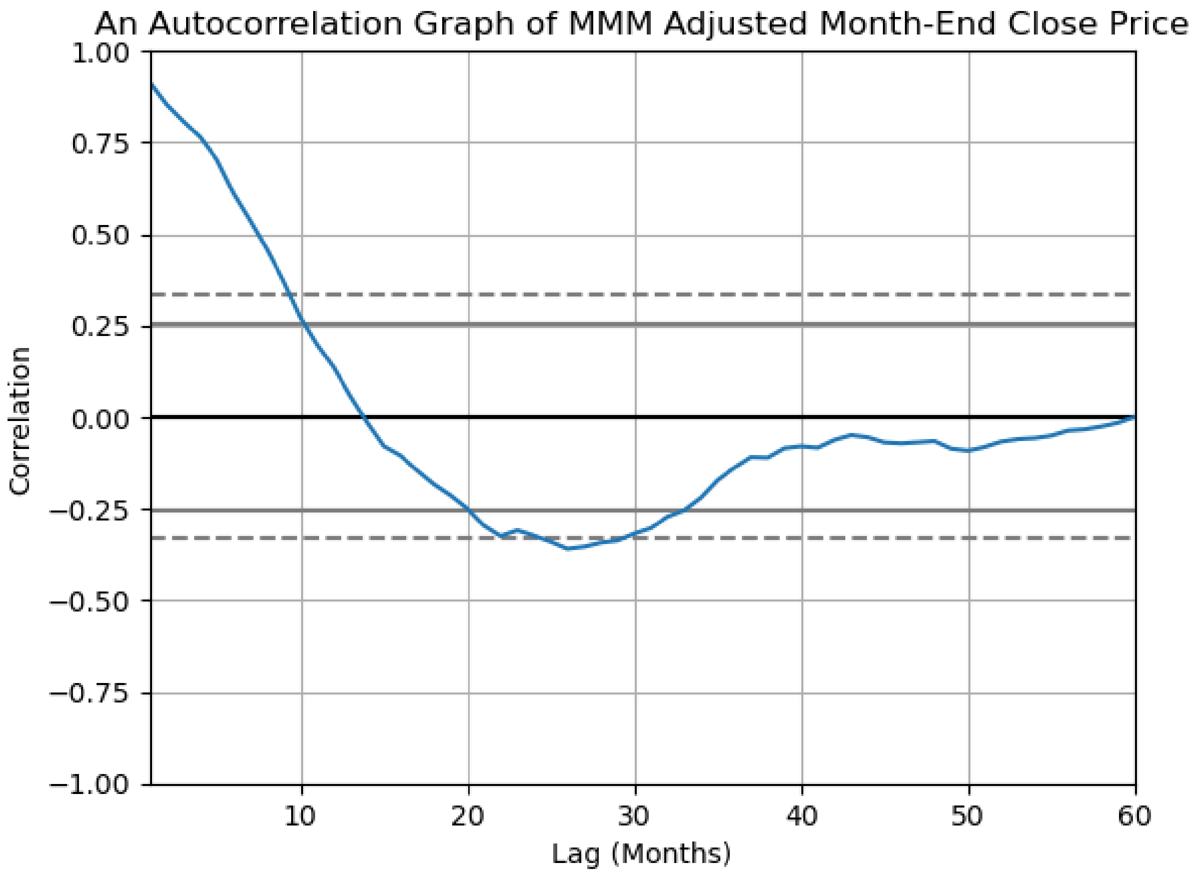


MMM_Auto_Correlation

This graph shows positive autocorrelation that becomes statisically insignificant around 9 months lag (shown by the dashed lines). This suggests that while there is positive autocorrelation in the short term, it becomes much less predictable over a longer period of time.

Smiilarly to IBM this doesn't have as much variation as AMZN or FB but is still not a great indicator of how the stock will preform 60 months from now.

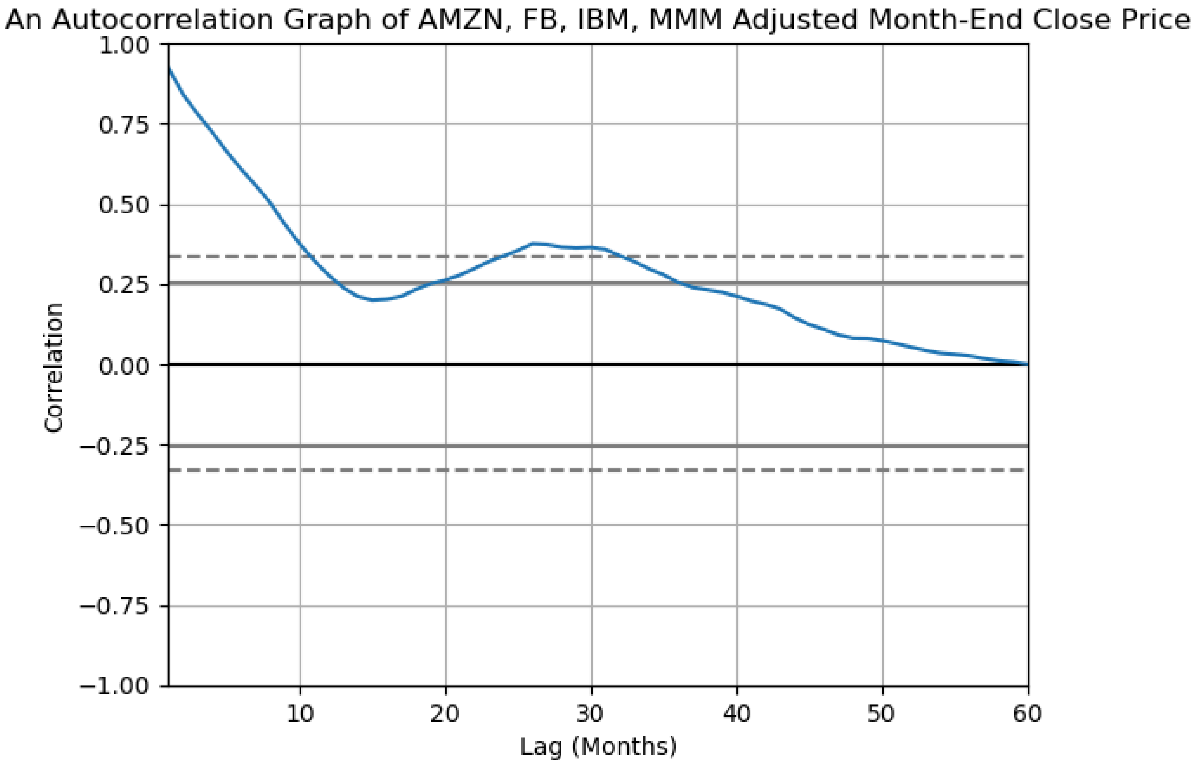
In [20]:
```
MMM_cor_plot = pd.plotting.autocorrelation_plot(mmm_month_end)
MMM_cor_plot.set_title("An Autocorrelation Graph of MMM Adjusted Month-End Close Pr
MMM_cor_plot.set_ylabel("Correlation")
MMM_cor_plot.set_xlabel("Lag (Months)")
plt.show()
```



ALL STOCK Auto Correlation

All stocks together show positive autocorrelation that becomes statisically insignificant around 10 months lag (shown by the dashed lines). This suggests that while there is positive autocorrelation in the short term, it becomes much less predictable over a longer period of time.

In [21]:
```
All_Stocks_cor_plot = pd.plotting.autocorrelation_plot(all_stocks_month_end)
All_Stocks_cor_plot.set_title("An Autocorrelation Graph of AMZN, FB, IBM, MMM Adjus
All_Stocks_cor_plot.set_ylabel("Correlation")
All_Stocks_cor_plot.set_xlabel("Lag (Months)")
plt.show()
```



QUESTION 5 SHIFT TRICK AND MONTHLY RETURNS

The next 4 graphs show the Monthly Return calculated using the shift trick and the growth forumula, (present - past / past)

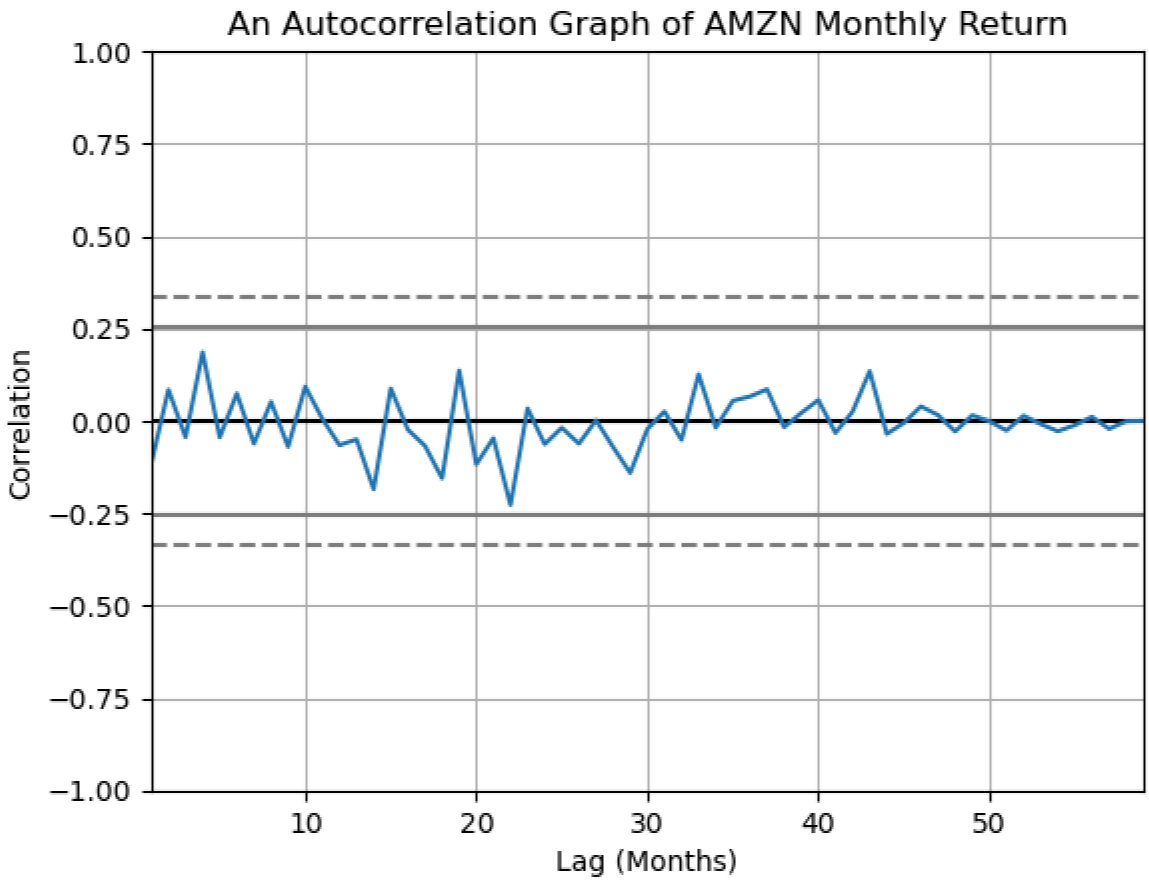
They suggest that there is no strong autocorrelation at any specific lag as the coefficients all fall within the the confidence bounds. This reinforces that randomness is often observed in stock prices patterns.

In [22]:
```
#AMZN Monthly Return Shift and AutoCorrelation

# With shift(), we can shift the `TimeSeries` (not the time range)
amzn_df = pd.DataFrame({"amzn_beforeShift":amzn_month_end, "amzn_afterShift" : amzn
"amzn_monthly_return" :  (amzn_month_end - amzn_month_end.shift(1)) / amzn_month_en

amzn_df = amzn_df.dropna()

amzn_monthly_return_cor_plot = pd.plotting.autocorrelation_plot(amzn_df["amzn_month
amzn_monthly_return_cor_plot.set_title("An Autocorrelation Graph of AMZN Monthly Re
amzn_monthly_return_cor_plot.set_ylabel("Correlation")
amzn_monthly_return_cor_plot.set_xlabel("Lag (Months)")
plt.show()
```

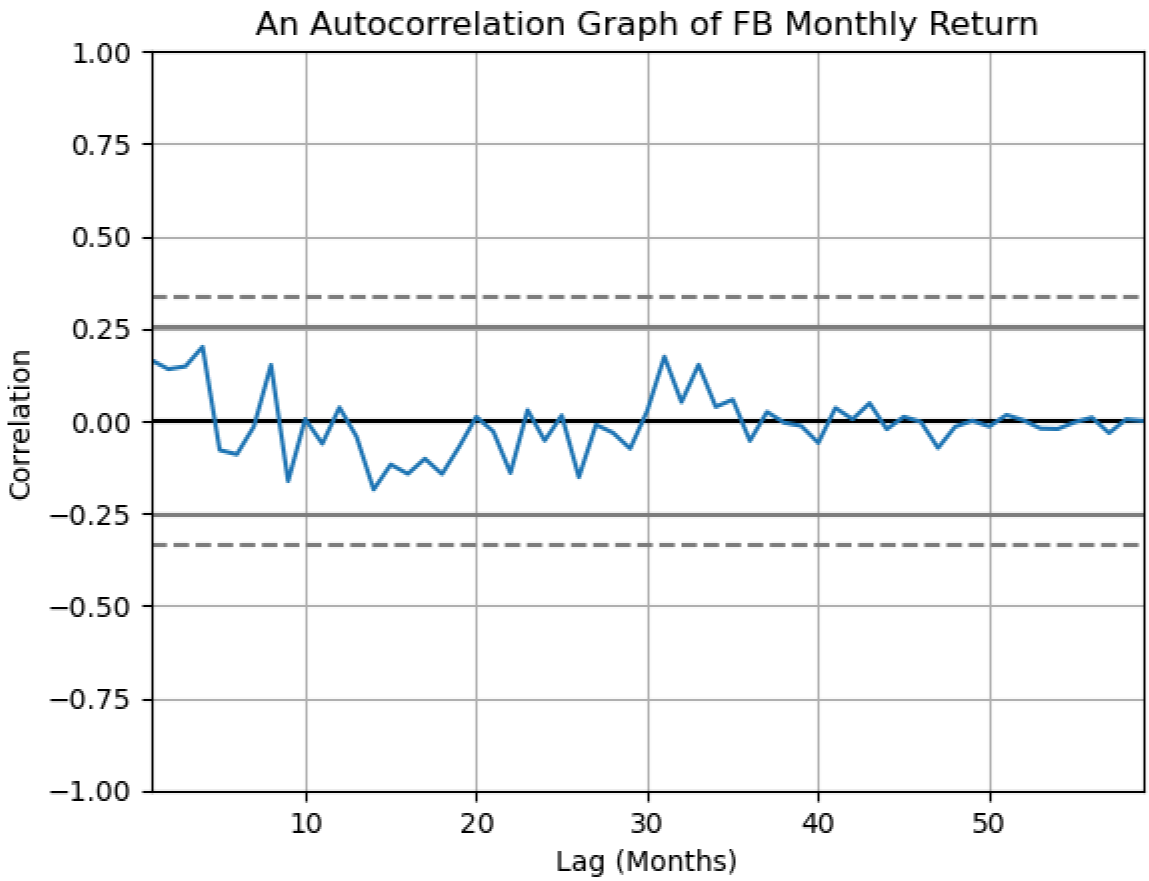


In [23]:
```
#FB Monthly Return Shift and AutoCorrelation

# With shift(), we can shift the `TimeSeries` (not the time range)
fb_df = pd.DataFrame({"fb_beforeShift":fb_month_end, "fb_afterShift" : fb_month_end
"fb_monthly_return" :  (fb_month_end - fb_month_end.shift(1)) / fb_month_end.shift(

fb_df = fb_df.dropna()

fb_monthly_return_cor_plot = pd.plotting.autocorrelation_plot(fb_df["fb_monthly_ret
fb_monthly_return_cor_plot.set_title("An Autocorrelation Graph of FB Monthly Return
fb_monthly_return_cor_plot.set_ylabel("Correlation")
fb_monthly_return_cor_plot.set_xlabel("Lag (Months)")
plt.show()
```

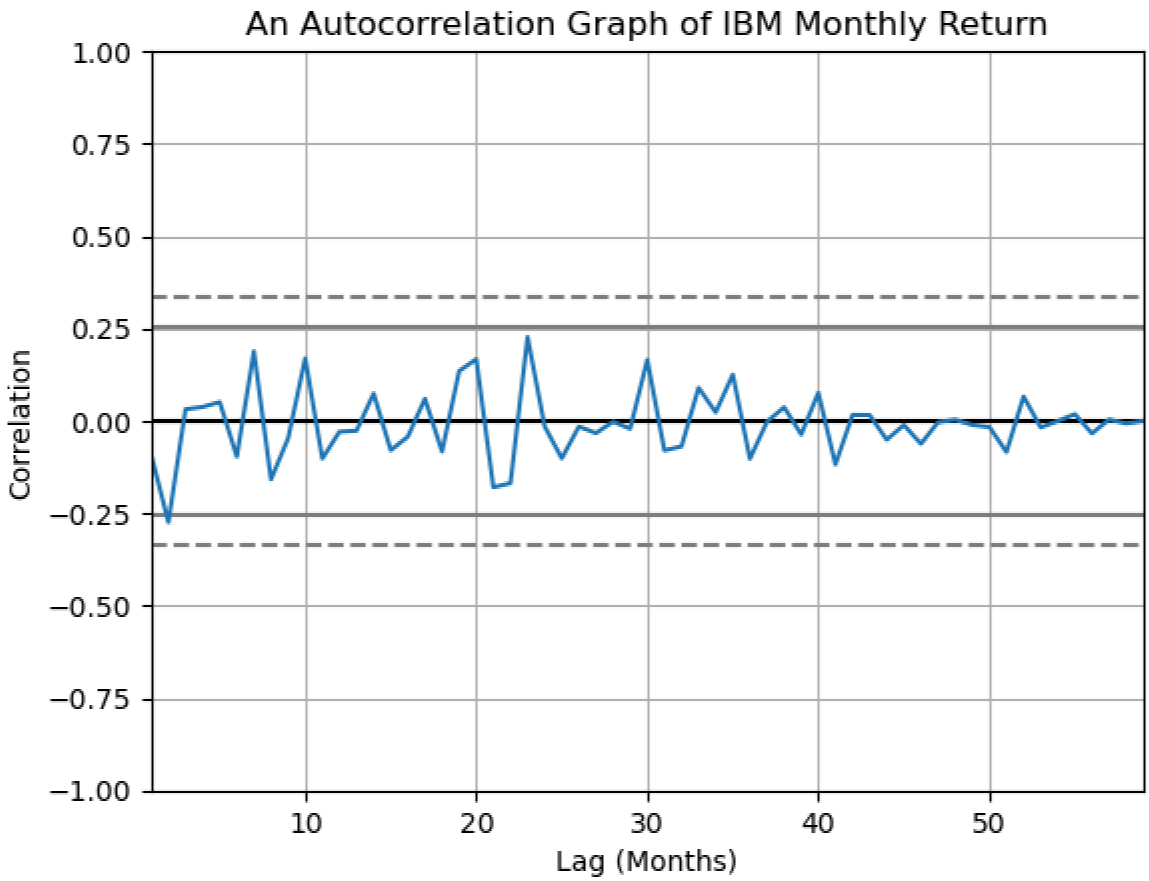


In [24]:
```
#IBM Monthly Return Shift and AutoCorrelation

# With shift(), we can shift the `TimeSeries` (not the time range)
ibm_df = pd.DataFrame({"ibm_beforeShift":ibm_month_end, "ibm_afterShift" : ibm_mont
"ibm_monthly_return" :  (ibm_month_end - ibm_month_end.shift(1)) / ibm_month_end.sh

ibm_df = ibm_df.dropna()

ibm_monthly_return_cor_plot = pd.plotting.autocorrelation_plot(ibm_df["ibm_monthly_
ibm_monthly_return_cor_plot.set_title("An Autocorrelation Graph of IBM Monthly Retu
ibm_monthly_return_cor_plot.set_ylabel("Correlation")
ibm_monthly_return_cor_plot.set_xlabel("Lag (Months)")
plt.show()
```

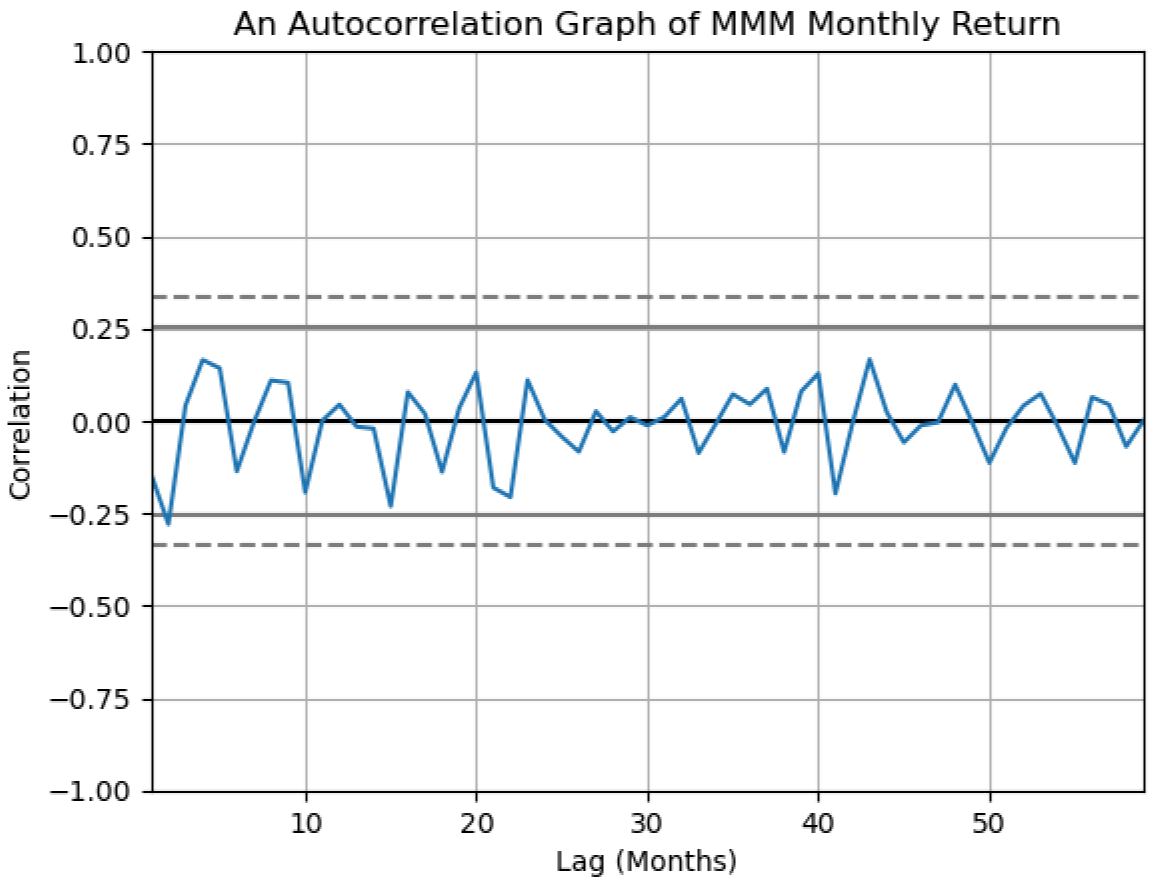


In [25]:
```
#MMM Monthly Return Shift and AutoCorrelation

# With shift(), we can shift the `TimeSeries` (not the time range)
mmm_df = pd.DataFrame({"mmm_beforeShift":mmm_month_end, "mmm_afterShift" : mmm_mont
"mmm_monthly_return" :  (mmm_month_end - mmm_month_end.shift(1)) / mmm_month_end.sh

mmm_df = mmm_df.dropna()

mmm_monthly_return_cor_plot = pd.plotting.autocorrelation_plot(mmm_df["mmm_monthly_
mmm_monthly_return_cor_plot.set_title("An Autocorrelation Graph of MMM Monthly Retu
mmm_monthly_return_cor_plot.set_ylabel("Correlation")
mmm_monthly_return_cor_plot.set_xlabel("Lag (Months)")
plt.show()
```



In [26]:
```
amzn_returns = amzn_df['amzn_monthly_return']
fb_returns = fb_df['fb_monthly_return']
ibm_returns = ibm_df['ibm_monthly_return']
mmm_returns = mmm_df['mmm_monthly_return']


all_stocks_returns_df = pd.DataFrame({
    'AMZN': amzn_returns,
    'FB': fb_returns,
    'IBM': ibm_returns,
    'MMM': mmm_returns
})

all_stocks_returns_df
```

Out[26]:

Date	AMZN	FB	IBM	MMM
2019-05-31	-0.078613	-0.082368	-0.083921	-0.149824
2019-06-28	0.066792	0.087508	0.085912	0.085070
2019-07-31	-0.014179	0.006373	0.074982	0.007961
2019-08-30	-0.048474	-0.044071	-0.074967	-0.065935
2019-09-30	-0.022733	-0.040879	0.072972	0.016572
2019-10-31	0.023475	0.076202	-0.080388	0.003589
2019-11-29	0.013587	0.052126	0.017259	0.037880
2019-12-31	0.026122	0.017903	-0.003050	0.039171
2020-01-31	0.087064	-0.016273	0.072292	-0.100669
2020-02-28	-0.062214	-0.046753	-0.085027	-0.050854
2020-03-31	0.035021	-0.133371	-0.147676	-0.085299
2020-04-30	0.268900	0.227278	0.131885	0.112886
2020-05-29	-0.012785	0.099555	0.008084	0.039966
2020-06-30	0.129567	0.008797	-0.033066	-0.002876
2020-07-31	0.147114	0.117144	0.017968	-0.035387
2020-08-31	0.090461	0.155832	0.016143	0.093267
2020-09-30	-0.087579	-0.106753	-0.013300	-0.017421
2020-10-30	-0.035754	0.004620	-0.082272	-0.001374
2020-11-30	0.043440	0.052678	0.122256	0.089174
2020-12-31	0.028058	-0.013756	0.019106	0.011926
2021-01-29	-0.015576	-0.054291	-0.053781	0.004978
2021-02-26	-0.035328	-0.002748	0.011831	0.004797
2021-03-31	0.000372	0.143273	0.120491	0.100651
2021-04-30	0.120663	0.103725	0.064685	0.023147
2021-05-31	-0.070470	0.011228	0.024429	0.037507
2021-06-30	0.067355	0.057737	0.019827	-0.021720
2021-07-30	-0.032722	0.024705	-0.038406	-0.003474
2021-08-31	0.043034	0.064777	0.007064	-0.008656
2021-09-30	-0.053518	-0.105409	-0.010047	-0.099209

	AMZN	FB	IBM	MMM
Date				
2021-10-29	0.026602	-0.046613	-0.099547	0.018584
2021-11-30	0.039924	0.002751	-0.007825	-0.040610
2021-12-31	-0.049252	0.036646	0.141418	0.044637
2022-01-31	-0.102830	-0.068649	-0.000673	-0.065361
2022-02-28	0.026673	-0.326342	-0.071755	-0.095971
2022-03-31	0.061437	0.053689	0.061301	0.001547
2022-04-29	-0.237525	-0.098444	0.016844	-0.031300
2022-05-31	-0.032764	-0.034070	0.062886	0.045616
2022-06-30	-0.116459	-0.167269	0.016926	-0.133164
2022-07-29	0.270596	-0.013333	-0.073660	0.106870
2022-08-31	-0.060615	0.024073	-0.005517	-0.122968
2022-09-30	-0.108622	-0.167250	-0.075049	-0.111379
2022-10-31	-0.093451	-0.313384	0.163959	0.138371
2022-11-30	-0.057595	0.267711	0.089560	0.013157
2022-12-30	-0.129894	0.018967	-0.053795	-0.048027
2023-01-31	0.227738	0.237909	-0.043722	-0.040360
2023-02-28	-0.086299	0.174331	-0.028515	-0.051386
2023-03-31	0.096148	0.211501	0.013844	-0.024411
2023-04-28	0.020912	0.133906	-0.035701	0.010560
2023-05-31	0.143480	0.101531	0.031116	-0.108187
2023-06-30	0.081108	0.084089	0.040594	0.072661
2023-07-31	0.025468	0.110182	0.077498	0.113997
2023-08-31	0.032391	-0.071281	0.030099	-0.029011
2023-09-29	-0.078907	0.014600	-0.044473	-0.122340
2023-10-31	0.046963	0.003531	0.030934	-0.028520
2023-11-30	0.097678	0.085903	0.108672	0.106421
2023-12-29	0.040044	0.081950	0.031471	0.103462
2024-01-31	0.021456	0.102215	0.122959	-0.136937
2024-02-29	0.138918	0.257626	0.016644	-0.007418

	AMZN	FB	IBM	MMM
Date				
2024-03-29	0.020480	-0.009283	0.032049	0.151433

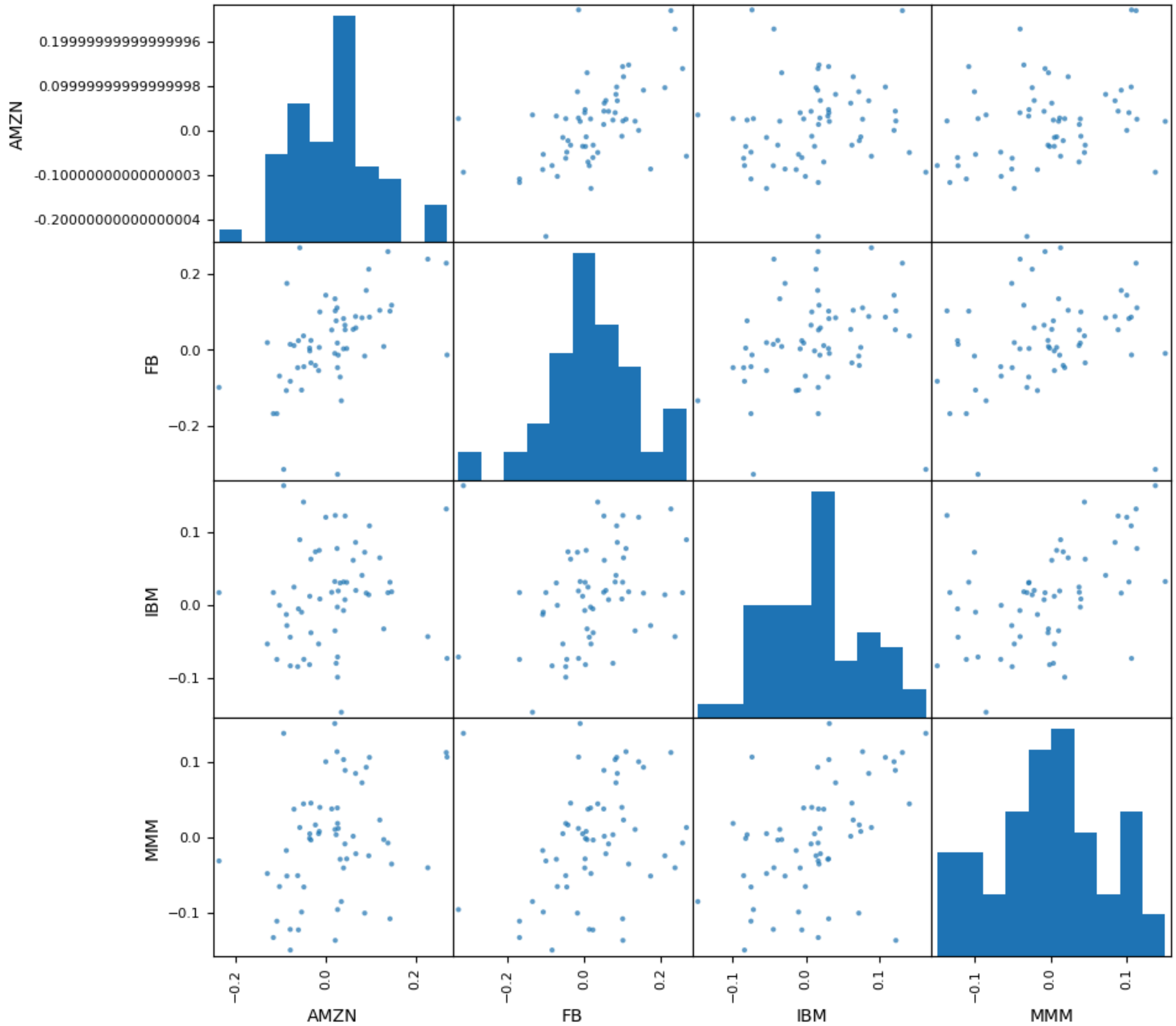
Question 6

The scatter plot matrix suggests that there might be a low to moderate relationship between the monthly returns of these stocks, but none of the relationships appears to be very strong. FB and AMZN have a low postive linear relationship, the same could be said about MMM and IBM or FB and MM but none are strong or good indicators of how the stocks would preform together. While the returns sometimes move in the same direction there is not enough of a relationship to imply a reliable predictive relationship Ie. If Facebook were to increase it does not mean that AMZN would also increase.

This could be an example of where the movements of these companies are not tightly tied to each other. This could be a good example of a portfolio that would minimze risk due to the independant nature of the stocks preformance.

In [27]:
```
#Importing scatter matrix
from pandas.plotting import scatter_matrix
```

In [28]:
```
All_stocks_scatter_plot = scatter_matrix(all_stocks_returns_df, alpha=0.70, figsize
```


0.19999999999999996
0.09999999999999998
0.0
-0.10000000000000003
-0.20000000000000004
AMZN
0.2
0.0
−0.2
FB
0.1
0.0
−0.1
IBM
0.1
0.0
−0.1
MMM
−0.2
0.0
0.2
AMZN
−0.2
0.0
0.2
FB
−0.1
0.0
0.1
IBM
−0.1
0.0
0.1
MMM