All of these stocks are resellers of Apple Products aside from Bitcoin. My motivation to see if they move in a covariant way is because if Apple sells well in these resellers, then Apple also does well. Judging by the study, CLuster 2 would make the most sense to invest in but, also an area of opportunity for growth for Apple.

```
!pip install yfinance
!pip install vega_datasets
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: yfinance in /usr/local/lib/python3.10/dist-packages (0.2.18)
Requirement already satisfied: pandas>=1.3.0 in /usr/local/lib/python3.10/dist-packages (from yfinance) (1.5.3)
Requirement already satisfied: numpy>=1.16.5 in /usr/local/lib/python3.10/dist-packages (from yfinance) (1.22.4)
Requirement already satisfied: requests>=2.26 in /usr/local/lib/python3.10/dist-packages (from yfinance) (2.27.1)
Requirement already satisfied: multitasking>=0.0.7 in /usr/local/lib/python3.10/dist-packages (from yfinance) (0.0.11)
Requirement already satisfied: lxml>=4.9.1 in /usr/local/lib/python3.10/dist-packages (from yfinance) (4.9.2)
Requirement already satisfied: appdirs>=1.4.4 in /usr/local/lib/python3.10/dist-packages (from yfinance) (1.4.4)
Requirement already satisfied: pytz>=2022.5 in /usr/local/lib/python3.10/dist-packages (from yfinance) (2022.7.1)
Requirement already satisfied: frozendict>=2.3.4 in /usr/local/lib/python3.10/dist-packages (from yfinance) (2.3.7)
Requirement already satisfied: cryptography>=3.3.2 in /usr/local/lib/python3.10/dist-packages (from yfinance) (40.0.2)
Requirement already satisfied: beautifulsoup4>=4.11.1 in /usr/local/lib/python3.10/dist-packages (from yfinance) (4.11.2)
Requirement already satisfied: html5lib>=1.1 in /usr/local/lib/python3.10/dist-packages (from yfinance) (1.1)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4>=4.11.1->yfinance) (2.4.1)
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.10/dist-packages (from cryptography>=3.3.2->yfinance) (1.15.1)
Requirement already satisfied: six>=1.9 in /usr/local/lib/python3.10/dist-packages (from html5lib>=1.1->yfinance) (1.16.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.3.0->yfinance) (2.8.2)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26->yfinance) (1.26.15
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26->yfinance) (2022.12.7)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26->yfinance) (2.0
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26->yfinance) (3.4)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from cffi>=1.12->cryptography>=3.3.2->yfinance) (2.
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: vega_datasets in /usr/local/lib/python3.10/dist-packages (0.9.0)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from vega_datasets) (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas->vega_datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->vega_datasets) (2022.7.1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas->vega_datasets) (1.22.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas->vega_datasets)
```

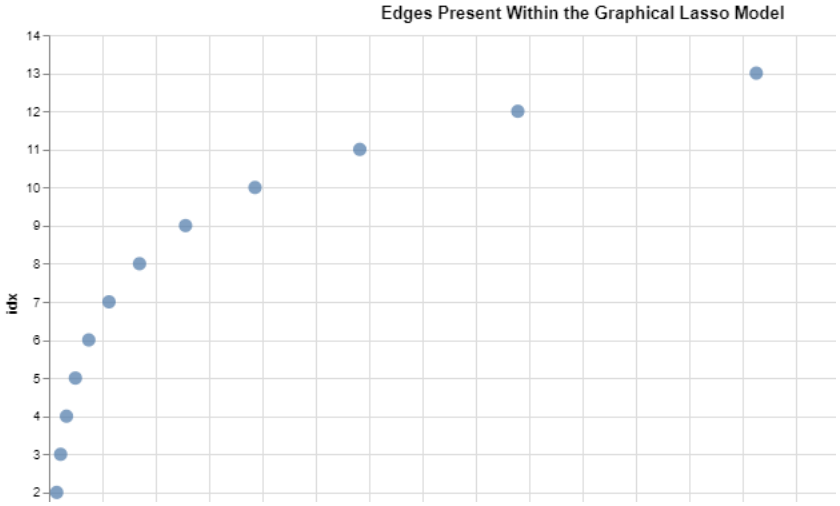
```
import yfinance as yf
from time import time,ctime, clock_gettime
from time import gmtime, time, time_ns

def ifs(input):
    ni = ''
    if input =='gff':
        input = 'GFF'
        ni = "GF=F"
    elif input == 'zff':
        input = 'ZFF'
        ni = "ZF=F"
    else:
        input = input.upper()
        ins = "="
        before = "F"
        ni = input.replace(before, ins + before , 1)
    print(ni)
    data = yf.download(
        tickers = ni,
        period = "1y",
        interval = "1d",
        group_by = 'ticker',
        auto_adjust = True,
        prepost = True,
        threads = True,
        proxy = None
    )
    epoch = ctime()
    filename = input
    data.to_csv(filename)


import numpy as np
import pandas as pd
from scipy.stats import pearsonr
```

```
symbol_dict = {"aapl": "Apple Inc.", "t": "AT&T Inc.", "btcf": "Bitcoin",
"tgt": "Target Corporation", "bby": "Best Buy Co., Inc.", "cost": "Costco Wholesale Corporation",
"amzn": "Amazon.com, Inc.", "vz": "Verizon Communications Inc.", "tmus": "T-Mobile US, Inc.", "wmt": "Walmart Inc."}

sym, names = np.array(sorted(symbol_dict.items())).T

for i in sym:
    ifs(i)

quotes = []
lens = []
for symbol in sym:
    symbol = symbol.upper()
    t = pd.read_csv(symbol)
    lens.append(t.shape[0])
mm = np.amin(lens)-1
print("min length of data: ",mm)

for symbol in sym:
    symbol = symbol.upper()
    t = pd.read_csv(symbol)
    t= t.truncate(after=mm)
    quotes.append(t)
mi = np.vstack([q["Close"] for q in quotes]) #min
ma = np.vstack([q["Open"] for q in quotes]) #max

volatility = ma - mi
```

```
 AAPL
[**********100%**********]  1 of 1 completed
AMZN
[**********100%**********]  1 of 1 completed
BBY
[**********100%**********]  1 of 1 completed
BTC=F
[**********100%**********]  1 of 1 completed
COST
[**********100%**********]  1 of 1 completed
T
[**********100%**********]  1 of 1 completed
TGT
[**********100%**********]  1 of 1 completed
TMUS
[**********100%**********]  1 of 1 completed
VZ
[**********100%**********]  1 of 1 completed
WMT
[**********100%**********]  1 of 1 completed
min length of data:  250
```

```
from sklearn import covariance
import altair as alt
alphas = np.logspace(-1.5, 1, num=15)
edge_model = covariance.GraphicalLassoCV(alphas=alphas)
X = volatility.copy().T
X /= X.std(axis=0)
l =edge_model.fit(X)
n= []
print(type(l.alphas))
for  i in range(len(l.alphas)):
    print(l.alphas[i])
    dict = {"idx":i , "alpha":l.alphas[i]}
    n.append(dict)

dd = pd.DataFrame(n)
alt.Chart(dd).mark_point(filled=True, size=100).encode(
    y=alt.Y('idx'),
    x=alt.X('alpha'),tooltip=['alpha'],).properties(
        width=800,
        height=400,
        title="Edges Present Within the Graphical Lasso Model"
    ).interactive()
```

```
<class 'numpy.ndarray'>
0.03162277660168379
0.047705826961439296
0.07196856730011521
0.10857111194022041
0.16378937069540642
0.2470911227985605
0.372759372031494
0.5623413251903491
0.8483428982440722
1.279802213997954
1.9306977288832505
2.9126326549087382
4.39397056076079
6.628703161826448
10.0
```




```
from sklearn import cluster

_, labels = cluster.affinity_propagation(edge_model.covariance_, random_state=0)
n_labels = labels.max()

gdf = pd.DataFrame()
for i in range(n_labels + 1):
    print(f"Cluster {i + 1}: {', '.join(np.array(sym)[labels == i])}")
    l = np.array(sym)[labels == i]
    ss = np.array(names)[labels == i]
    dict = {"cluster":(i+1), "symbols":l, "size":len(l), "names":ss}
    gdf = gdf.append(dict, ignore_index=True, sort=True)

gdf.head(15)
```

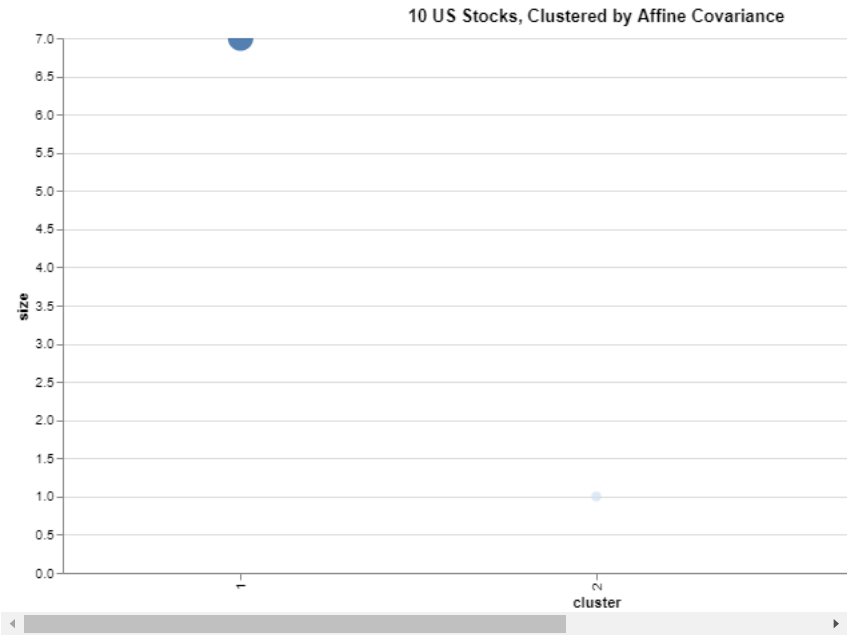
```
Cluster 1: aapl, amzn, bby, cost, tgt, tmus, wmt
Cluster 2: btcf
Cluster 3: t, vz
<ipython-input-66-0a4fe66658ad>:12: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future

<ipython-input-66-0a4fe66658ad>:12: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future

<ipython-input-66-0a4fe66658ad>:12: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future
```

	cluster	names	size	symbols
0	1	[Apple Inc., Amazon.com, Inc., Best Buy Co., I...	7	[aapl, amzn, bby, cost, tgt, tmus, wmt]
1	2	[Bitcoin]	1	[btcf]

```
for i in gdf['cluster']:
    print("cluster ",i)
    d = gdf[gdf['cluster'].eq(i)]
    for j in d.names:
        print(j, ", ")
```

```
 cluster  1
 ['Apple Inc.' 'Amazon.com, Inc.' 'Best Buy Co., Inc.'
  'Costco Wholesale Corporation' 'Target Corporation' 'T-Mobile US, Inc.'
  'Walmart Inc.'] ,
 cluster  2
 ['Bitcoin'] ,
 cluster  3
 ['AT&T Inc.' 'Verizon Communications Inc.'] ,
```

```
import altair as alt
def runCluster():
    c = alt.Chart(gdf).mark_circle(size=60).encode(
        x= alt.X('cluster:N'),
        y= alt.Y('size:Q'),
        color='size:Q',
        tooltip=['names'],
        size=alt.Size('size:Q')
    ).properties(
        width=800,
        height=400,
        title="10 US Stocks, Clustered by Affine Covariance"
    ).interactive()
    #.configure_title("40 Top Global Commodities, Clustered by Affine Covariance")

    chart =c
    return chart
runCluster()
```



Based on this experiment, Apple, Amazon, Best Buy, Costco, Target, and Walmart all move together in the past year. One pattern I observed, is that despite T-Mobile, AT&T, and Verizon being also being resellers of Apple products, they still dont follow the same trend as the other corporations.

```
!pip install plotly
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (5.13.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly) (8.2.2)
```

```
import plotly.graph_objects as go
```

```
import pandas as pd
from datetime import datetime

df_symbol = pd.read_csv('AAPL')
```

```
df_symbol.columns
```

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

```
df_symbol.head()
```

	Date	Open	High	Low	Close	Volume
0	2022-05-18	145.978337	146.485304	139.069578	139.984131	109742900
1	2022-05-19	139.049710	140.819143	135.789181	136.534729	136095600
2	2022-05-20	138.264381	139.864825	131.822850	136.773285	137426100
3	2022-05-23	136.972114	142.409647	136.832946	142.260544	117726300
4	2022-05-24	139.974181	141.127299	136.514842	139.526855	104132700

```
fig = go.Figure(data=[go.Candlestick(x=df_symbol['Date'],
                open=df_symbol['Open'],
                high=df_symbol['High'],
                low=df_symbol['Low'],
                close=df_symbol['Close'])])

fig.show()
```


```
import plotly.express as px

df2 = px.data.stocks()
fig = px.line(df2, x='date', y="AAPL")
fig.show()
```


```
df2.columns
```

```
Index(['date', 'GOOG', 'AAPL', 'AMZN', 'FB', 'NFLX', 'MSFT'], dtype='object')
```

```
df2.head(2)
```

	date	GOOG	AAPL	AMZN	FB	NFLX	MSFT
0	2018-01-01	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1	2018-01-08	1.018172	1.011943	1.061881	0.959968	1.053526	1.015988

```
df2['AAPL']
```

```
0      1.000000
1      1.011943
2      1.019771
3      0.980057
4      0.917143
         ...
100    1.546914
101    1.572286
102    1.596800
103    1.656000
104    1.678000
Name: AAPL, Length: 105, dtype: float64
```

```
df_symbol.columns
```

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

```
df_symbol['Close']
```

```
0      139.984131
1      136.534729
2      136.773285
3      142.260544
4      139.526855
          ...
246    173.510010
247    172.570007
248    172.070007
249    172.070007
250    172.690002
Name: Close, Length: 251, dtype: float64
```

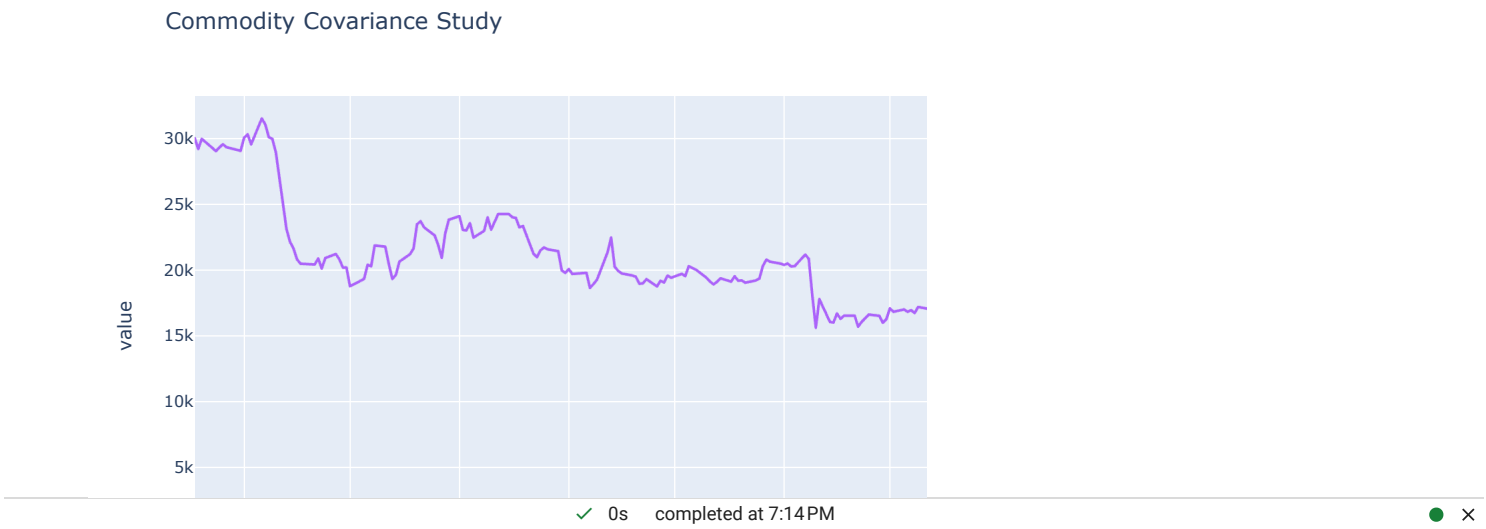
```
import plotly.express as px
fig = px.line(df_symbol, x='Date', y="Close")
fig.show()
```


```
def getDateColumn():
  df = pd.read_csv('AAPL')
  return df['Date']
```

```
symUpper = [x.upper() for x in sym] #make all symbols in sym to uppercase
# print(symUpper)
gdf = pd.DataFrame(columns=symUpper) #form a new global dataframe, gdf, for purpose of graphing
gdf['Date'] = getDateColumn()            #get a common index for dates, for every commodity or equity
for i in range(len(symUpper)):           #iterate the length of the uppercase symbols
  df_x = pd.read_csv( symUpper[i])       #create one dataframe to hold the csv contents
  gdf[symUpper[i]] = df_x['Close']       #extract the price series from the 'Closed' column
print(gdf.head(3))
```

```
        AAPL        AMZN        BBY     BTCF        COST          T  \
0  139.984131  107.112503  72.140831  30000.0  426.255035  19.066074
1  136.534729  107.319000  69.946655  29185.0  419.832397  19.047224
2  136.773285  107.591003  69.030815  29945.0  413.380005  19.226294

         TGT        TMUS         VZ         WMT        Date
0  157.540955  125.239998  45.986362  120.487877  2022-05-18
1  149.566895  125.879997  46.183895  117.181183  2022-05-19
2  151.448318  126.040001  46.588352  117.309120  2022-05-20
```

```
fig = px.line(gdf, x="Date", y=gdf.columns,
              hover_data={"Date": "|%B %d, %Y"},
              title='Commodity Covariance Study')
fig.update_xaxes(
    dtick="M1",
    tickformat="%b\n%Y")
fig.show()
```


Commodity Covariance Study
30k
25k
20k
15k
10k
5k
value
0s completed at 7:14 PM