```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import pingouin as pg
from sklearn.preprocessing import MinMaxScaler
```

Additional Material

Time Series Plots available in Tableau

https://public.tableau.com/app/profile/mohammed.bookwala/vizzes

Final Presentation Link

https://gamma.app/docs/Correlation-Analysis-of-Global-Commodities-and-US-Stock-Indices-2-sw6v7xizjed4t3w

Final Report Link

https://drive.google.com/file/d/1sAN8z4UdXO2PzBvDjLMyUde28uWsl6MJ/view?usp=sharing

Datasets

```python
path = r"C:\Files\College_Files\\"

df = pd.read_csv(path+"Stock Market Dataset.csv")
df1 = pd.read_csv(path+"commodities_12_22.csv")
df2 = pd.read_csv(path+"Gold Futures Historical Data.csv")
df3 = pd.read_csv(path+"crude-oil-price.csv")
df4 = pd.read_csv(path+"US Dollar Index (DXY).csv")
df5 = pd.read_csv(path+"NASDAQ_100.csv")
df6 = pd.read_csv(path+"all_commodities_data.csv")
```

Dataset Cleaning

```python
df6 = df6[['date','high','commodity']]

df6 = df6.pivot(index='date', columns='commodity', values='high')

df6.reset_index(inplace=True)

df6.columns
```

```
Index(['date', 'Copper', 'Gold', 'Palladium', 'Platinum', 'Silver'],
dtype='object', name='commodity')
```

```python
df6 = df6.rename_axis(None, axis=1)

df6 = df6.dropna()

df6 = df6.rename(columns={'date': 'Date'})

df6.Date = pd.to_datetime(df6.Date)

df6 = df6.drop('Gold',axis =1)

df5 = df5[['date','high']]

df5.columns = ['Date','NASDAQ_100']

df5.Date = pd.to_datetime(df5.Date)

df4.Date = pd.to_datetime(df4.Date)

df4 = df4.dropna()

df4.reset_index(drop=True, inplace=True)

df4.Date.unique()
```

```
<DatetimeArray>
['1971-01-04 00:00:00', '1971-01-05 00:00:00', '1971-01-06 00:00:00',
 '1971-01-07 00:00:00', '1971-01-08 00:00:00', '1971-01-11 00:00:00',
 '1971-01-12 00:00:00', '1971-01-13 00:00:00', '1971-01-14 00:00:00',
 '1971-01-15 00:00:00',
 ...
 '2024-03-22 00:00:00', '2024-03-25 00:00:00', '2024-03-26 00:00:00',
 '2024-03-27 00:00:00', '2024-03-28 00:00:00', '2024-04-01 00:00:00',
 '2024-04-02 00:00:00', '2024-04-03 00:00:00', '2024-04-04 00:00:00',
 '2024-04-05 00:00:00']
Length: 13529, dtype: datetime64[ns]
```

```python
df4.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13529 entries, 0 to 13528
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       13529 non-null  datetime64[ns]
 1   Open       13529 non-null  float64
 2   High       13529 non-null  float64
 3   Low        13529 non-null  float64
 4   Close      13529 non-null  float64
 5   Adj Close  13529 non-null  float64
 6   Volume     13529 non-null  float64
dtypes: datetime64[ns](1), float64(6)
memory usage: 740.0 KB
```

```
df4 = df4[['Date','Open']]

df4.columns = 'Date', 'USD(DXY)'

df3.columns

Index(['date', 'price', 'percentChange', 'change'], dtype='object')

df3.isna().sum()

date             0
price            0
percentChange    1
change           1
dtype: int64

df3.columns = ['Date', 'Crude_Oil_Price','PercentChange','Change']

df3.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 496 entries, 0 to 495
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Date             496 non-null    object
 1   Crude_Oil_Price  496 non-null    float64
 2   PercentChange    495 non-null    float64
 3   Change           495 non-null    float64
dtypes: float64(3), object(1)
memory usage: 15.6+ KB

df3.Date = pd.to_datetime(df3.Date)

df3.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 496 entries, 0 to 495
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Date             496 non-null    datetime64[ns, UTC]
 1   Crude_Oil_Price  496 non-null    float64
 2   PercentChange    495 non-null    float64
 3   Change           495 non-null    float64
dtypes: datetime64[ns, UTC](1), float64(3)
memory usage: 15.6 KB

df3 = df3.drop(['PercentChange','Change'], axis = 1)

df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Date      5000 non-null   object
 1   Price     5000 non-null   object
 2   Open      5000 non-null   object
 3   High      5000 non-null   object
 4   Low       5000 non-null   object
 5   Vol.      4990 non-null   object
 6   Change %  5000 non-null   object
dtypes: object(7)
memory usage: 273.6+ KB
```

```
df2 = df2.dropna()
```

```
df2.reset_index(drop=True, inplace=True)
```

```
df2.Date = pd.to_datetime(df2.Date)
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_24748\2397979236.py:1:
UserWarning: Parsing dates in %d-%m-%Y format when dayfirst=False (the
default) was specified. Pass `dayfirst=True` or specify a format to
silence this warning.
  df2.Date = pd.to_datetime(df2.Date)
```

```
df2.Price = df2.Price.str.replace(',','').astype(float)
```

```
df2['Vol.'] = df2['Vol.'].str.replace('K','').astype(float)
```

```
df2['Vol.'] = df2['Vol.'] *1000
```

```
df2 = df2.drop(['Open', 'High', 'Low', 'Change %'], axis = 1)
```

```
df2.columns = ['Date','Gold_Price','Vol.']
```

```
df1.dropna(inplace=True)
```

```
df1.reset_index(drop=True,inplace=True)
```

```
df1.Date = pd.to_datetime(df1.Date)
```

```
df.dropna(inplace=True)
```

```
df.reset_index(drop=True,inplace=True)
```

```
df = df.drop('Unnamed: 0', axis=1)
```

```
df.Date = pd.to_datetime(df.Date)
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_24748\4238552302.py:1:
UserWarning: Parsing dates in %d-%m-%Y format when dayfirst=False (the
```

```
default) was specified. Pass `dayfirst=True` or specify a format to
silence this warning.
  df.Date = pd.to_datetime(df.Date)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 609 entries, 0 to 608
Data columns (total 38 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Date               609 non-null    datetime64[ns]
 1   Natural_Gas_Price  609 non-null    float64
 2   Natural_Gas_Vol.   609 non-null    float64
 3   Crude_oil_Price    609 non-null    float64
 4   Crude_oil_Vol.     609 non-null    float64
 5   Copper_Price       609 non-null    float64
 6   Copper_Vol.        609 non-null    float64
 7   Bitcoin_Price      609 non-null    object
 8   Bitcoin_Vol.       609 non-null    float64
 9   Platinum_Price     609 non-null    object
 10  Platinum_Vol.      609 non-null    float64
 11  Ethereum_Price     609 non-null    object
 12  Ethereum_Vol.      609 non-null    float64
 13  S&P_500_Price      609 non-null    object
 14  Nasdaq_100_Price   609 non-null    object
 15  Nasdaq_100_Vol.    609 non-null    float64
 16  Apple_Price        609 non-null    float64
 17  Apple_Vol.         609 non-null    float64
 18  Tesla_Price        609 non-null    float64
 19  Tesla_Vol.         609 non-null    float64
 20  Microsoft_Price    609 non-null    float64
 21  Microsoft_Vol.     609 non-null    float64
 22  Silver_Price       609 non-null    float64
 23  Silver_Vol.        609 non-null    float64
 24  Google_Price       609 non-null    float64
 25  Google_Vol.        609 non-null    float64
 26  Nvidia_Price       609 non-null    float64
 27  Nvidia_Vol.        609 non-null    float64
 28  Berkshire_Price    609 non-null    object
 29  Berkshire_Vol.     609 non-null    float64
 30  Netflix_Price      609 non-null    float64
 31  Netflix_Vol.       609 non-null    float64
 32  Amazon_Price       609 non-null    float64
 33  Amazon_Vol.        609 non-null    float64
 34  Meta_Price         609 non-null    float64
 35  Meta_Vol.          609 non-null    float64
 36  Gold_Price         609 non-null    object
 37  Gold_Vol.          609 non-null    float64
```

```
dtypes: datetime64[ns](1), float64(30), object(7)
memory usage: 180.9+ KB
```

```python
for column in df.columns:
    try:
       df[column] =  df[column].str.replace(',','').astype(float)
    except:
        continue

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 609 entries, 0 to 608
Data columns (total 38 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Date               609 non-null    datetime64[ns]
 1   Natural_Gas_Price  609 non-null    float64
 2   Natural_Gas_Vol.   609 non-null    float64
 3   Crude_oil_Price    609 non-null    float64
 4   Crude_oil_Vol.     609 non-null    float64
 5   Copper_Price       609 non-null    float64
 6   Copper_Vol.        609 non-null    float64
 7   Bitcoin_Price      609 non-null    float64
 8   Bitcoin_Vol.       609 non-null    float64
 9   Platinum_Price     609 non-null    float64
 10  Platinum_Vol.      609 non-null    float64
 11  Ethereum_Price     609 non-null    float64
 12  Ethereum_Vol.      609 non-null    float64
 13  S&P_500_Price      609 non-null    float64
 14  Nasdaq_100_Price   609 non-null    float64
 15  Nasdaq_100_Vol.    609 non-null    float64
 16  Apple_Price        609 non-null    float64
 17  Apple_Vol.         609 non-null    float64
 18  Tesla_Price        609 non-null    float64
 19  Tesla_Vol.         609 non-null    float64
 20  Microsoft_Price    609 non-null    float64
 21  Microsoft_Vol.     609 non-null    float64
 22  Silver_Price       609 non-null    float64
 23  Silver_Vol.        609 non-null    float64
 24  Google_Price       609 non-null    float64
 25  Google_Vol.        609 non-null    float64
 26  Nvidia_Price       609 non-null    float64
 27  Nvidia_Vol.        609 non-null    float64
 28  Berkshire_Price    609 non-null    float64
 29  Berkshire_Vol.     609 non-null    float64
 30  Netflix_Price      609 non-null    float64
 31  Netflix_Vol.       609 non-null    float64
 32  Amazon_Price       609 non-null    float64
 33  Amazon_Vol.        609 non-null    float64
```

```
 34  Meta_Price         609 non-null    float64
 35  Meta_Vol.          609 non-null    float64
 36  Gold_Price         609 non-null    float64
 37  Gold_Vol.          609 non-null    float64
dtypes: datetime64[ns](1), float64(37)
memory usage: 180.9 KB

df2.columns = ['Date', 'Gold', 'Vol.']

Gold = df1[df1.Date>'2019-11-14']
Gold = Gold[['Date','Gold']]

Gold = pd.concat([Gold,df2])

Gold

           Date    Gold     Vol.
0    2022-06-15  1814.8      NaN
1    2022-06-14  1813.5      NaN
2    2022-06-13  1831.8      NaN
3    2022-06-10  1875.5      NaN
4    2022-06-09  1852.8      NaN
...         ...     ...      ...
4985 2000-01-28   286.0  15190.0
4986 2000-01-27   287.1  14490.0
4987 2000-01-26   286.5  19630.0
4988 2000-01-25   286.6  36120.0
4989 2000-01-24   288.1  32140.0

[5656 rows x 3 columns]

Gold1 = df[['Date','Gold_Price']]

Gold1 = Gold1[Gold1.Date>'2022-06-15']

Gold1.columns = ['Date', 'Gold']

Gold = pd.concat([Gold1, Gold])

Crude_Oil = df3[df3.Date >='2000']

Crude_Oil = Crude_Oil[Crude_Oil.Date<='2023-12-21']

Gold = Gold.reset_index(drop=True)

Crude_Oil = Crude_Oil.reset_index(drop=True)

Crude_Oil['Date'] =
pd.to_datetime(Crude_Oil['Date']).dt.tz_localize(None)
Gold_Crude = pd.merge(Gold, Crude_Oil, on='Date', how='inner')

Gold_Crude = Gold_Crude.drop('Vol.', axis =1)
```

```
Gold_Crude_Dxy = pd.merge(Gold_Crude, df4, on='Date', how='inner')

merged_df = pd.merge(Gold_Crude_Dxy, df6, on='Date', how='inner')

merged_df = pd.merge(merged_df, df5, on='Date', how='inner')
```

Cleaned and Normalized Dataset

- **`merged_df`**: The final cleaned and merged dataset.
- **`normalized_df`**: The scaled dataset used for visualizations.
- **`corr`**: The pairwise correlation matrix.

```
merged_df = merged_df[merged_df['Date']>='2010']

corr = pg.pairwise_corr(merged_df, method='pearson')

corr['p-unc'] = corr['p-unc'].round(5)

corr = corr[corr['p-unc']<=0.05]

date = merged_df['Date']

commodities = merged_df.drop('Date', axis=1)

scaler = MinMaxScaler()

normalized_df = pd.DataFrame(scaler.fit_transform(commodities),
columns=commodities.columns)

normalized_df.insert(0, 'Date', date)

merged_df.head()
```

```
        Date    Gold  Crude_Oil_Price    USD(DXY)  Copper    Palladium  \
0 2023-12-01  2089.7            71.65  103.360001  3.9100  1000.000000
1 2023-09-01  1967.1            88.80  103.620003  3.8620  1217.199951
2 2023-06-01  1995.5            70.78  104.150002  3.7115  1384.000000
3 2023-03-01  1845.4            75.80  105.040001  4.1810  1421.699951
4 2022-09-01  1709.3            78.72  108.839996  3.5180  2074.000000

      Platinum     Silver  NASDAQ_100
0   932.000000  25.565001  16013.7500
1   965.599976  24.840000  15618.8496
2  1018.200012  23.875000  14493.3096
```

```
3   961.500000  21.150000  12054.4805
4   804.000000  17.715000  12290.3301
```

normalized_df.head()

```
         Date      Gold  Crude_Oil_Price  USD(DXY)    Copper  Palladium  \
0 2023-12-01  1.000000         0.555369  0.843070  0.693316   0.232927

1 2023-09-01  0.880530         0.735724  0.850516  0.675342   0.322661

2 2023-06-01  0.908205         0.546219  0.865693  0.618985   0.391572

3 2023-03-01  0.761937         0.599011  0.891180  0.794795   0.407147

4 2022-09-01  0.629312         0.629719  1.000000  0.546527   0.676637


   Platinum    Silver  NASDAQ_100
0  0.185765  0.414123    0.971836
1  0.215473  0.387936    0.944936
2  0.261981  0.353079    0.868264
3  0.211848  0.254651    0.702130
4  0.072591  0.130576    0.718196
```

corr

```
                  X                Y   method alternative    n
r  \
0              Gold  Crude_Oil_Price  pearson   two-sided  103
0.243864
2              Gold           Copper  pearson   two-sided  103
0.622621
3              Gold        Palladium  pearson   two-sided  103
0.663496
5              Gold           Silver  pearson   two-sided  103
0.580323
6              Gold       NASDAQ_100  pearson   two-sided  103
0.651902
7   Crude_Oil_Price         USD(DXY)  pearson   two-sided  103 -
0.622805
8   Crude_Oil_Price           Copper  pearson   two-sided  103
0.708096
10  Crude_Oil_Price         Platinum  pearson   two-sided  103
0.705804
11  Crude_Oil_Price           Silver  pearson   two-sided  103
0.603659
13         USD(DXY)           Copper  pearson   two-sided  103 -
0.385675
14         USD(DXY)        Palladium  pearson   two-sided  103
0.433714
```

```
15          USD(DXY)         Platinum  pearson   two-sided  103 -
0.917850
16          USD(DXY)           Silver  pearson   two-sided  103 -
0.645369
17          USD(DXY)       NASDAQ_100  pearson   two-sided  103
0.639968
18            Copper        Palladium  pearson   two-sided  103
0.355178
19            Copper         Platinum  pearson   two-sided  103
0.490728
20            Copper           Silver  pearson   two-sided  103
0.716601
21            Copper       NASDAQ_100  pearson   two-sided  103
0.309125
22         Palladium         Platinum  pearson   two-sided  103 -
0.452054
24         Palladium       NASDAQ_100  pearson   two-sided  103
0.866976
25          Platinum           Silver  pearson   two-sided  103
0.733160
26          Platinum       NASDAQ_100  pearson   two-sided  103 -
0.609832

             CI95%    p-unc       BF10     power
0     [0.05, 0.42]  0.01305      2.569  0.705217
2     [0.49, 0.73]  0.00000  4.382e+09  1.000000
3     [0.54, 0.76]  0.00000  3.735e+11  1.000000
5      [0.44, 0.7]  0.00000  8.409e+07  0.999999
6     [0.52, 0.75]  0.00000  9.867e+10  1.000000
7   [-0.73, -0.49]  0.00000  4.464e+09  1.000000
8      [0.6, 0.79]  0.00000  1.152e+14  1.000000
10    [0.59, 0.79]  0.00000  8.356e+13  1.000000
11    [0.46, 0.71]  0.00000  6.919e+08  1.000000
13  [-0.54, -0.21]  0.00006    359.271  0.983195
14    [0.26, 0.58]  0.00000   3749.229  0.996581
15  [-0.94, -0.88]  0.00000   6.83e+38  1.000000
16  [-0.75, -0.52]  0.00000   4.78e+10  1.000000
17    [0.51, 0.74]  0.00000  2.661e+10  1.000000
18    [0.17, 0.51]  0.00023     98.142  0.961611
19    [0.33, 0.62]  0.00000  1.042e+05  0.999701
20     [0.61, 0.8]  0.00000  3.895e+14  1.000000
21    [0.12, 0.47]  0.00149     17.846  0.894285
22  [-0.59, -0.28]  0.00000   1.02e+04  0.998325
24    [0.81, 0.91]  0.00000   1.12e+29  1.000000
25    [0.63, 0.81]  0.00000  4.774e+15  1.000000
26  [-0.72, -0.47]  0.00000  1.245e+09  1.000000

#merged_df.to_excel(r'C:\Files\College_Files\
time_series_commodities.xlsx', index=False)
```

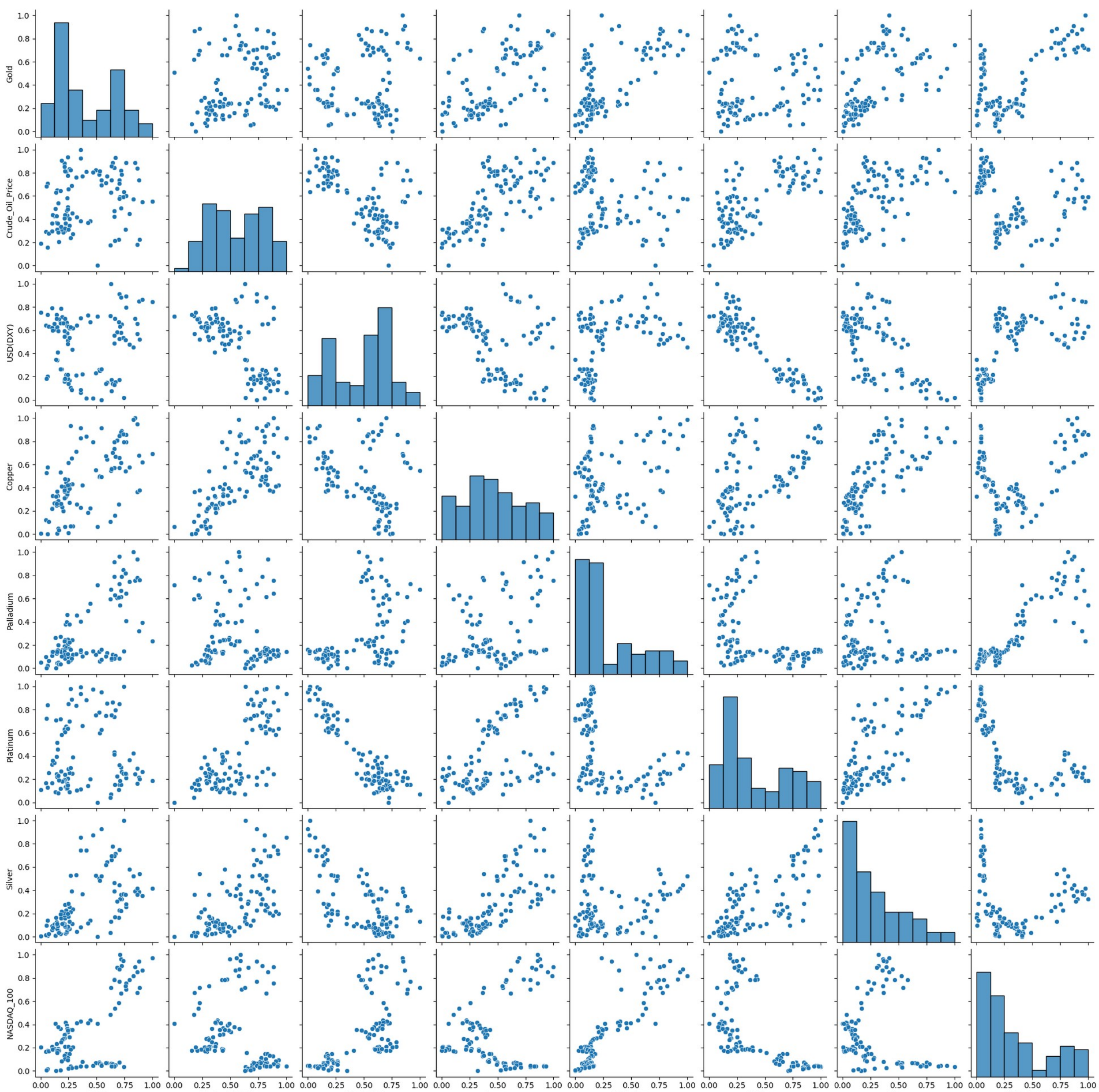
```
#normalized_df.to_excel(r'C:\Files\College_Files\
normalized_commodities.xlsx', index=False)
```

Visualizations

Initial Pair Plot

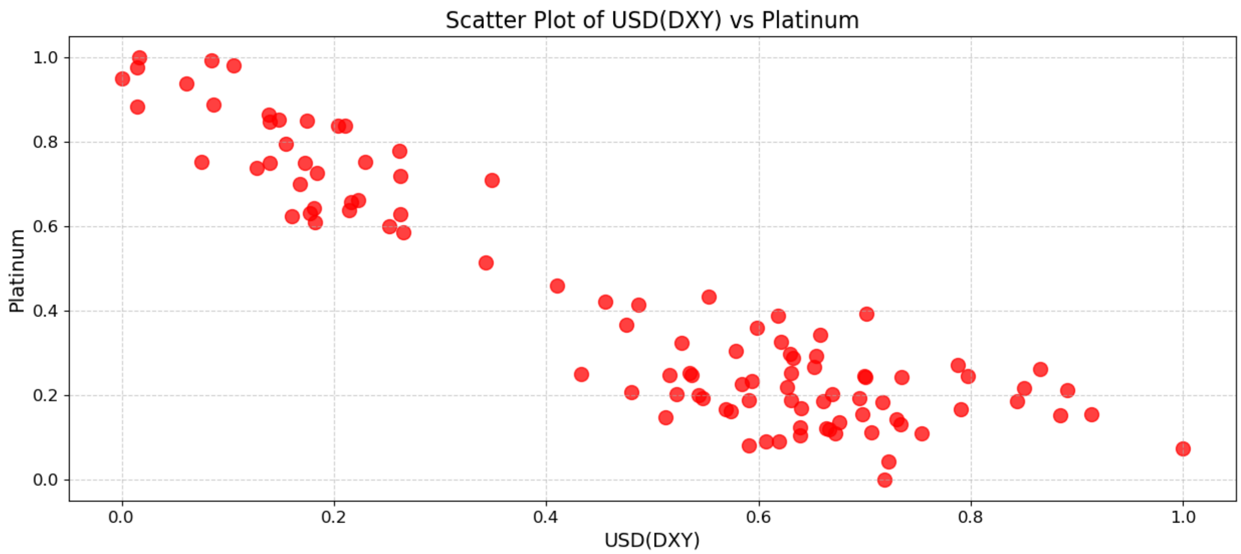
```
sns.pairplot(normalized_df)
```

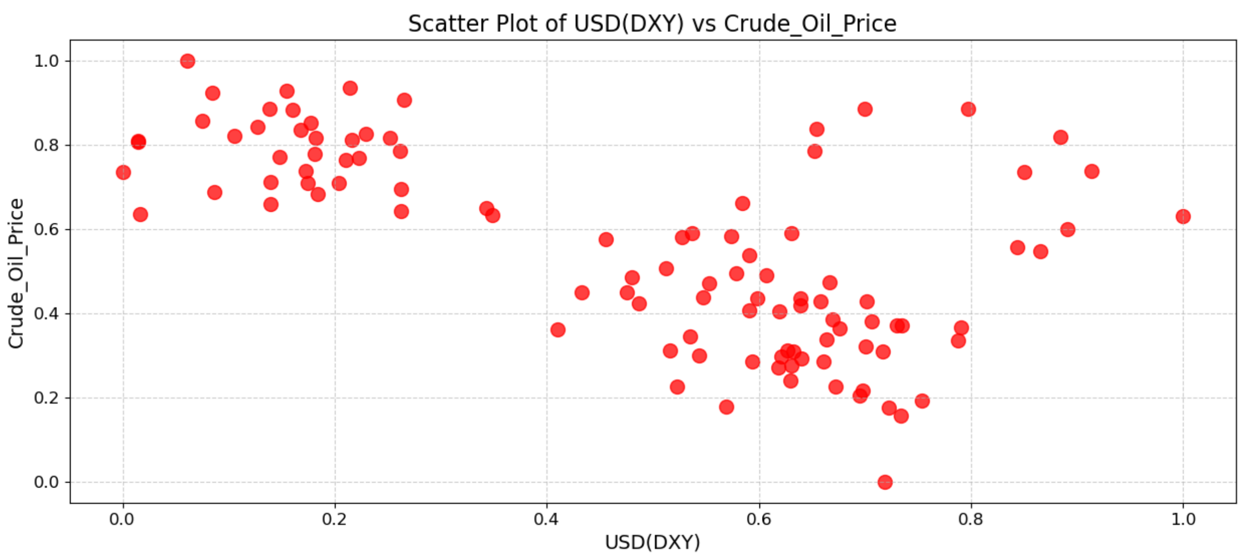
```
<seaborn.axisgrid.PairGrid at 0x224f8e97f80>
```

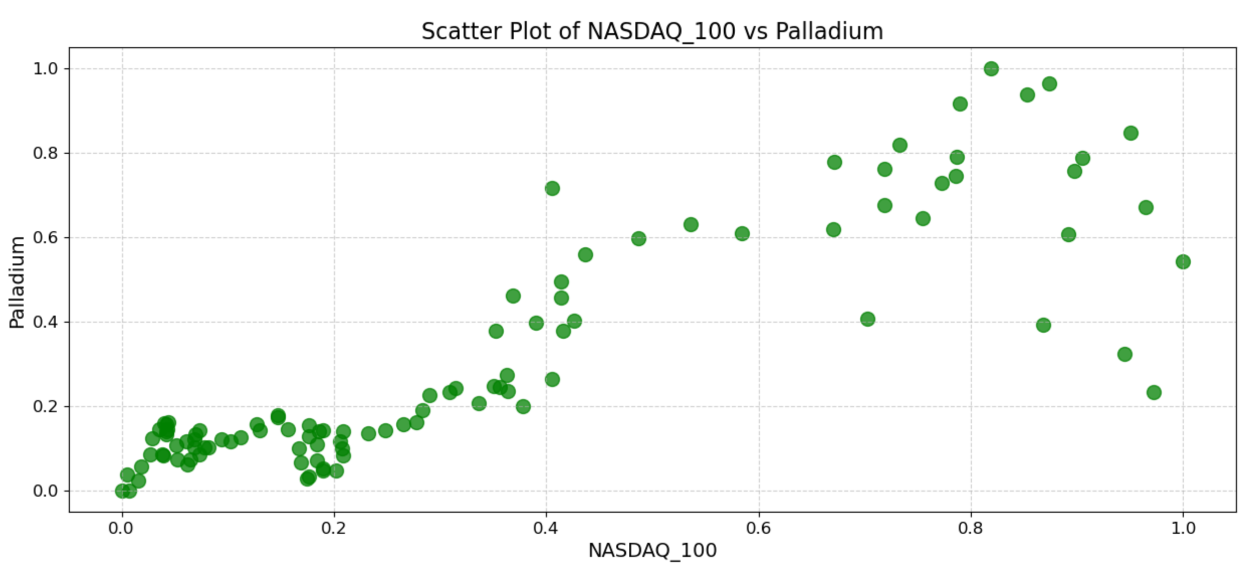


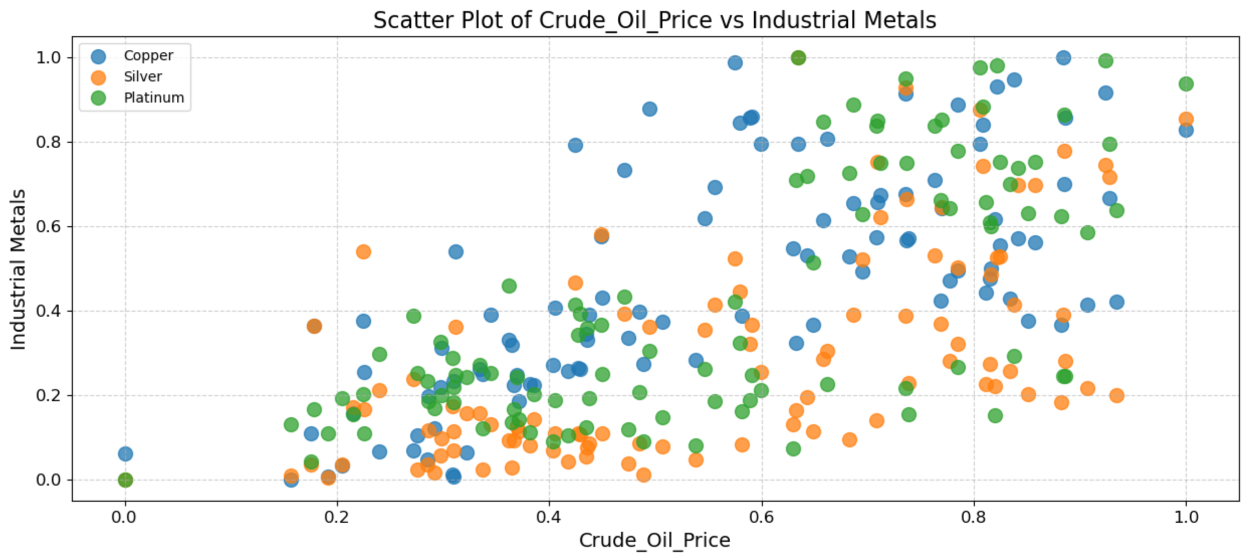
Correlation Scatter Plots

```
def corr_scatter(x,y,color,combined_name=None):

    if combined_name is None:
        combined_name = y

    if len(y) > 1:

            plt.figure(figsize=(15,6))

            for i in range(len(y)):
                color=None
                plt.scatter(normalized_df[x], normalized_df[y[i]],
marker='o', s=100, alpha=0.75,label=y[i])

            plt.grid(True, linestyle='--', alpha=0.6)

            plt.xlabel(x, fontsize=14)
            plt.ylabel(combined_name, fontsize=14)
            plt.title(f'Scatter Plot of {x} vs {combined_name}',
fontsize=16)


            plt.xticks(fontsize=12)
            plt.yticks(fontsize=12)
            plt.legend()
            plt.show()
    else:

        plt.figure(figsize=(15,6))
        plt.scatter(normalized_df[x], normalized_df[y], color=color,
marker='o', s=100, alpha=0.75)

        plt.grid(True, linestyle='--', alpha=0.6)

        plt.xlabel(x, fontsize=14)
        plt.ylabel(y[0], fontsize=14)
        plt.title(f'Scatter Plot of {x} vs {y[0]}', fontsize=16)

        plt.xticks(fontsize=12)
        plt.yticks(fontsize=12)

        plt.show()

corr_scatter('USD(DXY)',['Platinum'],'red')
corr_scatter('USD(DXY)',['Crude_Oil_Price'],'red')
corr_scatter('NASDAQ_100',['Palladium'],'green')
corr_scatter('Crude_Oil_Price',
['Copper','Silver','Platinum'],'','Industrial Metals')
```


Scatter Plot of USD(DXY) vs Platinum
Platinum
USD(DXY)
0.0
0.2
0.4
0.6
0.8
1.0

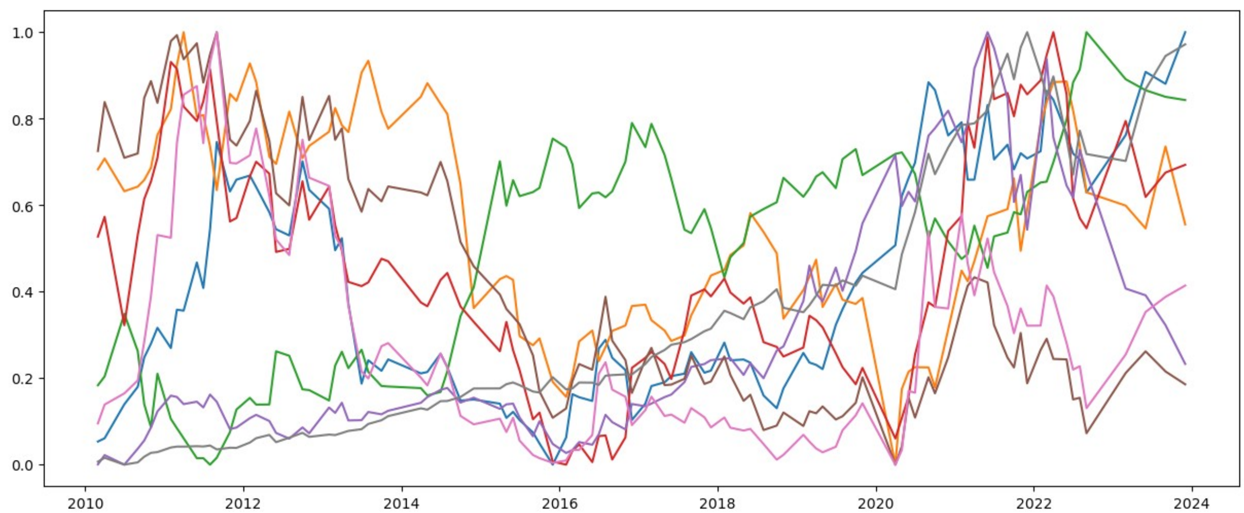

Scatter Plot of USD(DXY) vs Crude_Oil_Price
Crude_Oil_Price
USD(DXY)
0.0
0.2
0.4
0.6
0.8
1.0


Scatter Plot of NASDAQ_100 vs Palladium
Palladium
NASDAQ_100
0.0
0.2
0.4
0.6
0.8
1.0



Line Plots All Columns

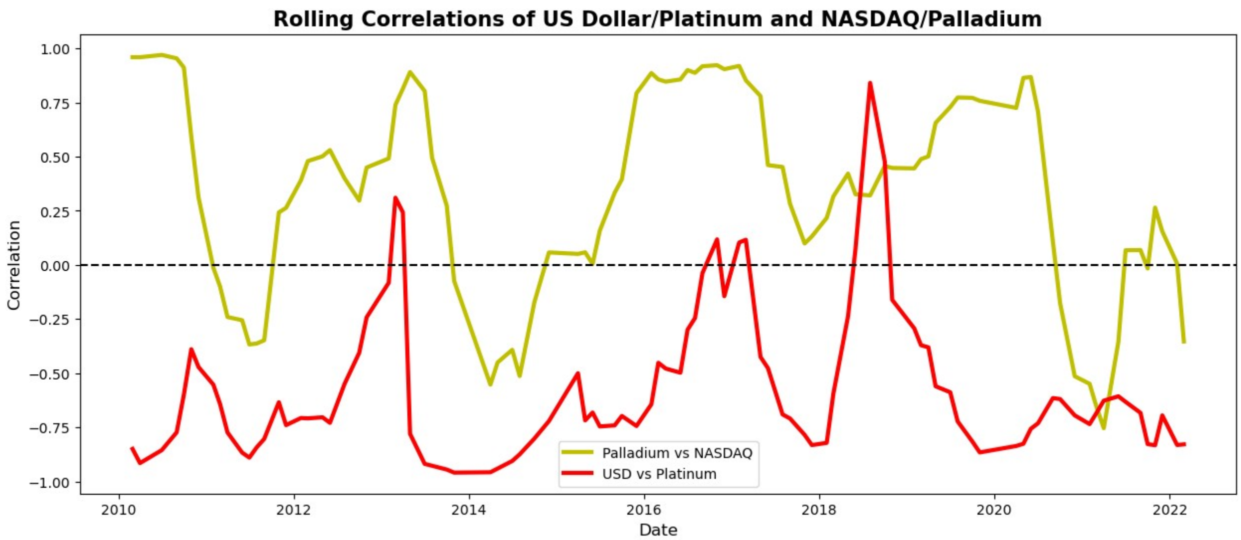
```
plt.figure(figsize=(15,6))
for commodity in commodities.columns:
    plt.plot(normalized_df.Date,normalized_df[commodity])
```



Rolling Correlations

```
rolling_corr_PallNasd =
merged_df['Palladium'].rolling(window=10).corr(merged_df['NASDAQ_100']
)
rolling_corr_USDPLA =
merged_df['USD(DXY)'].rolling(window=10).corr(merged_df['Platinum'])

plt.figure(figsize=(15, 6))
```

```
plt.plot(merged_df.Date, rolling_corr_PallNasd, label='Palladium vs
NASDAQ', color='y', linewidth=3)
plt.plot(merged_df.Date, rolling_corr_USDPLA, label='USD vs Platinum',
color='r', linewidth=3)

plt.axhline(y=0, color='black', linestyle='--')

plt.xlabel('Date', fontsize=12)
plt.ylabel('Correlation', fontsize=12)

plt.title('Rolling Correlations of US Dollar/Platinum and
NASDAQ/Palladium', fontsize=15, fontweight='bold')

plt.legend()
plt.show()
```



Unfiltered Correlation Heatmaps (p-value may be >0.05 for some values)

```
plt.figure(figsize=(12, 8))

sns.heatmap(
    commodities.corr(),
    annot=True,
    fmt='.2f',
    linewidths=2,
    cmap='coolwarm',
    square=True
)

plt.title("Heatmap of Unfiltered Correlations", fontsize=16,
fontweight='bold')
```

```python
plt.xticks(rotation=45, fontsize=12, fontweight='bold')
plt.yticks(rotation=0, fontsize=12, fontweight='bold')

plt.show()
```