```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
data = pd.read_excel("/content/Google Dataset.xlsx")
```

```
data[data['Change %']==0.0]
```

	Month Starting	Open	High	Low	Close	Change %	Avg. Volume
100	Aug. 01, 2014	28.52	29.37	28.00	28.58	0.0	26313579
105	Mar. 27, 2014	28.40	28.40	27.65	27.85	0.0	432192

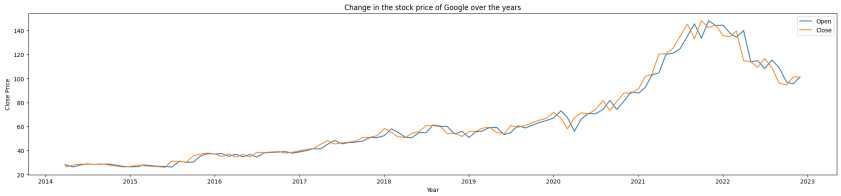
```
data['Month Starting'] = pd.to_datetime(data['Month Starting'], errors='coerce').dt.date
```

```
#Replacing the missing values after cross verifying
data['Month Starting'][31] = pd.to_datetime('2020-05-01')
data['Month Starting'][43] = pd.to_datetime('2019-05-01')
data['Month Starting'][55] = pd.to_datetime('2018-05-01')
```

```
<ipython-input-6-65ec725f754b>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  data['Month Starting'][31] = pd.to_datetime('2020-05-01')
<ipython-input-6-65ec725f754b>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  data['Month Starting'][43] = pd.to_datetime('2019-05-01')
<ipython-input-6-65ec725f754b>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  data['Month Starting'][55] = pd.to_datetime('2018-05-01')
```

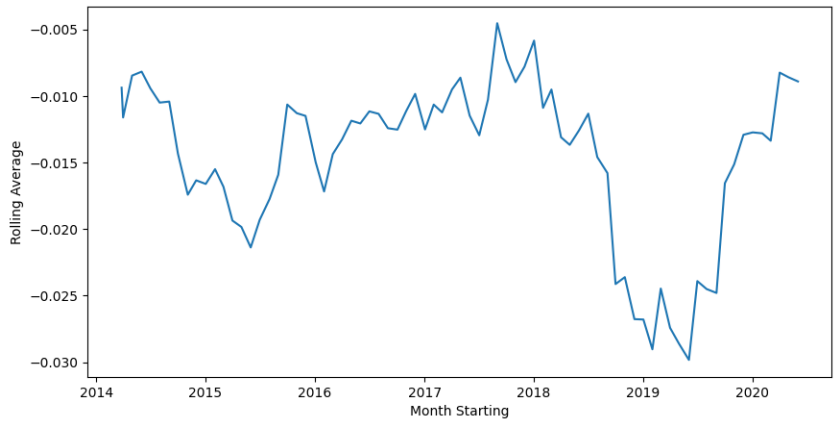
```
plt.figure(figsize=(25,5))
plt.plot(data['Month Starting'],data['Open'], label='Open')
plt.plot(data['Month Starting'],data['Close'], label='Close')
plt.xlabel('Year')
plt.ylabel('Close Price')
plt.legend()
plt.title('Change in the stock price of Google over the years')
```

```
Text(0.5, 1.0, 'Change in the stock price of Google over the years')
```



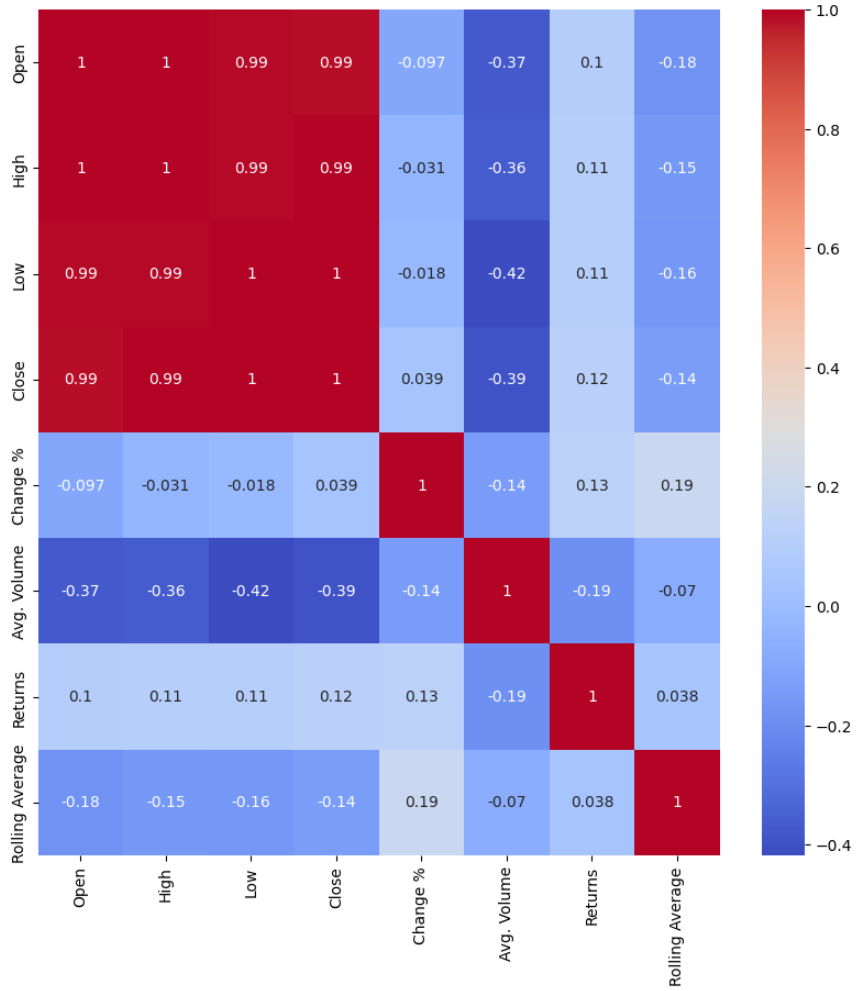

```
# Calculating the daily returns
data['Returns'] = data['Close'].pct_change()

# Calculating the rolling average of the returns
data['Rolling Average'] = data['Returns'].rolling(window=30).mean()

plt.figure(figsize=(10,5))

''' Creating a line plot using the 'Month Starting' column as the x-axis
and the 'Rolling Average' column as the y-axis'''

sns.lineplot(x='Month Starting', y='Rolling Average', data=data)
```

```
<Axes: xlabel='Month Starting', ylabel='Rolling Average'>
```




```
corr = data.corr()
plt.figure(figsize=(10,10))
sns.heatmap(corr, annot=True, cmap='coolwarm')
```

```
<ipython-input-11-98070f5f2505>:1: FutureWarning: The default value of numeric_only
  corr = data.corr()
<Axes: >
```




```
from sklearn.preprocessing import StandardScaler
```

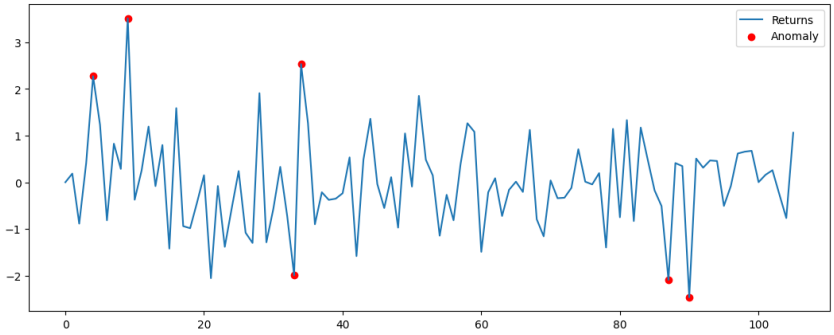
```
scaler = StandardScaler()
data['Returns'] = scaler.fit_transform(data['Returns'].values.reshape(-1,1))
data.head()
```

	Month Starting	Open	High	Low	Close	Change %	Avg. Volume	Returns	Rolling Average
0	2022-12-01	101.38	102.59	100.67	101.28	-0.0017	21771536	NaN	NaN
1	2022-11-01	95.59	101.45	83.45	101.45	0.0717	28294944	0.185181	NaN
2	2022-10-03	97.22	105.10	91.90	94.66	-0.0155	27843110	-0.886997	NaN

```
data['Returns'] = data['Returns'].fillna(data['Returns'].mean())
data['Rolling Average'] = data['Rolling Average'].fillna(data['Rolling Average'].mean())
```

```
from sklearn.ensemble import IsolationForest
model = IsolationForest(contamination=0.05)
model.fit(data[['Returns']])

# Predicting anomalies
data['Anomaly'] = model.predict(data[['Returns']])
data['Anomaly'] = data['Anomaly'].map({1: 0, -1: 1})

# Ploting the results
plt.figure(figsize=(13,5))
plt.plot(data.index, data['Returns'], label='Returns')
plt.scatter(data[data['Anomaly'] == 1].index, data[data['Anomaly'] == 1]['Returns'], color='red')
plt.legend(['Returns', 'Anomaly'])
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does n
  warnings.warn(
```



✓ 2s completed at 3:44 PM