2019-2024-market-data-analysis

March 29, 2025

1 Analysis Project

The goal here is to exercise the use of data cleaning and data visualization techniques for the use of data analytics.

The project focuses on US stock market data.

The steps taken in this exercise: 1. Review Metadata 2. Import Kaggle Dataset 3. Data Exploration & Cleaning 4. Organize Data Into Groups 5. Stock Visualizations 6. Market Visualizations 7. Commodity Visualizations 8. Combined Analysis

1.0.1 1) Reviewing Metadata

It is critical to gain a contextual understanding before touching the data by reviewing the metadata.

- **Date**: The date of the recorded data, formatted as DD-MM-YYYY.
- **Natural_Gas_Price**: Price of natural gas in USD per million British thermal units (MMBtu).
- **Natural_Gas_Vol.**: Trading volume of natural gas.
- **Crude_oil_Price**: Price of crude oil in USD per barrel.
- **Crude_oil_Vol.**: Trading volume of crude oil.
- **Copper_Price**: Price of copper in USD per pound.
- **Copper_Vol.**: Trading volume of copper.
- **Bitcoin_Price**: Price of Bitcoin in USD.
- **Bitcoin_Vol.**: Trading volume of Bitcoin.
- **Platinum_Price**: Price of platinum in USD per troy ounce.
- **Platinum_Vol.**: Trading volume of platinum.
- **Ethereum_Price**: Price of Ethereum in USD.
- **Ethereum_Vol.**: Trading volume of Ethereum.
- **S&P_500_Price**: Price index of the S&P 500.
- **Nasdaq_100_Price**: Price index of the Nasdaq 100.
- **Nasdaq_100_Vol.**: Trading volume for the Nasdaq 100 index.
- **Apple_Price**: Stock price of Apple Inc. in USD.
- **Apple_Vol.**: Trading volume of Apple Inc. stock.
- **Tesla_Price**: Stock price of Tesla Inc. in USD.
- **Tesla_Vol.**: Trading volume of Tesla Inc. stock.
- **Microsoft_Price**: Stock price of Microsoft Corporation in USD.
- **Microsoft_Vol.**: Trading volume of Microsoft Corporation stock.
- **Silver_Price**: Price of silver in USD per troy ounce.
- **Silver_Vol.**: Trading volume of silver.

- **Google_Price**: Stock price of Alphabet Inc. (Google) in USD.
- **Google_Vol.**: Trading volume of Alphabet Inc. stock.
- **Nvidia_Price**: Stock price of Nvidia Corporation in USD.
- **Nvidia_Vol.**: Trading volume of Nvidia Corporation stock.
- **Berkshire_Price**: Stock price of Berkshire Hathaway Inc. in USD.
- **Berkshire_Vol.**: Trading volume of Berkshire Hathaway Inc. stock.
- **Netflix_Price**: Stock price of Netflix Inc. in USD.
- **Netflix_Vol.**: Trading volume of Netflix Inc. stock.
- **Amazon_Price**: Stock price of Amazon.com Inc. in USD.
- **Amazon_Vol.**: Trading volume of Amazon.com Inc. stock.
- **Meta_Price**: Stock price of Meta Platforms, Inc. (formerly Facebook) in USD.
- **Meta_Vol.**: Trading volume of Meta Platforms, Inc. stock.
- **Gold_Price**: Price of gold in USD per troy ounce.
- **Gold_Vol.**: Trading volume of gold.

Reference: https://www.kaggle.com/datasets/saketk511/2019-2024-us-stock-market-data.

1.0.2 2) Importing Dataset

Gather the stock information data from a source in Kaggle. The file is taken from a directory and converted into a dataframe on the notebook, where it'll be analyzed.

[1]:
```
import kagglehub

# Download latest version
data_path = kagglehub.dataset_download("saketk511/
 ↪2019-2024-us-stock-market-data")

print("Path to dataset files:", data_path)
```

```
Downloading from
https://www.kaggle.com/api/v1/datasets/download/saketk511/2019-2024-us-stock-
market-data?dataset_version_number=1…

100%|                          | 155k/155k [00:00<00:00, 1.00MB/s]

Extracting files…
Path to dataset files:
/Users/chevalier/.cache/kagglehub/datasets/saketk511/2019-2024-us-stock-market-
data/versions/1
```

[2]:
```
import os

files = os.listdir(data_path)
print(files)
```

```
['Stock Market Dataset.csv']
```

[3]:
```python
file_path = os.path.join(data_path, files[0])
print(file_path)
```

```
/Users/chevalier/.cache/kagglehub/datasets/saketk511/2019-2024-us-stock-market-
data/versions/1/Stock Market Dataset.csv
```

[4]:
```python
import pandas as pd

df = pd.read_csv(file_path, index_col=0)
df.head(5)
```

[4]:
```
         Date  Natural_Gas_Price  Natural_Gas_Vol.  Crude_oil_Price  \
0  02-02-2024              2.079               NaN            72.28
1  01-02-2024              2.050          161340.0            73.82
2  31-01-2024              2.100          142860.0            75.85
3  30-01-2024              2.077          139750.0            77.82
4  29-01-2024              2.490            3590.0            76.78

   Crude_oil_Vol.  Copper_Price  Copper_Vol. Bitcoin_Price  Bitcoin_Vol.  \
0             NaN        3.8215          NaN     43,194.70       42650.0
1        577940.0        3.8535          NaN     43,081.40       47690.0
2        344490.0        3.9060          NaN     42,580.50       56480.0
3        347240.0        3.9110          NaN     42,946.20       55130.0
4        331930.0        3.8790          NaN     43,299.80       45230.0

  Platinum_Price  …  Berkshire_Price Berkshire_Vol.  Netflix_Price  \
0          901.6  …         5,89,498        10580.0         564.64
1          922.3  …         5,81,600         9780.0         567.51
2          932.6  …         5,78,020         9720.0         564.11
3          931.7  …         5,84,680         9750.0         562.85
4          938.3  …         5,78,800        13850.0         575.79

  Netflix_Vol. Amazon_Price  Amazon_Vol.  Meta_Price    Meta_Vol. Gold_Price  \
0    4030000.0       171.81  117220000.0      474.99   84710000.0   2,053.70
1    3150000.0       159.28   66360000.0      394.78   25140000.0   2,071.10
2    4830000.0       155.20   49690000.0      390.14   20010000.0   2,067.40
3    6120000.0       159.00   42290000.0      400.06   18610000.0   2,050.90
4    6880000.0       161.26   42840000.0      401.02   17790000.0   2,034.90

   Gold_Vol.
0        NaN
1   260920.0
2   238370.0
3   214590.0
4     1780.0

[5 rows x 38 columns]
```

1.0.3 3) Data Exploration & Cleaning

Understand the data and prepare it for future analysis.

[5]:
```
df.shape
```

[5]: (1243, 38)

Inspect how many days were not captured for each of the columns.

[6]:
```
nulls=df.isnull().sum().to_frame()
nulls.rename(columns={0:'Nulls'},inplace=True)
nulls
```

[6]:

	Nulls
Date	0
Natural_Gas_Price	0
Natural_Gas_Vol.	4
Crude_oil_Price	0
Crude_oil_Vol.	23
Copper_Price	0
Copper_Vol.	37
Bitcoin_Price	0
Bitcoin_Vol.	0
Platinum_Price	0
Platinum_Vol.	607
Ethereum_Price	0
Ethereum_Vol.	0
S&P_500_Price	0
Nasdaq_100_Price	0
Nasdaq_100_Vol.	1
Apple_Price	0
Apple_Vol.	0
Tesla_Price	0
Tesla_Vol.	0
Microsoft_Price	0
Microsoft_Vol.	0
Silver_Price	0
Silver_Vol.	47
Google_Price	0
Google_Vol.	0
Nvidia_Price	0
Nvidia_Vol.	0
Berkshire_Price	0
Berkshire_Vol.	0
Netflix_Price	0
Netflix_Vol.	0
Amazon_Price	0
Amazon_Vol.	0

```
Meta_Price          0
Meta_Vol.           0
Gold_Price          0
Gold_Vol.           2
```

[7]:

```
columns_to_fill = ['Natural_Gas_Vol.','Crude_oil_Vol.','Copper_Vol.
 ↪','Platinum_Vol.','Nasdaq_100_Vol.','Silver_Vol.','Gold_Vol.']

for column in columns_to_fill:
    df[column] = df[column].ffill()
    df[column] = df[column].bfill()
```

[8]:

```
nulls=df.isnull().sum().to_frame()
nulls.rename(columns={0:'Nulls'},inplace=True)
nulls
```

[8]:

	Nulls
Date	0
Natural_Gas_Price	0
Natural_Gas_Vol.	0
Crude_oil_Price	0
Crude_oil_Vol.	0
Copper_Price	0
Copper_Vol.	0
Bitcoin_Price	0
Bitcoin_Vol.	0
Platinum_Price	0
Platinum_Vol.	0
Ethereum_Price	0
Ethereum_Vol.	0
S&P_500_Price	0
Nasdaq_100_Price	0
Nasdaq_100_Vol.	0
Apple_Price	0
Apple_Vol.	0
Tesla_Price	0
Tesla_Vol.	0
Microsoft_Price	0
Microsoft_Vol.	0
Silver_Price	0
Silver_Vol.	0
Google_Price	0
Google_Vol.	0
Nvidia_Price	0
Nvidia_Vol.	0
Berkshire_Price	0
Berkshire_Vol.	0

```
Netflix_Price          0
Netflix_Vol.           0
Amazon_Price           0
Amazon_Vol.            0
Meta_Price             0
Meta_Vol.              0
Gold_Price             0
Gold_Vol.              0
```

Verify that there are no duplicated days, which could potentially hinder analysis.

[9]:
```
df[df.duplicated()]
```

[9]:
```
Empty DataFrame
Columns: [Date, Natural_Gas_Price, Natural_Gas_Vol., Crude_oil_Price,
Crude_oil_Vol., Copper_Price, Copper_Vol., Bitcoin_Price, Bitcoin_Vol.,
Platinum_Price, Platinum_Vol., Ethereum_Price, Ethereum_Vol., S&P_500_Price,
Nasdaq_100_Price, Nasdaq_100_Vol., Apple_Price, Apple_Vol., Tesla_Price,
Tesla_Vol., Microsoft_Price, Microsoft_Vol., Silver_Price, Silver_Vol.,
Google_Price, Google_Vol., Nvidia_Price, Nvidia_Vol., Berkshire_Price,
Berkshire_Vol., Netflix_Price, Netflix_Vol., Amazon_Price, Amazon_Vol.,
Meta_Price, Meta_Vol., Gold_Price, Gold_Vol.]
Index: []

[0 rows x 38 columns]
```

Change the data types of date and the other columns to be datetime and float formats.

[10]:
```
df['Date']=pd.to_datetime(df.Date, format='%d-%m-%Y')

for col in df.columns[1:]:
    df[col] = df[col].replace(',', '', regex=True).astype('float64')
```

Verify the data types are numerical for all columns excluding the date.

[11]:
```
types=df.dtypes.to_frame()
types.rename(columns={0:'Types'},inplace=True)
types
```

[11]:
```
                              Types
Date                 datetime64[ns]
Natural_Gas_Price           float64
Natural_Gas_Vol.            float64
Crude_oil_Price             float64
Crude_oil_Vol.              float64
Copper_Price                float64
Copper_Vol.                 float64
Bitcoin_Price               float64
Bitcoin_Vol.                float64
```

```
Platinum_Price            float64
Platinum_Vol.             float64
Ethereum_Price            float64
Ethereum_Vol.             float64
S&P_500_Price             float64
Nasdaq_100_Price          float64
Nasdaq_100_Vol.           float64
Apple_Price               float64
Apple_Vol.                float64
Tesla_Price               float64
Tesla_Vol.                float64
Microsoft_Price           float64
Microsoft_Vol.            float64
Silver_Price              float64
Silver_Vol.               float64
Google_Price              float64
Google_Vol.               float64
Nvidia_Price              float64
Nvidia_Vol.               float64
Berkshire_Price           float64
Berkshire_Vol.            float64
Netflix_Price             float64
Netflix_Vol.              float64
Amazon_Price              float64
Amazon_Vol.               float64
Meta_Price                float64
Meta_Vol.                 float64
Gold_Price                float64
Gold_Vol.                 float64
```

Inspect some descriptive statistics of the dataset.

[12]:
```
df.describe()
```

[12]:
```
                                Date  Natural_Gas_Price  Natural_Gas_Vol.  \
count                           1243        1243.000000       1243.000000
mean   2021-08-02 10:03:34.320193024           3.494714     131572.510056
min              2019-02-04 00:00:00           1.482000       1200.000000
25%              2020-04-28 12:00:00           2.347500      91700.000000
50%              2021-08-03 00:00:00           2.702000     127370.000000
75%              2022-11-05 12:00:00           4.055500     169340.000000
max              2024-02-02 00:00:00           9.647000     381970.000000
std                              NaN           1.822540      64320.925408

       Crude_oil_Price  Crude_oil_Vol.  Copper_Price   Copper_Vol.  \
count      1243.000000    1.243000e+03   1243.000000   1243.000000
mean         67.577064    3.944626e+05      3.541957  35668.270314
min         -37.630000    1.702000e+04      2.100500     10.000000
```

```
25%        55.095000   2.774400e+05     2.858750     380.000000
50%        69.230000   3.647200e+05     3.666000   21530.000000
75%        80.455000   5.037300e+05     4.137250   68030.000000
max       123.700000   1.770000e+06     4.937500  176040.000000
std        20.465500   2.169229e+05     0.702819   38006.113950

       Bitcoin_Price  Bitcoin_Vol.  Platinum_Price  …  Berkshire_Price  \
count    1243.000000  1.243000e+03     1243.000000  …      1243.000000
mean    25241.903057  4.033918e+07      959.003620  …    404273.051488
min      3397.700000  2.600000e+02      595.200000  …    240000.000000
25%     10014.600000  7.907500e+04      889.775000  …    318984.500000
50%     23055.100000  2.153100e+05      944.700000  …    418349.000000
75%     37784.200000  6.151050e+05     1020.400000  …    471500.000000
max     67527.900000  4.470000e+09     1297.100000  …    589498.000000
std     16029.009055  2.940889e+08      108.012849  …     86369.903899

       Berkshire_Vol.  Netflix_Price  Netflix_Vol.  Amazon_Price  \
count     1243.000000    1243.000000  1.243000e+03   1243.000000
mean      2426.524537     404.839541  7.057401e+06    128.683234
min         80.000000     166.370000  1.140000e+06     79.410000
25%        345.000000     323.010000  3.990000e+06     96.260000
50%       1510.000000     384.150000  5.610000e+06    128.730000
75%       3225.000000     495.365000  7.910000e+06    158.110000
max      13850.000000     691.690000  1.333900e+08    186.570000
std       2660.497572     114.989473  6.384187e+06     30.808631

        Amazon_Vol.   Meta_Price     Meta_Vol.   Gold_Price      Gold_Vol.
count  1.243000e+03  1243.000000  1.243000e+03  1243.000000    1243.000000
mean   7.413005e+07   239.728134  2.325851e+07  1759.246742  210998.238134
min    1.763000e+07    88.910000  5.470000e+06  1272.000000       0.000000
25%    5.264500e+07   183.355000  1.478500e+07  1669.600000  152060.000000
50%    6.520000e+07   224.430000  1.934000e+07  1804.200000  197970.000000
75%    8.674500e+07   301.650000  2.711500e+07  1912.800000  258370.000000
max    3.113500e+08   474.990000  2.304100e+08  2089.700000  813410.000000
std    3.245753e+07    71.015427  1.555486e+07   203.258901  115077.705917

[8 rows x 38 columns]
```

Key Learnings The timeframe captured by the dataset is spans between Febraury 2019 and February 2024.

There are 1243 days of information gathered. For each day, we have prices and volumes of commodities, crypto, markets, and stocks.

1.0.4 4) Organize Data Into Groups

Separate the data so it is comparable to other similar columns.

[13]:
```
stock_prices = df[['Apple_Price', 'Tesla_Price', 'Microsoft_Price',␣
↪'Google_Price', 'Nvidia_Price', 'Berkshire_Price',
                    'Netflix_Price', 'Amazon_Price', 'Meta_Price']]

market_prices = df[['S&P_500_Price', 'Nasdaq_100_Price']]

commodity_prices = df[['Natural_Gas_Price', 'Crude_oil_Price', 'Copper_Price',␣
↪'Platinum_Price', 'Silver_Price', 'Gold_Price']]
```

1.0.5 5) Stock Visualizations

[14]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

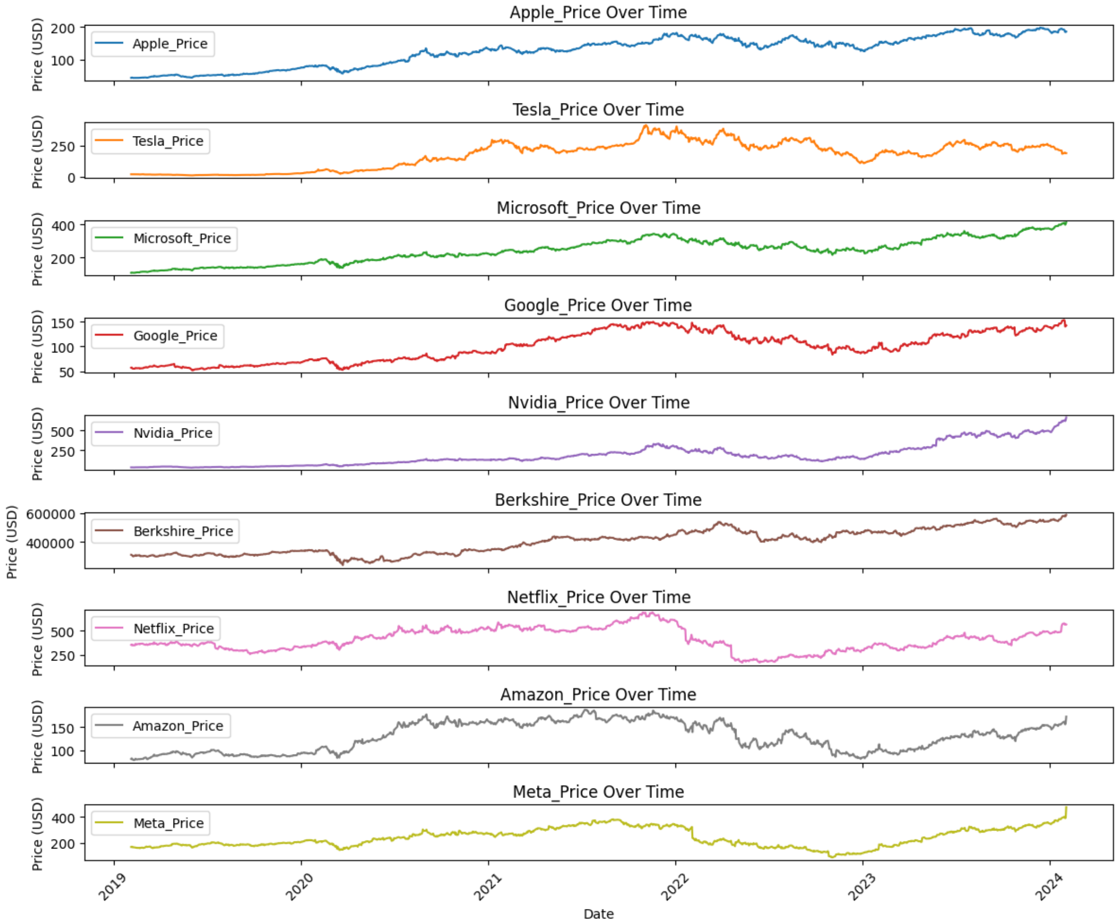
Line Plots:

Analyzing the movements of key stocks over time.

Companies with the highest growth since 2019 include Nvidia and Microsoft.

Netflix grew the least of the bunch.

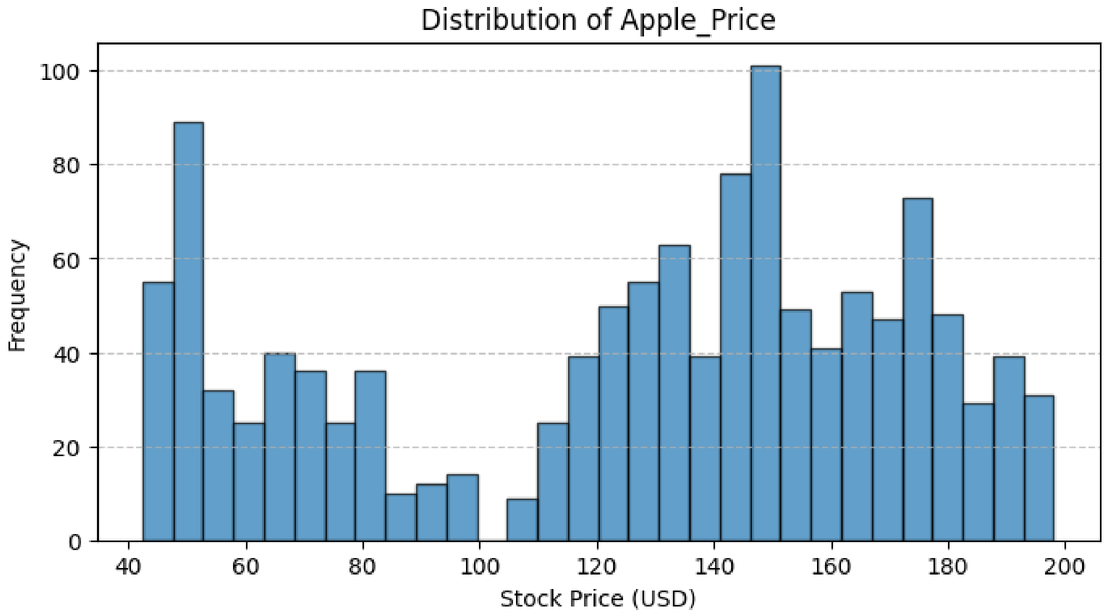
[15]:
```
colors = sns.color_palette("tab10", len(stock_prices.columns))

fig, axes = plt.subplots(nrows=len(stock_prices.columns), ncols=1,␣
↪figsize=(12,10), sharex=True)

for i, (stock,color) in enumerate(zip(stock_prices.columns,colors)):
    axes[i].plot(df['Date'], df[stock], label=stock, color=color)
    axes[i].set_title(f'{stock} Over Time')
    axes[i].set_ylabel('Price (USD)')
    axes[i].legend()

plt.xticks(rotation=45)
plt.xlabel('Date')

plt.tight_layout()
plt.show()
```

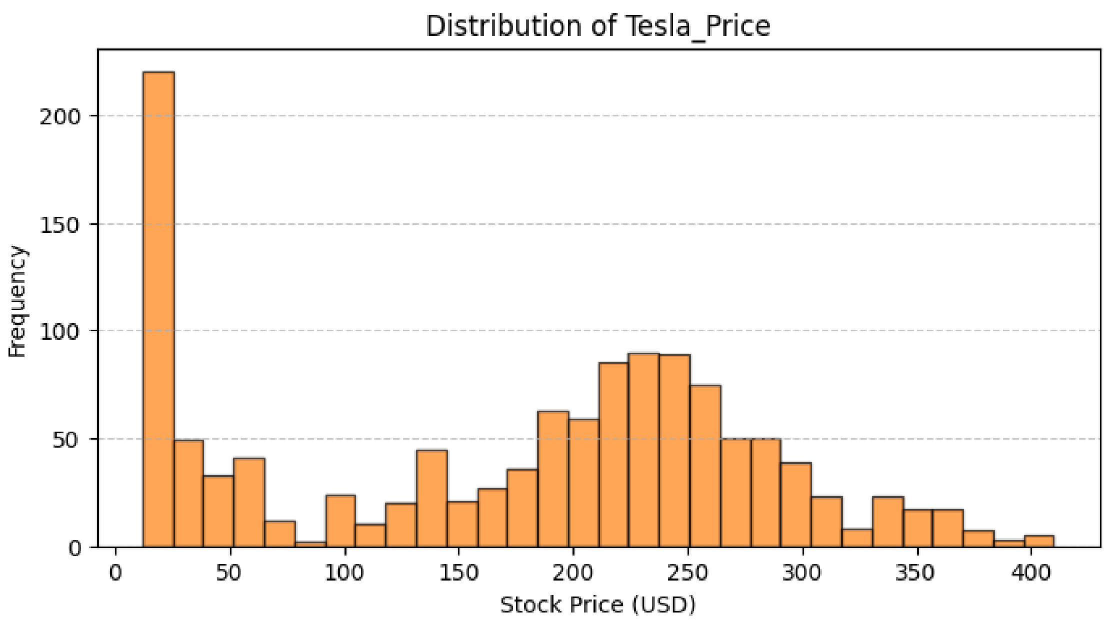


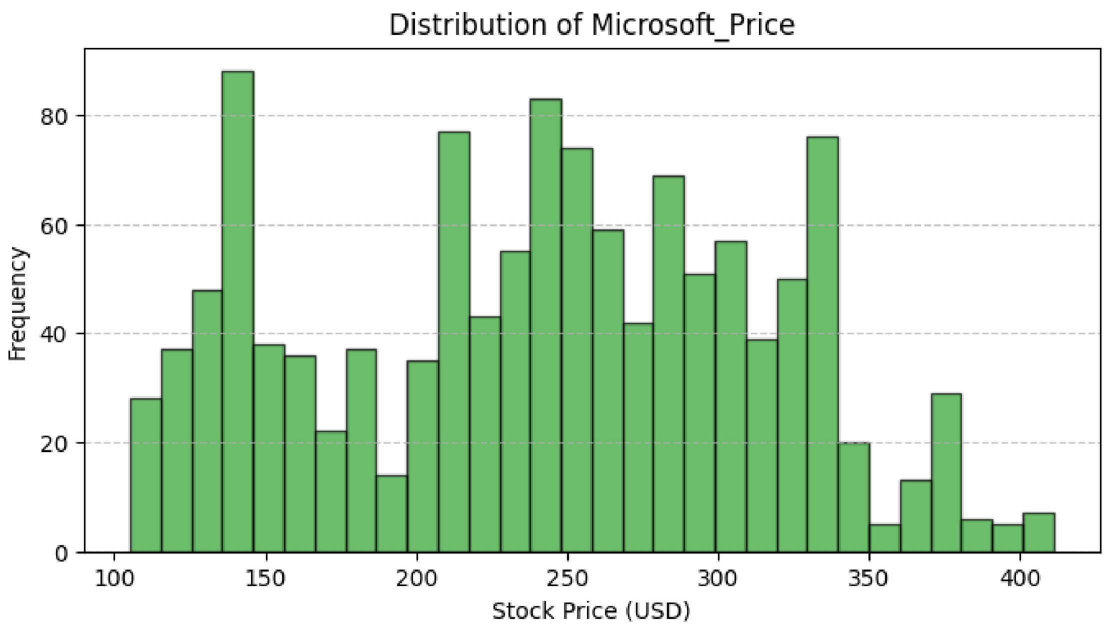
Histogram:

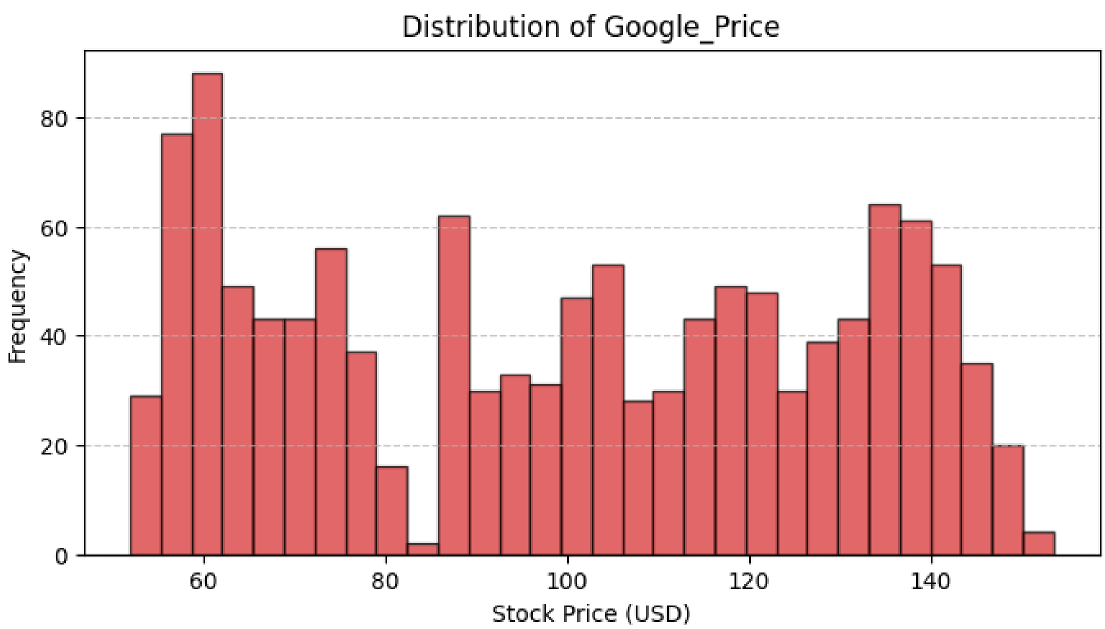
Check distribution of prices over the dataset period.

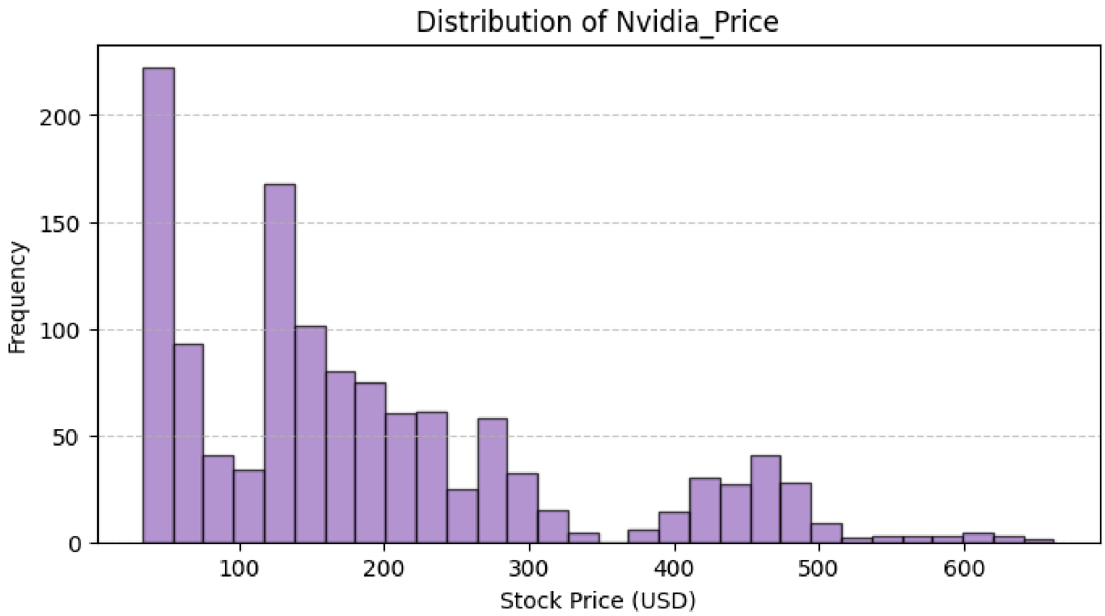
[16]:
```
for stock, color in zip(stock_prices.columns, colors):
    plt.figure(figsize=(8, 4))
    plt.hist(stock_prices[stock], bins=30, color=color, alpha=0.7,␣
 ↪edgecolor='black')
    plt.title(f'Distribution of {stock}')
    plt.xlabel('Stock Price (USD)')
    plt.ylabel('Frequency')
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.show()
```

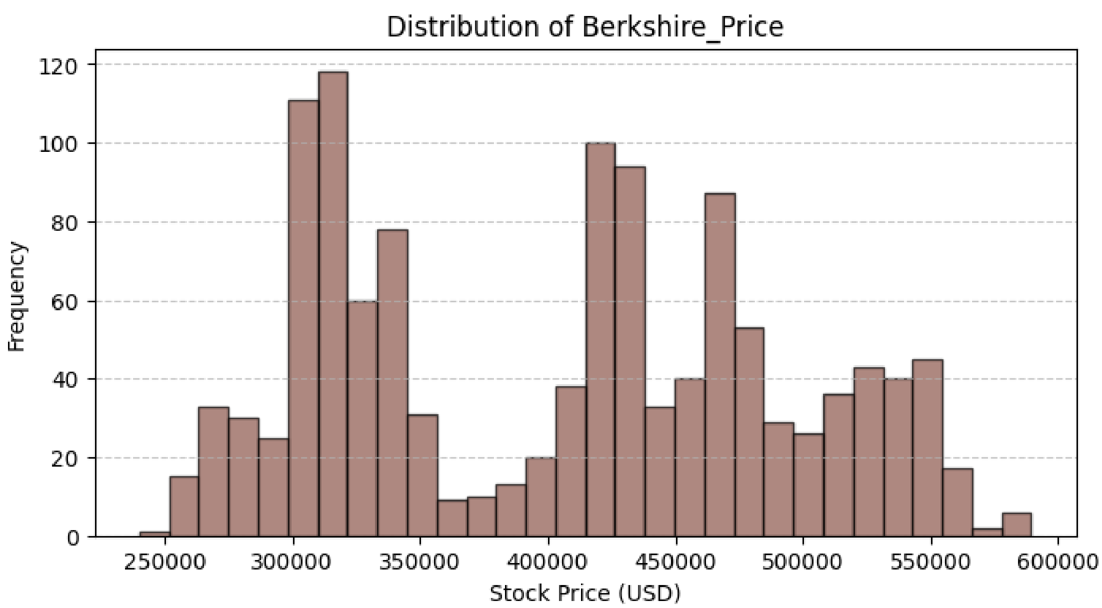

Distribution of Apple_Price
Frequency
0
20
40
60
80
100
40
60
80
100
120
140
160
180
200
Stock Price (USD)

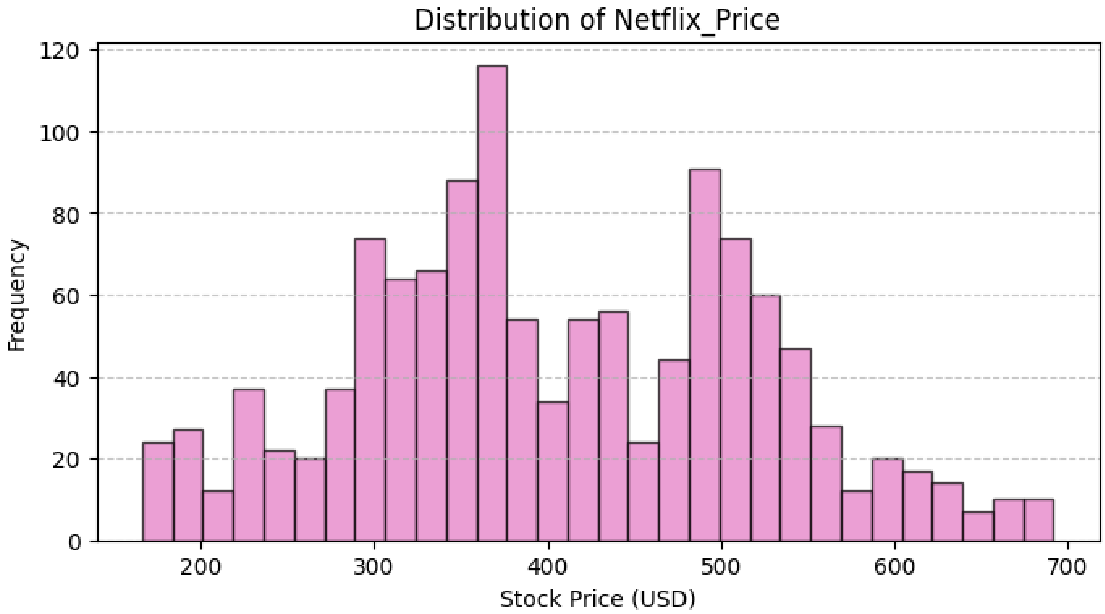

Distribution of Tesla_Price
Frequency
0
50
100
150
200
0
50
100
150
200
250
300
350
400
Stock Price (USD)

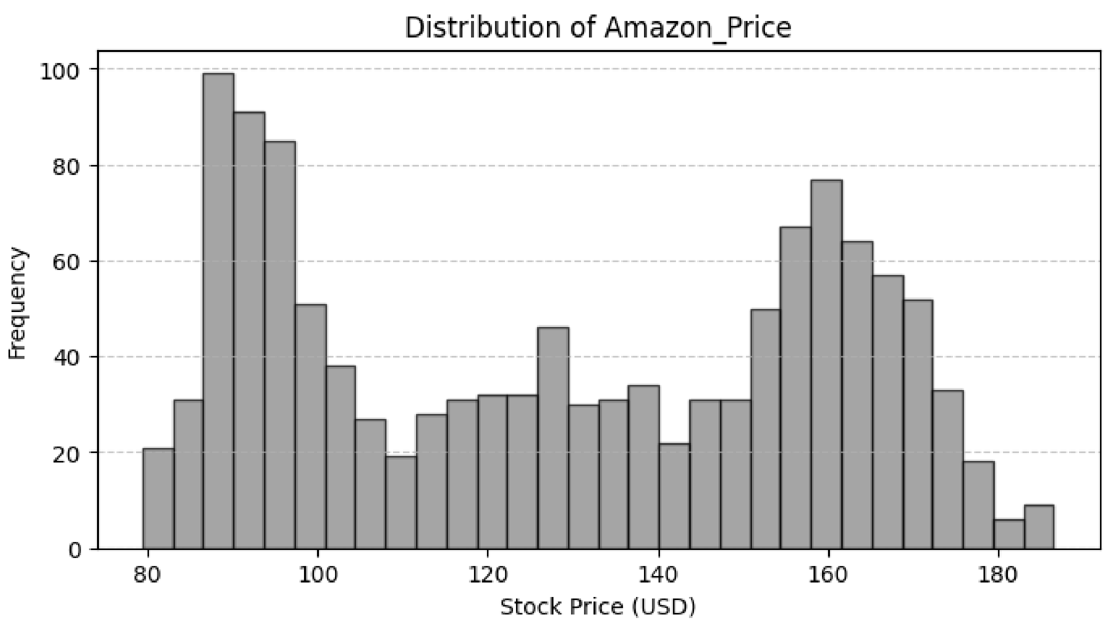

Distribution of Microsoft_Price
Frequency
Stock Price (USD)
0
20
40
60
80
100
150
200
250
300
350
400

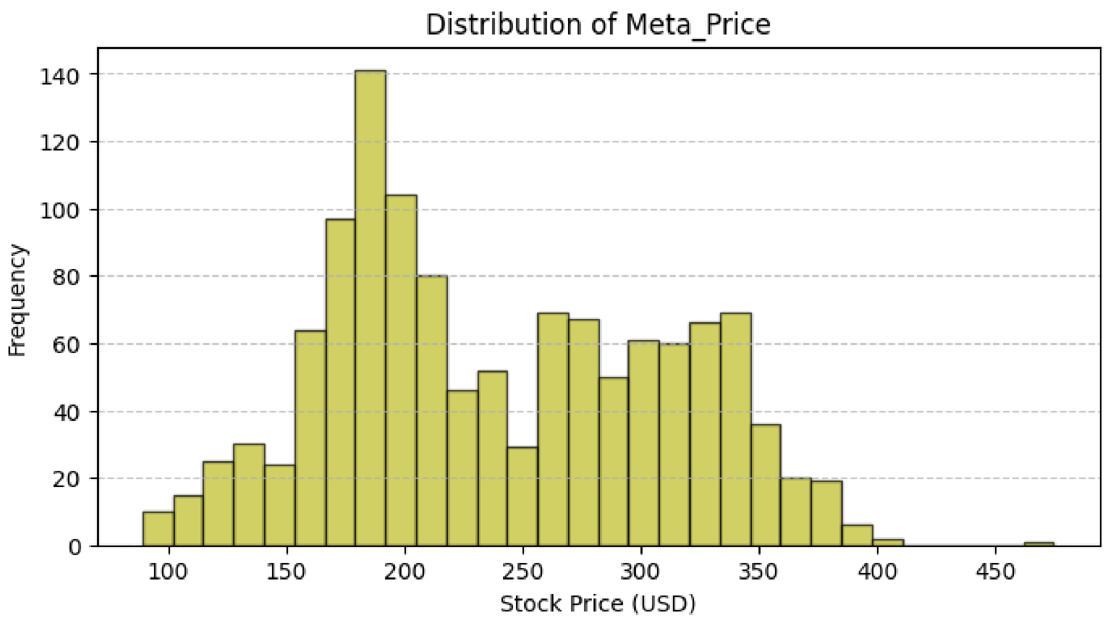

Distribution of Google_Price
Frequency
Stock Price (USD)
0
20
40
60
80
60
80
100
120
140


Distribution of Nvidia_Price
Frequency
Stock Price (USD)
0
50
100
150
200
100
200
300
400
500
600


Distribution of Berkshire_Price
Frequency
Stock Price (USD)
0
20
40
60
80
100
120
250000
300000
350000
400000
450000
500000
550000
600000


Distribution of Netflix_Price
Frequency
0
20
40
60
80
100
120
200
300
400
500
600
700
Stock Price (USD)

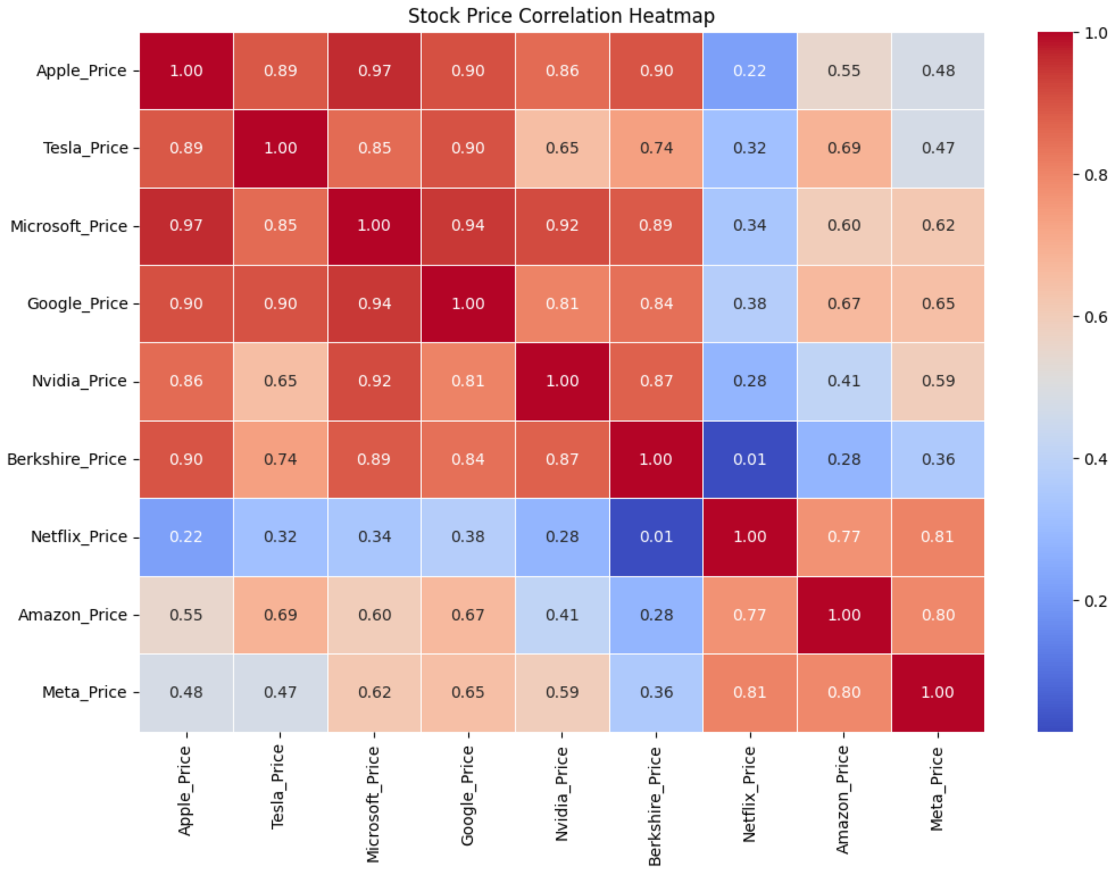

Distribution of Amazon_Price
Frequency
0
20
40
60
80
100
80
100
120
140
160
180
Stock Price (USD)



Heat Map:

Check for correlations of stock prices between companies.

[17]:
```
corr_matrix = stock_prices.corr()

plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Stock Price Correlation Heatmap')
plt.show()
```



Microsoft is the most correlated to the other stocks analyzed (old/new Mag7 and Berkshire).

Netflix is the least connected with a correlation significantly lower than the others. This is logical as it is a streaming service, while the others are mainly technology oriented (Berkshire being an exception).

Berkshire has investments in Apple (40-50% of its portfolio) and Amazon. This is why Berkshire, despite not being a tech company, has a strong positive correlation 0.9.

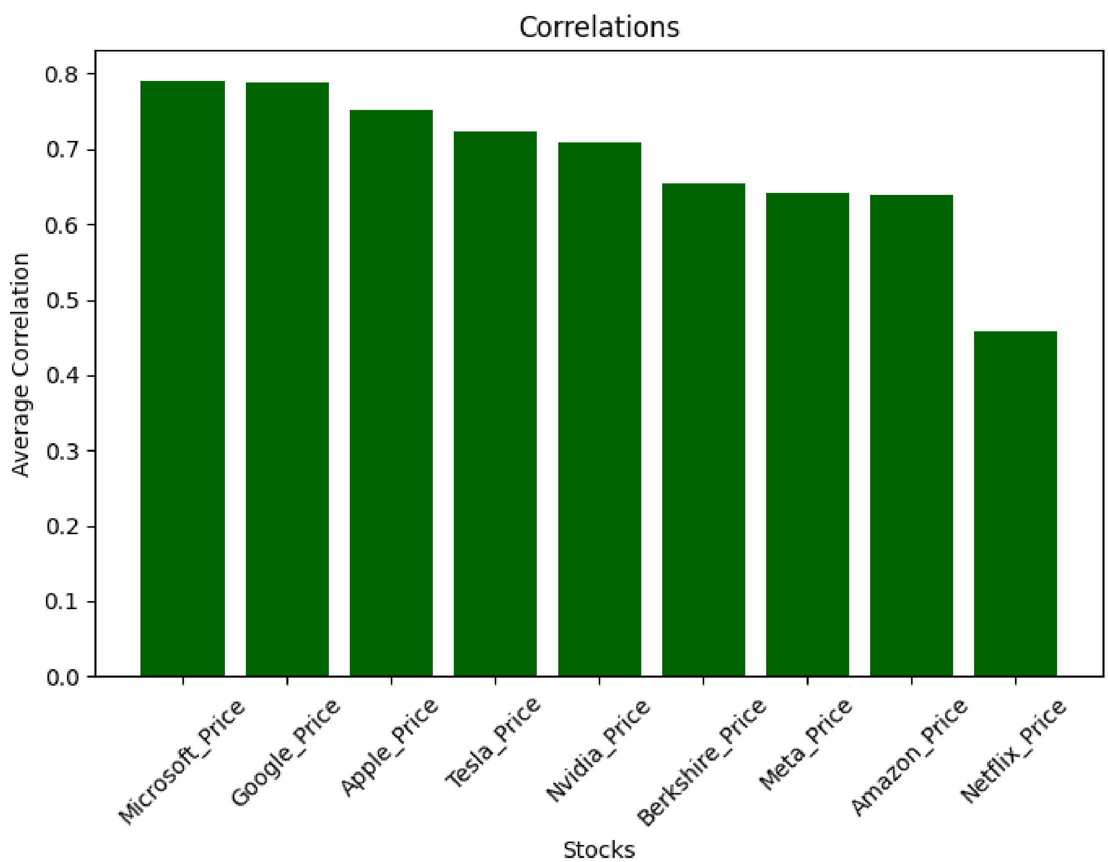
[18]:
```python
average_correlations=[]
for column in corr_matrix.columns:
    average_correlations.append([column,corr_matrix[column].mean()])
ac_frame = pd.DataFrame(average_correlations,columns=['Stocks','Average␣
↪Correlation']).sort_values(by='Average Correlation',ascending=False).
↪set_index('Stocks')
ac_frame
```

[18]:
```
                 Average Correlation
Stocks
Microsoft_Price             0.790848
Google_Price                0.787895
```

```
Apple_Price                 0.751222
Tesla_Price                 0.723545
Nvidia_Price                0.709145
Berkshire_Price             0.654055
Meta_Price                  0.640594
Amazon_Price                0.639962
Netflix_Price               0.457517
```

[19]:

```
plt.figure(figsize=(8, 5))
plt.bar(ac_frame.index, ac_frame['Average Correlation'], color='darkgreen')

plt.xticks(ac_frame.index, rotation=45)

plt.xlabel("Stocks")
plt.ylabel("Average Correlation")
plt.title("Correlations")
plt.show()
```



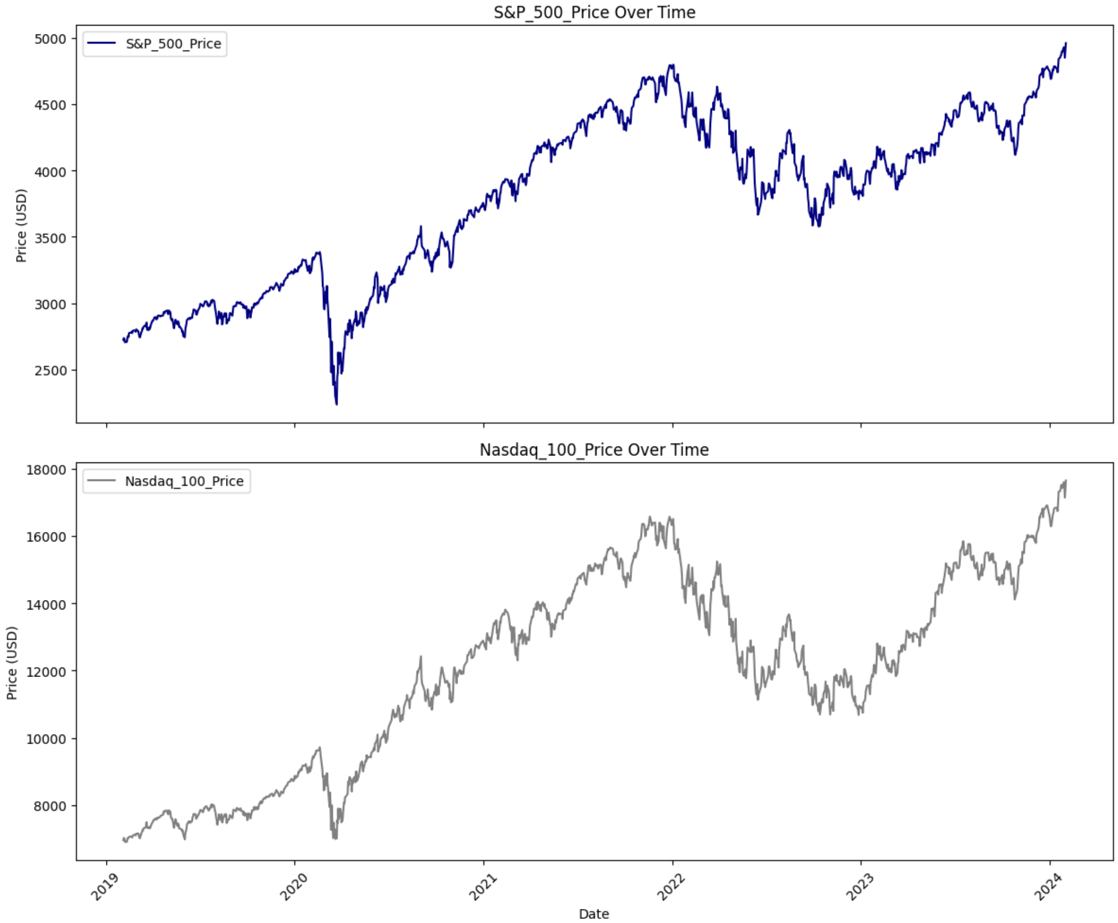
1.0.6 6) Market Visualizations

Line Plots:

Analyzing the movements of the markets over time.

Both the S&P500 and Nasdaq prices have grown tremendously in the timeframe. The S&P500 grew by nearly double and the Nasdaq has more than doubled.

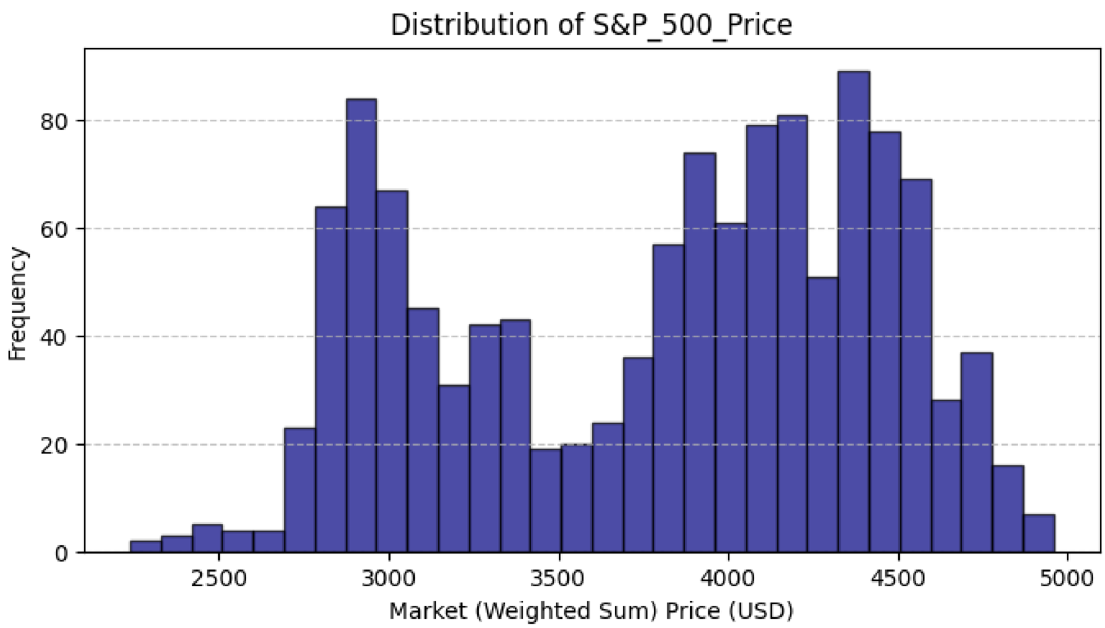
[20]:
```
colors = ['navy','gray']

fig, axes = plt.subplots(nrows=len(market_prices.columns), ncols=1,␣
 ↪figsize=(12,10), sharex=True)

for i, (market,color) in enumerate(zip(market_prices.columns,colors)):
    axes[i].plot(df['Date'], df[market], label=market, color=color)
    axes[i].set_title(f'{market} Over Time')
    axes[i].set_ylabel('Price (USD)')
    axes[i].legend()

plt.xticks(rotation=45)
plt.xlabel('Date')

plt.tight_layout()
plt.show()
```

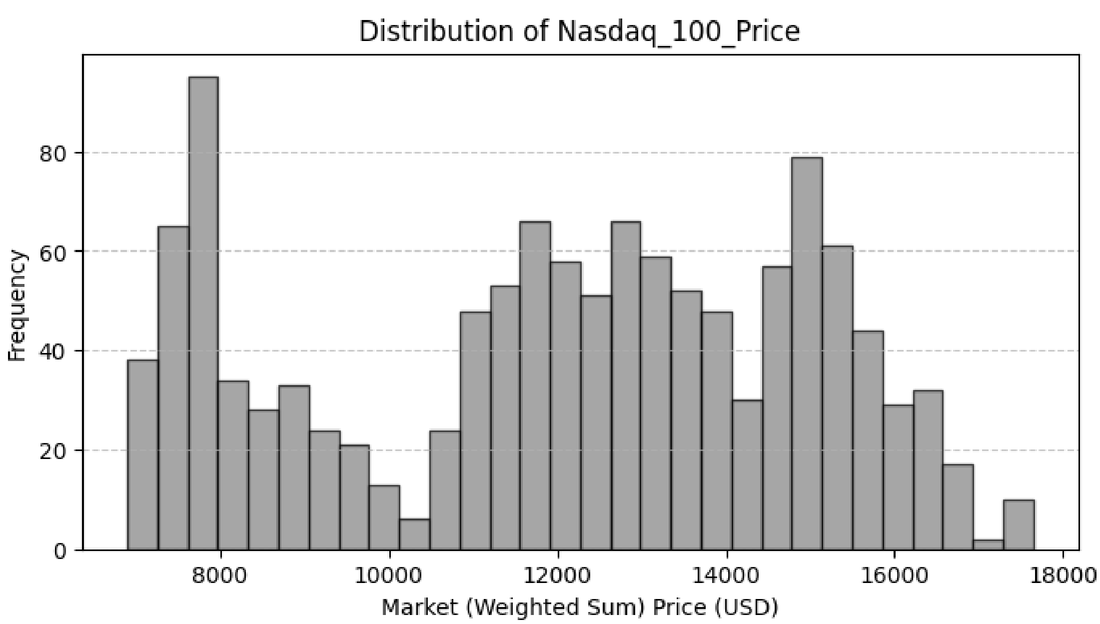


Histogram:

Check distribution of prices over the dataset period.

[21]:
```
for market, color in zip(market_prices.columns, colors):
    plt.figure(figsize=(8, 4))
    plt.hist(market_prices[market], bins=30, color=color, alpha=0.7,␣
 ↪edgecolor='black')
    plt.title(f'Distribution of {market}')
    plt.xlabel('Market (Weighted Sum) Price (USD)')
    plt.ylabel('Frequency')
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.show()
```



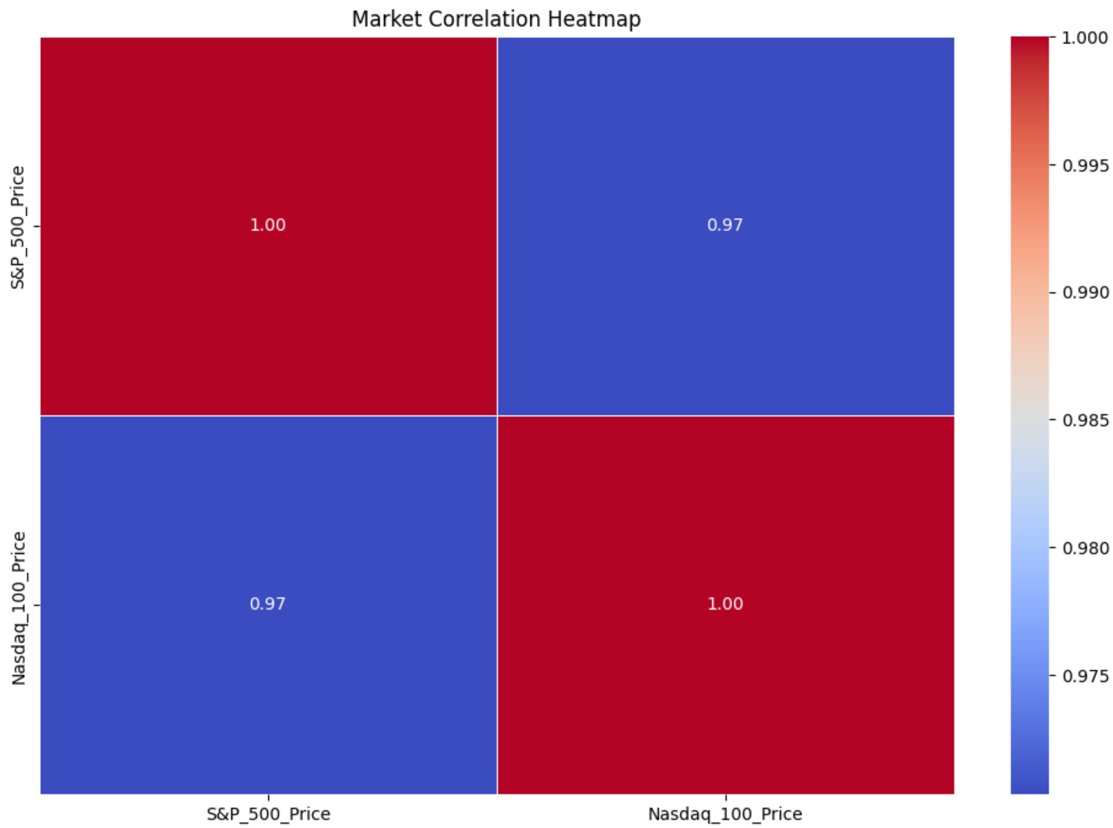


Heat Map:

Check for correlations of market prices between Nasdaq and S&P500.

Strong positive correlation between the them (0.985).

[22]:
```
corr_matrix = market_prices.corr()

plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Market Correlation Heatmap')
plt.show()
```

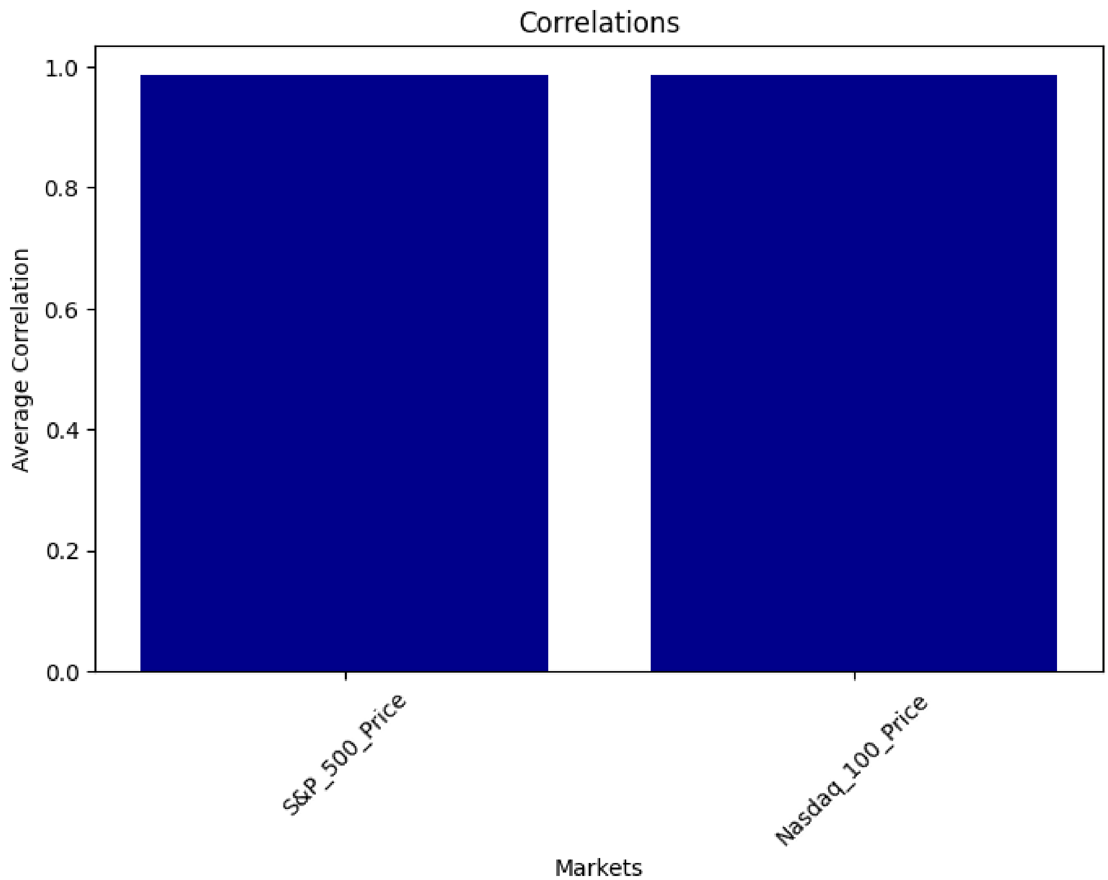


[23]:
```
average_correlations=[]
for column in corr_matrix.columns:
    average_correlations.append([column,corr_matrix[column].mean()])
ac_frame = pd.DataFrame(average_correlations,columns=['Markets','Average␣
↪Correlation']).sort_values(by='Average Correlation',ascending=False).
↪set_index('Markets')
ac_frame
```

[23]:

	Average Correlation
Markets	
S&P_500_Price	0.985157
Nasdaq_100_Price	0.985157

[24]:
```
plt.figure(figsize=(8, 5))
plt.bar(ac_frame.index, ac_frame['Average Correlation'], color='darkblue')

plt.xticks(ac_frame.index, rotation=45)

plt.xlabel("Markets")
plt.ylabel("Average Correlation")
plt.title("Correlations")
plt.show()
```



1.0.7 7) Commodity Visualizations

Line Plots:

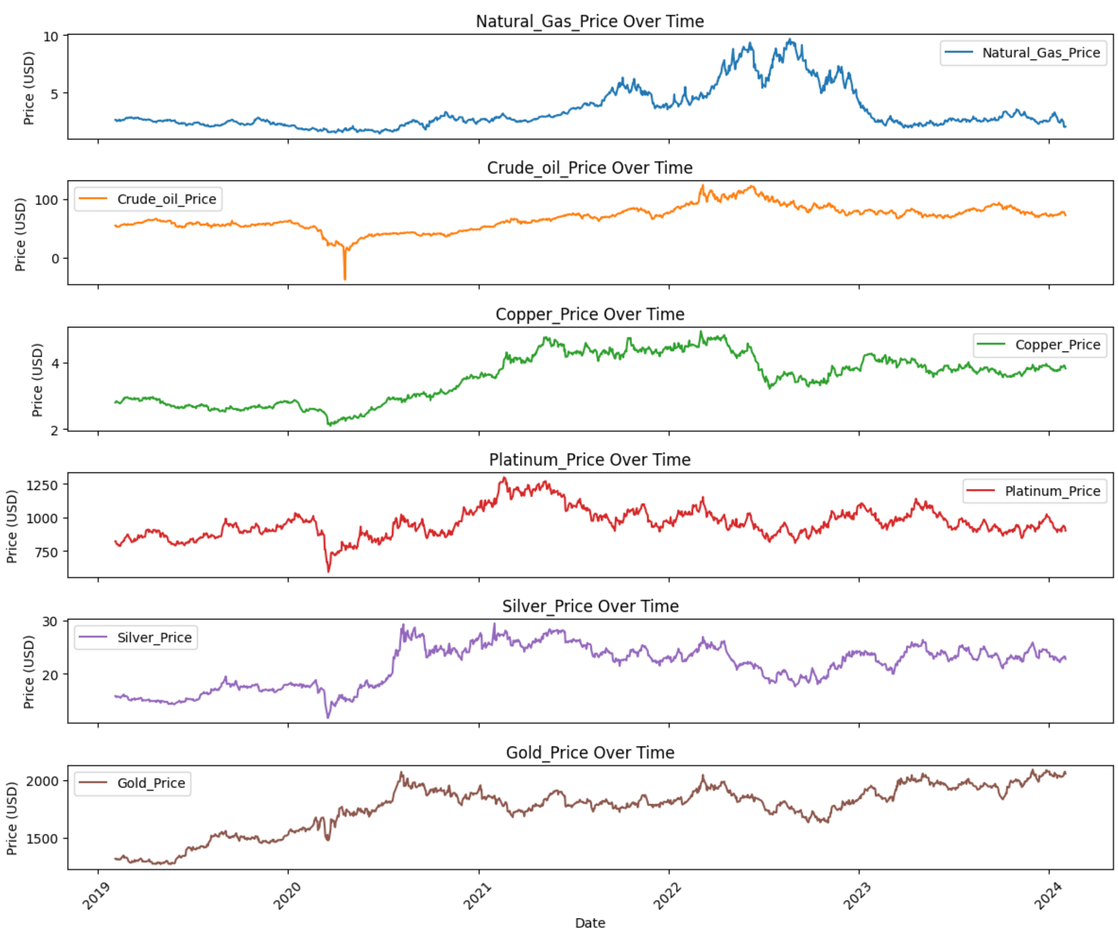
Analyzing the movements of the commodities over time.

Crude oil price went below zero for a brief moment between 2020 and 2021.

Most of the commodities have not changed in value over time. The main exception here is gold. Gold has risen by 500 USD between 2019 and 2024 according to the data.

[25]:

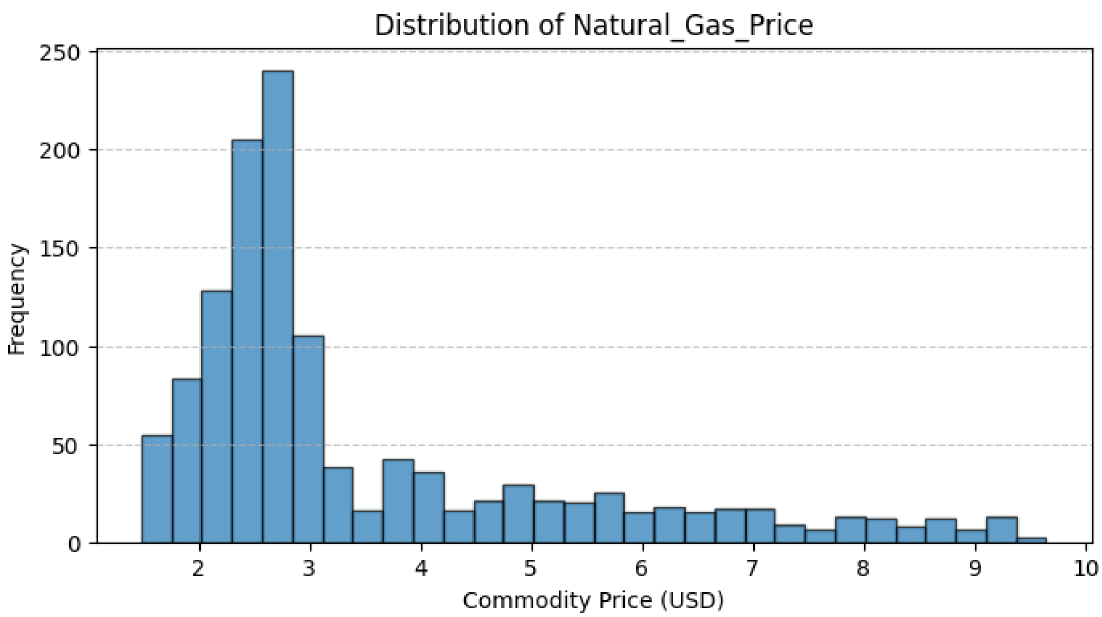
```python
colors = sns.color_palette("tab10", len(commodity_prices.columns))

fig, axes = plt.subplots(nrows=len(commodity_prices.columns), ncols=1,␣
 ↪figsize=(12,10), sharex=True)

for i, (commodity,color) in enumerate(zip(commodity_prices.columns,colors)):
    axes[i].plot(df['Date'], df[commodity], label=commodity, color=color)
    axes[i].set_title(f'{commodity} Over Time')
    axes[i].set_ylabel('Price (USD)')
    axes[i].legend()

plt.xticks(rotation=45)
plt.xlabel('Date')

plt.tight_layout()
plt.show()
```

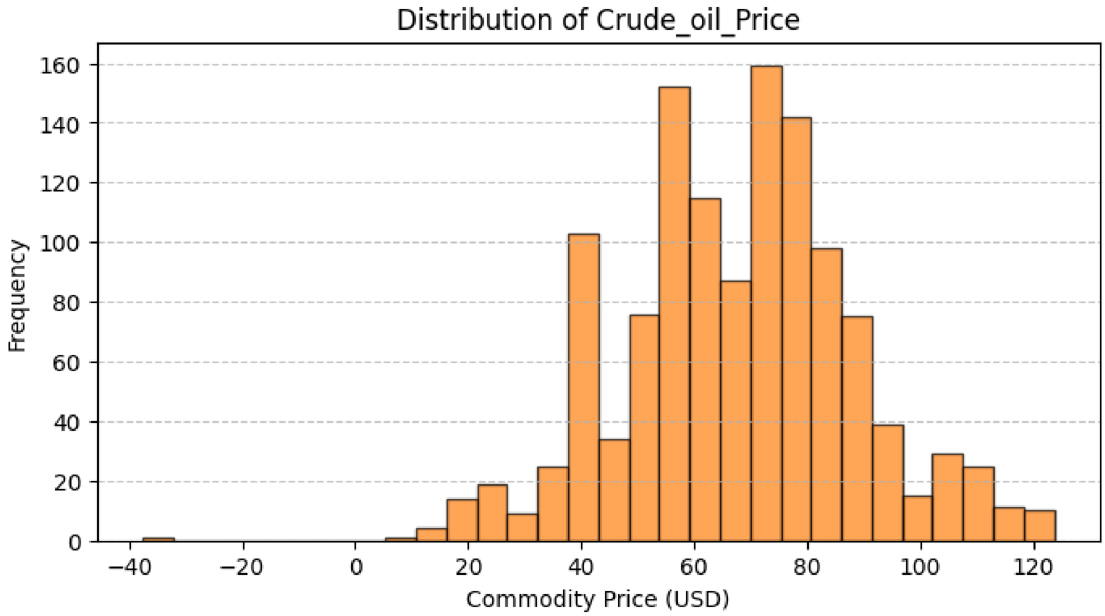


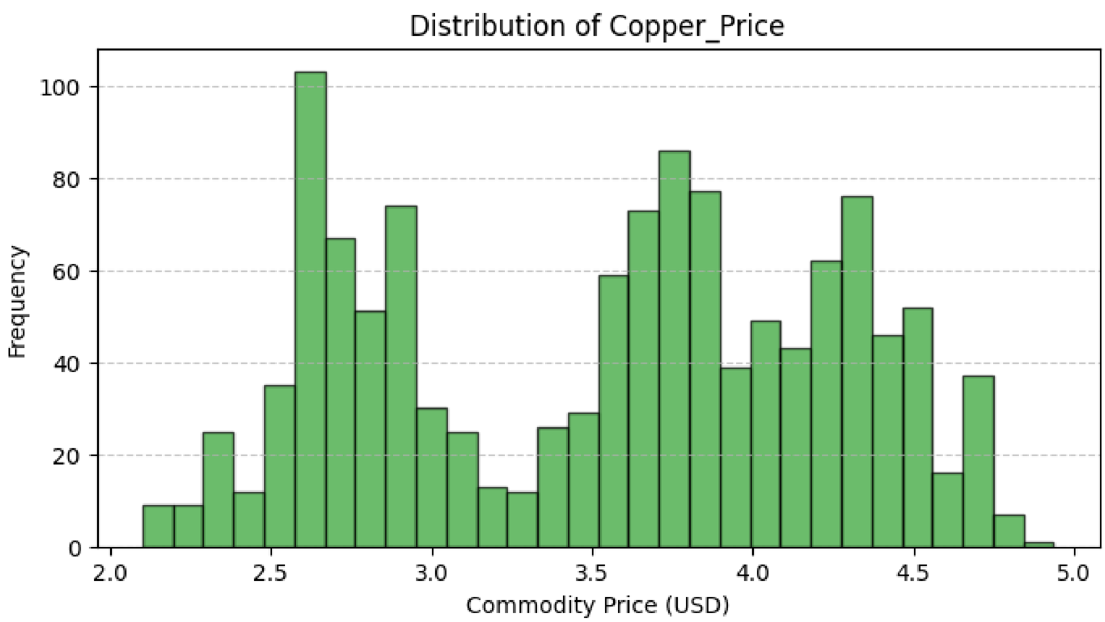
Histogram:

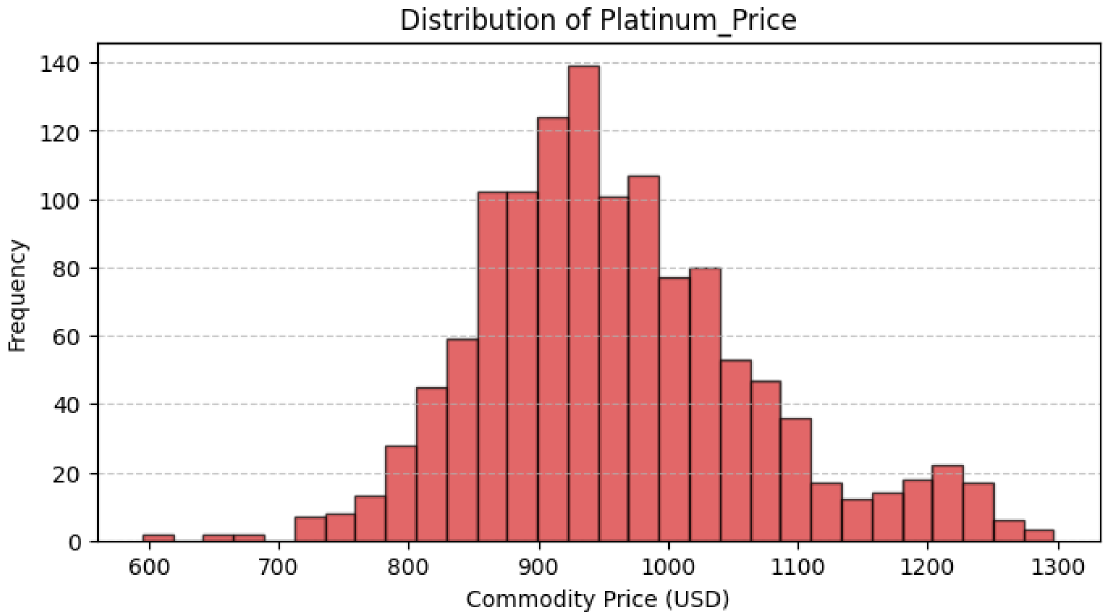
Check distribution of prices over the dataset period.

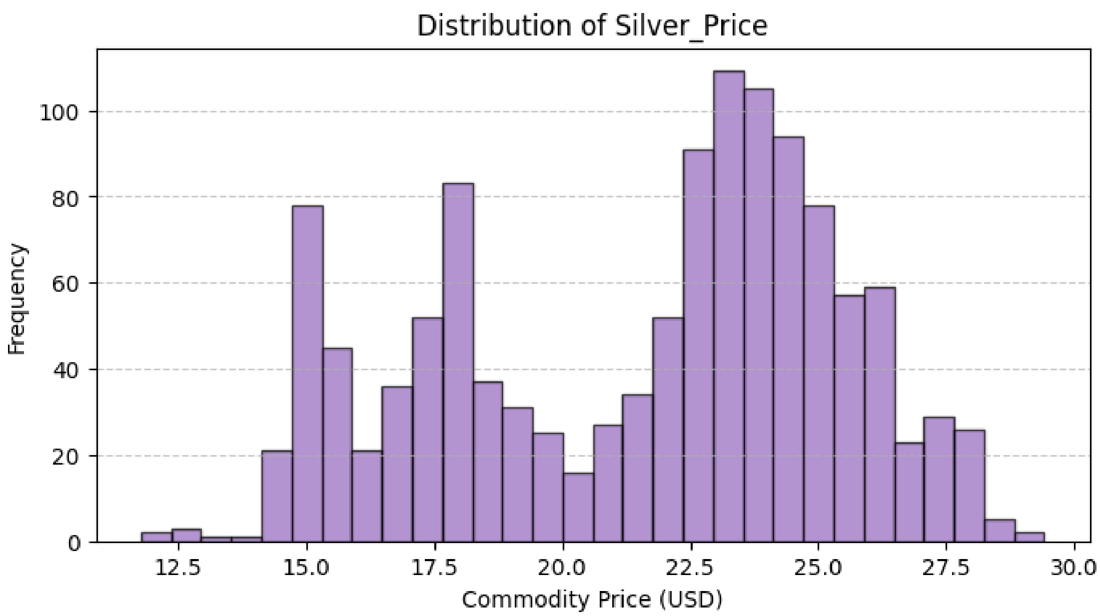
[26]:
```
for commodity, color in zip(commodity_prices.columns, colors):
    plt.figure(figsize=(8, 4))
    plt.hist(commodity_prices[commodity], bins=30, color=color, alpha=0.7,␣
 ↪edgecolor='black')
    plt.title(f'Distribution of {commodity}')
    plt.xlabel('Commodity Price (USD)')
    plt.ylabel('Frequency')
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.show()
```

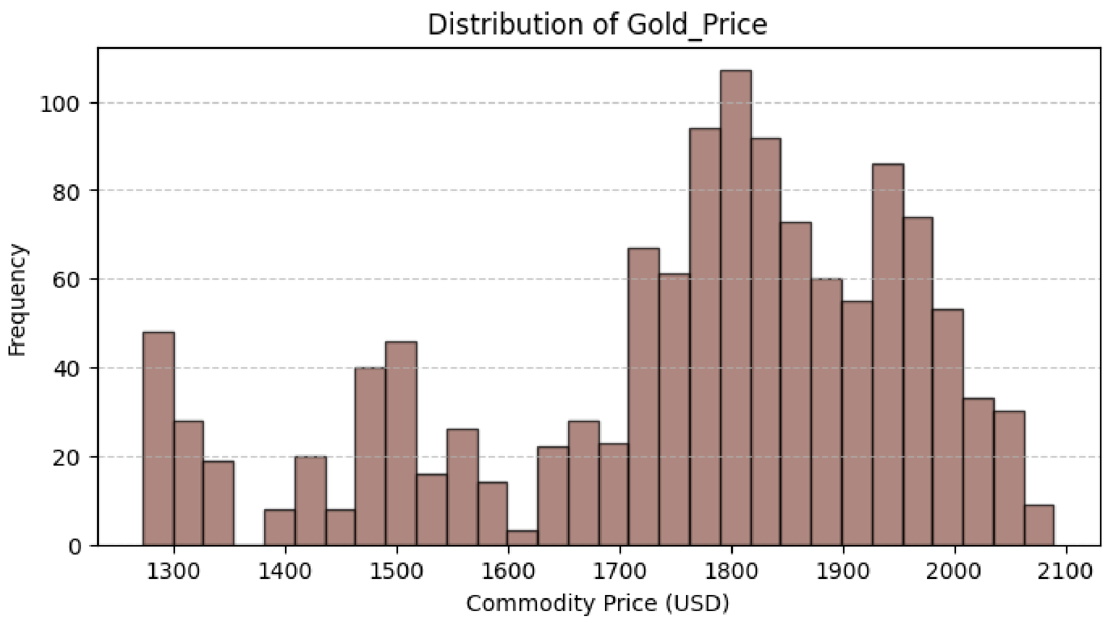



Distribution of Crude_oil_Price
Frequency
Commodity Price (USD)
0
20
40
60
80
100
120
140
160
−40
−20
0
20
40
60
80
100
120


Distribution of Copper_Price
Frequency
Commodity Price (USD)
0
20
40
60
80
100
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Distribution of Platinum_Price
Frequency
Commodity Price (USD)
0
20
40
60
80
100
120
140
600
700
800
900
1000
1100
1200
1300


Distribution of Silver_Price
Frequency
Commodity Price (USD)
0
20
40
60
80
100
12.5
15.0
17.5
20.0
22.5
25.0
27.5
30.0



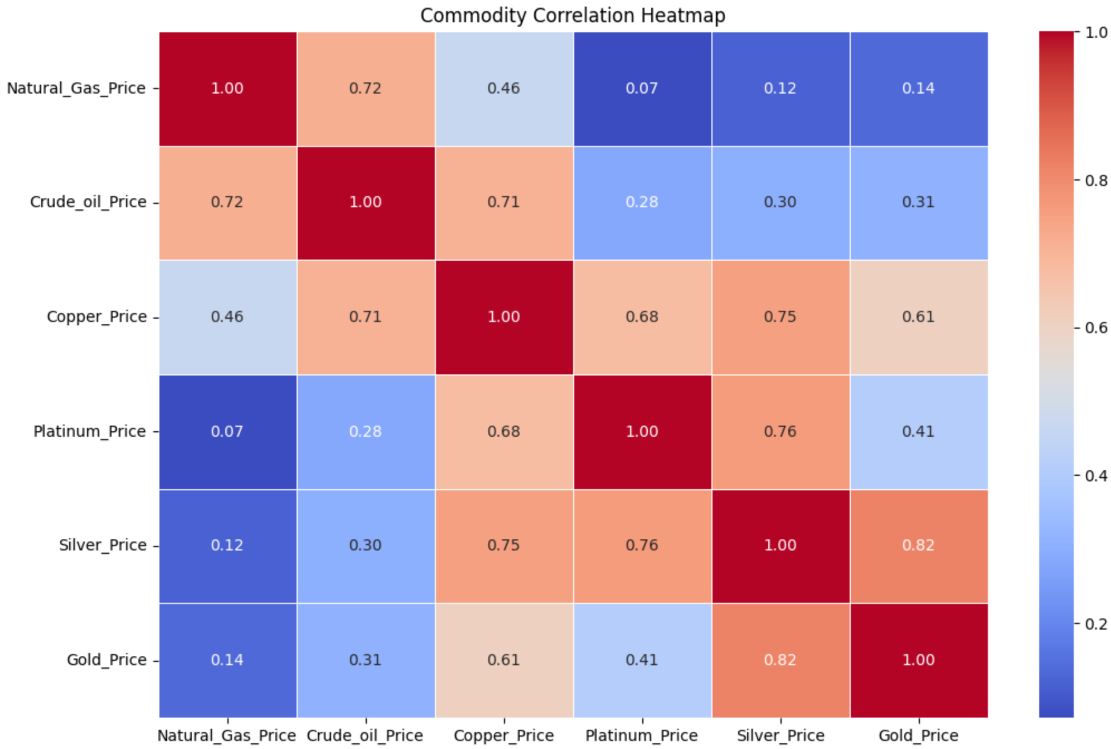
Heat Map:

Check for correlations of prices between commodities.

Most correlated to other commodities on average is copper (0.70). Its strongest relationships are with crude oil and silver.

Natural gas was the least correlated to the others on average (0.42).

[27]:
```
corr_matrix = commodity_prices.corr()

plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Commodity Correlation Heatmap')
plt.show()
```



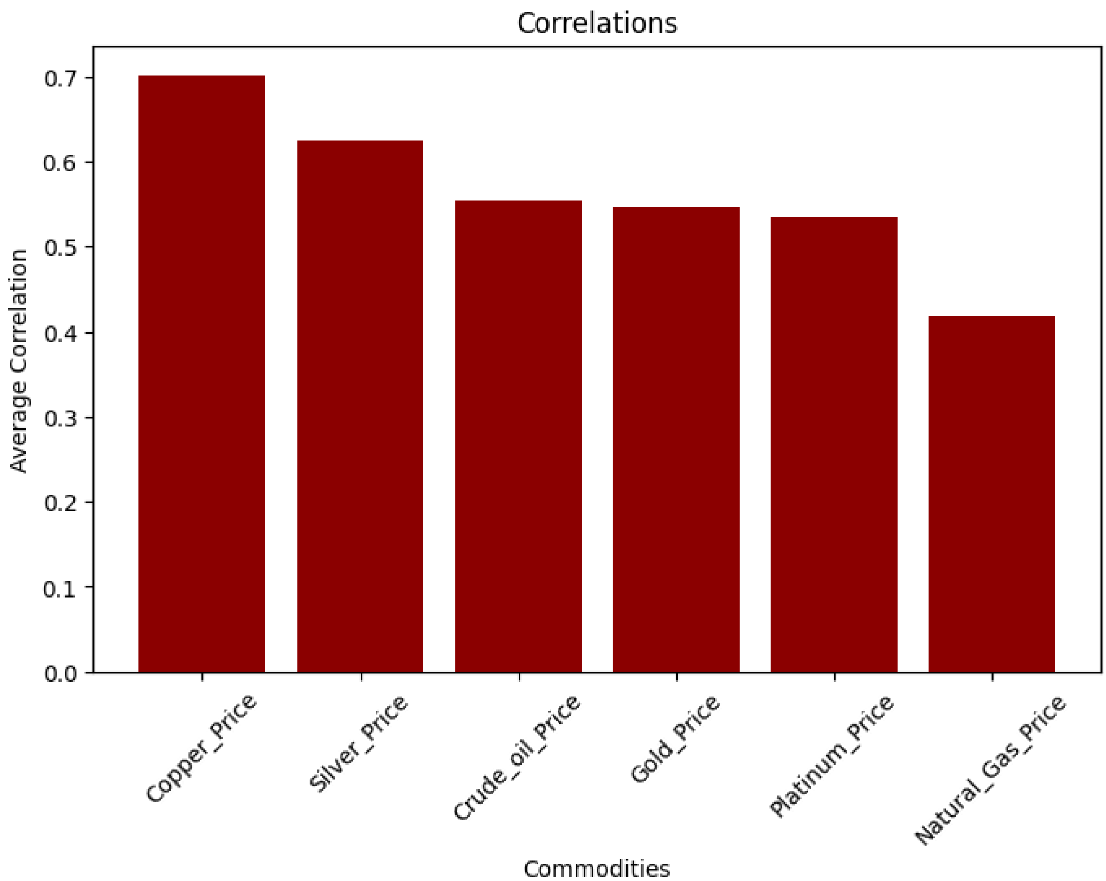
[28]:
```python
average_correlations=[]
for column in corr_matrix.columns:
    average_correlations.append([column,corr_matrix[column].mean()])
ac_frame = pd.DataFrame(average_correlations,columns=['Commodity','Average␣
↪Correlation']).sort_values(by='Average Correlation',ascending=False).
↪set_index('Commodity')
ac_frame
```

[28]:

	Average Correlation
Commodity	
Copper_Price	0.701322
Silver_Price	0.625721
Crude_oil_Price	0.553617
Gold_Price	0.546527
Platinum_Price	0.534400
Natural_Gas_Price	0.417986

[29]:
```python
plt.figure(figsize=(8, 5))
plt.bar(ac_frame.index, ac_frame['Average Correlation'], color='darkred')

plt.xticks(ac_frame.index, rotation=45)

plt.xlabel("Commodities")
```

```
plt.ylabel("Average Correlation")
plt.title("Correlations")
plt.show()
```



1.0.8 8) Combined Analysis

[30]:
```
merged = df[['Apple_Price', 'Tesla_Price', 'Microsoft_Price', 'Google_Price',
             'Nvidia_Price', 'Berkshire_Price','Netflix_Price', 'Amazon_Price',
             'Meta_Price','S&P_500_Price',␣
 ↪'Nasdaq_100_Price','Natural_Gas_Price',
             'Crude_oil_Price', 'Copper_Price', 'Platinum_Price',␣
 ↪'Silver_Price',
             'Gold_Price']]
```

[31]:
```
merged
```

[31]:
```
      Apple_Price  Tesla_Price  Microsoft_Price  Google_Price  Nvidia_Price  \
0          185.85       187.91           411.22        142.38        661.60
```

1	186.86	188.86	403.78	141.16	630.27
2	184.40	187.29	397.58	140.10	615.27
3	188.04	191.59	408.59	151.46	627.74
4	191.73	190.93	409.72	153.51	624.65
…	…	…	…	…	…
1238	42.60	20.39	105.67	55.12	37.04
1239	42.73	20.50	105.27	55.30	36.85
1240	43.56	21.15	106.03	56.14	38.25
1241	43.55	21.42	107.22	57.59	37.49
1242	42.81	20.86	105.74	57.07	37.30

	Berkshire_Price	Netflix_Price	Amazon_Price	Meta_Price	S&P_500_Price \
0	589498.0	564.64	171.81	474.99	4958.61
1	581600.0	567.51	159.28	394.78	4906.19
2	578020.0	564.11	155.20	390.14	4848.87
3	584680.0	562.85	159.00	400.06	4924.97
4	578800.0	575.79	161.26	401.02	4927.93
…	…	…	…	…	…
1238	300771.0	347.57	79.41	167.33	2707.88
1239	302813.0	344.71	80.72	166.38	2706.05
1240	308810.0	352.19	82.01	170.49	2731.61
1241	310700.0	355.81	82.94	171.16	2737.70
1242	312000.0	351.34	81.67	169.25	2724.87

	Nasdaq_100_Price	Natural_Gas_Price	Crude_oil_Price	Copper_Price \
0	17642.73	2.079	72.28	3.8215
1	17344.71	2.050	73.82	3.8535
2	17137.24	2.100	75.85	3.9060
3	17476.71	2.077	77.82	3.9110
4	17596.27	2.490	76.78	3.8790
…	…	…	…	…
1238	6913.13	2.583	52.72	2.8140
1239	6904.98	2.551	52.64	2.8320
1240	6997.62	2.662	54.01	2.8400
1241	7023.52	2.662	53.66	2.8205
1242	6959.96	2.660	54.56	2.7975

	Platinum_Price	Silver_Price	Gold_Price
0	901.60	22.796	2053.7
1	922.30	23.236	2071.1
2	932.60	23.169	2067.4
3	931.70	23.225	2050.9
4	938.30	23.134	2034.9
…	…	…	…
1238	802.20	15.809	1318.5
1239	800.80	15.713	1314.2
1240	807.10	15.701	1314.4

```
1241           821.35         15.836       1319.2
1242           822.50         15.886       1319.3

[1243 rows x 17 columns]
```

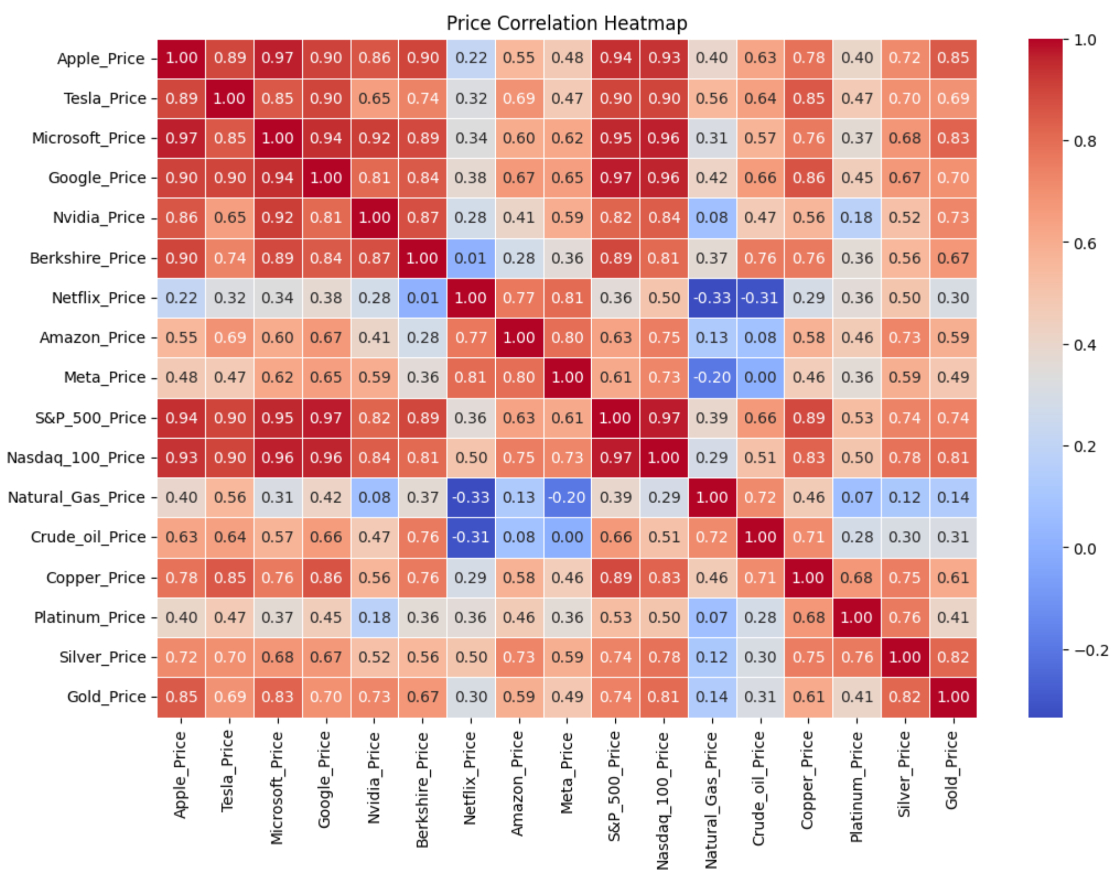
Heat Map:

Check for correlations of prices between commodities, stocks and markets.

It is logical that the markets have the highest correlations (0.76) as they represent the combination of all the prices.

The least correlated are natural gas prices (0.29). This means it is not linearly dependent on the other variables here.

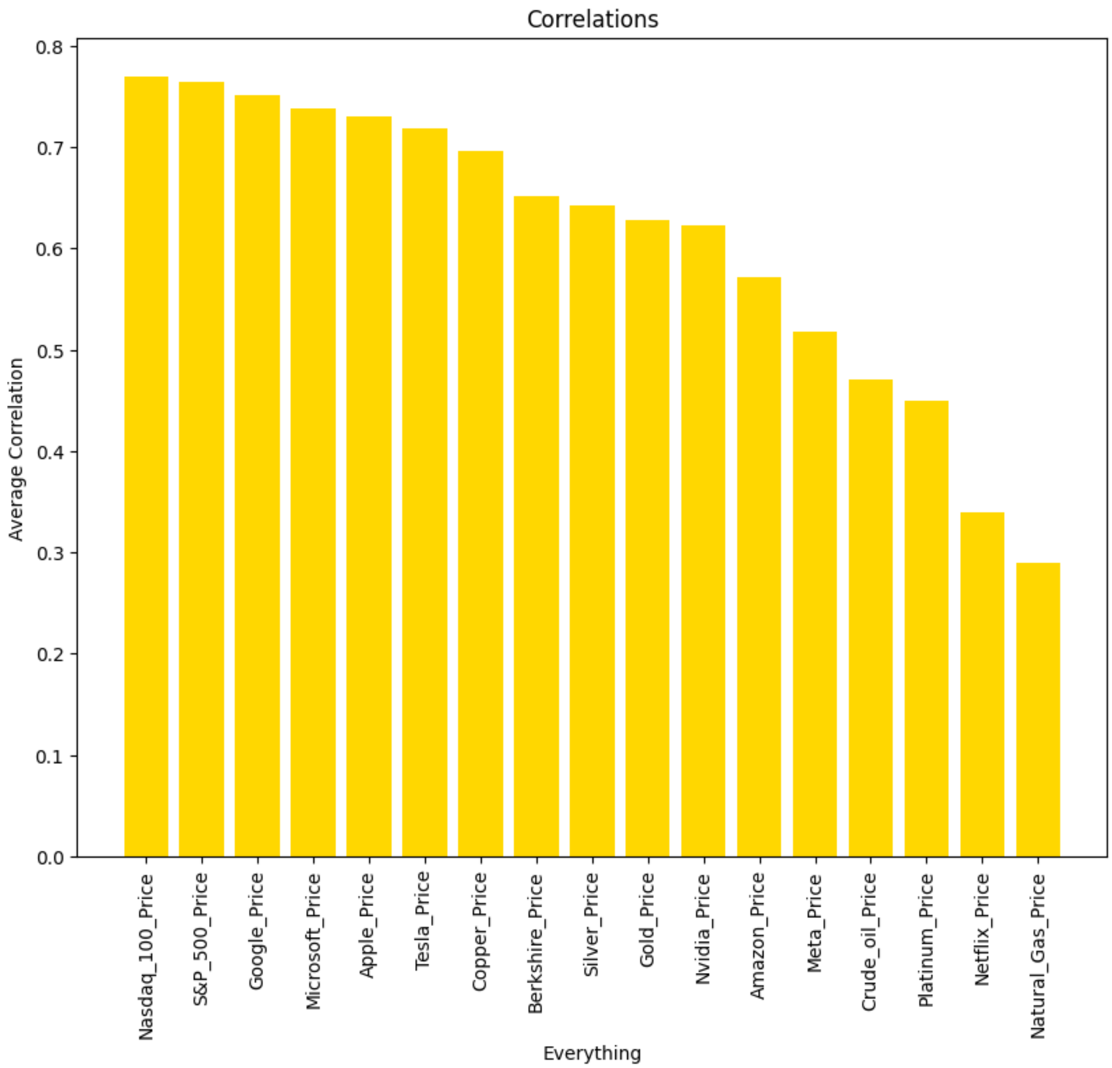
[32]:
```
corr_matrix = merged.corr()

plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Price Correlation Heatmap')
plt.show()
```



[33]:
```
average_correlations=[]
for column in corr_matrix.columns:
    average_correlations.append([column,corr_matrix[column].mean()])
ac_frame = pd.DataFrame(average_correlations,columns=['Price','Average␣
↪Correlation']).sort_values(by='Average Correlation',ascending=False).
↪set_index('Price')
ac_frame
```

[33]:
```
                     Average Correlation
Price
Nasdaq_100_Price                0.769442
S&P_500_Price                   0.764756
Google_Price                    0.751827
Microsoft_Price                 0.738718
Apple_Price                     0.730277
Tesla_Price                     0.718964
Copper_Price                    0.696349
Berkshire_Price                 0.651488
Silver_Price                    0.642931
Gold_Price                      0.628075
Nvidia_Price                    0.622571
Amazon_Price                    0.572071
Meta_Price                      0.518160
Crude_oil_Price                 0.471140
Platinum_Price                  0.450478
Netflix_Price                   0.340260
Natural_Gas_Price               0.289678
```

[34]:
```
plt.figure(figsize=(10, 8))
plt.bar(ac_frame.index, ac_frame['Average Correlation'], color='gold')

plt.xticks(ac_frame.index, rotation=90)

plt.xlabel("Everything")
plt.ylabel("Average Correlation")
plt.title("Correlations")
plt.show()
```



- **Nasdaq 100 & S&P 500 (0.77 & 0.76 correlation):**
  These indices have a strong correlation with commodities and major tech stocks because they represent the broader market. Their movement often reflects macroeconomic conditions, monetary policy, and investor sentiment, which also drive commodity prices and large-cap stocks.
- **Google & Microsoft (0.75 & 0.74 correlation):**
  These two tech giants show strong correlation with commodities and the market, likely due to their central role in the economy. Their revenues are tied to advertising, cloud computing, and enterprise software, all of which depend on economic growth and investment cycles.
- **Copper Price (0.70 correlation):**
  Copper is an industrial metal widely used in construction and electronics. Its strong correlation with stock prices suggests that investors see copper demand as an indicator of economic expansion, which also benefits companies in tech and manufacturing.

- **Gold Price (0.63 correlation):**
  While gold is traditionally a safe-haven asset, its correlation with equities suggests it may have been moving in response to inflation expectations rather than acting as a hedge against market declines.