```
!pip install yfinance
```

```
Requirement already satisfied: yfinance in /usr/local/lib/python3.11/dist-packages (0.2.56)
Requirement already satisfied: pandas>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from yfinance) (2.2.2)
Requirement already satisfied: numpy>=1.16.5 in /usr/local/lib/python3.11/dist-packages (from yfinance) (2.0.2)
Requirement already satisfied: requests>=2.31 in /usr/local/lib/python3.11/dist-packages (from yfinance) (2.32.3)
Requirement already satisfied: multitasking>=0.0.7 in /usr/local/lib/python3.11/dist-packages (from yfinance) (0.0.11)
Requirement already satisfied: platformdirs>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from yfinance) (4.3.7)
Requirement already satisfied: pytz>=2022.5 in /usr/local/lib/python3.11/dist-packages (from yfinance) (2025.2)
Requirement already satisfied: frozendict>=2.3.4 in /usr/local/lib/python3.11/dist-packages (from yfinance) (2.4.6)
Requirement already satisfied: peewee>=3.16.2 in /usr/local/lib/python3.11/dist-packages (from yfinance) (3.17.9)
Requirement already satisfied: beautifulsoup4>=4.11.1 in /usr/local/lib/python3.11/dist-packages (from yfinance) (4.13.4)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.11/dist-packages (from beautifulsoup4>=4.11.1->yfinance) (2.7)
Requirement already satisfied: typing-extensions>=4.0.0 in /usr/local/lib/python3.11/dist-packages (from beautifulsoup4>=4.11.1->yfinance) (4.13.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.3.0->yfinance) (2.9.0.post0)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.3.0->yfinance) (2025.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests>=2.31->yfinance) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests>=2.31->yfinance) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests>=2.31->yfinance) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests>=2.31->yfinance) (2025.1.31)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas>=1.3.0->yfinance) (1.17.0)
```

```
import pandas as pd
import yfinance as yf
import plotly.io as pio
import plotly.graph_objects as go
pio.templates.default ="plotly_white"
```

```
tickers={
    "apple_ticker":"AAPL",
    "google_ticker":"GOOGL",
    "facebook_ticker":"META",
    "intel_ticker":"INTC",
    "microsoft_ticker":"MSFT",
    "reliance_ticker":"RELIANCE.NS",
    "Tesla_ticker":"TSLA",
    "Amazon_ticker":"AMZN"
}
start_date="2024-10-01"
end_date="2024-12-31"

apple_data=yf.download(tickers["apple_ticker"],start=start_date, end=end_date)
google_data=yf.download(tickers["google_ticker"],start=start_date, end=end_date)
facebook_data=yf.download(tickers["facebook_ticker"],start=start_date, end=end_date)
Intel_data=yf.download(tickers["intel_ticker"],start=start_date, end=end_date)
```

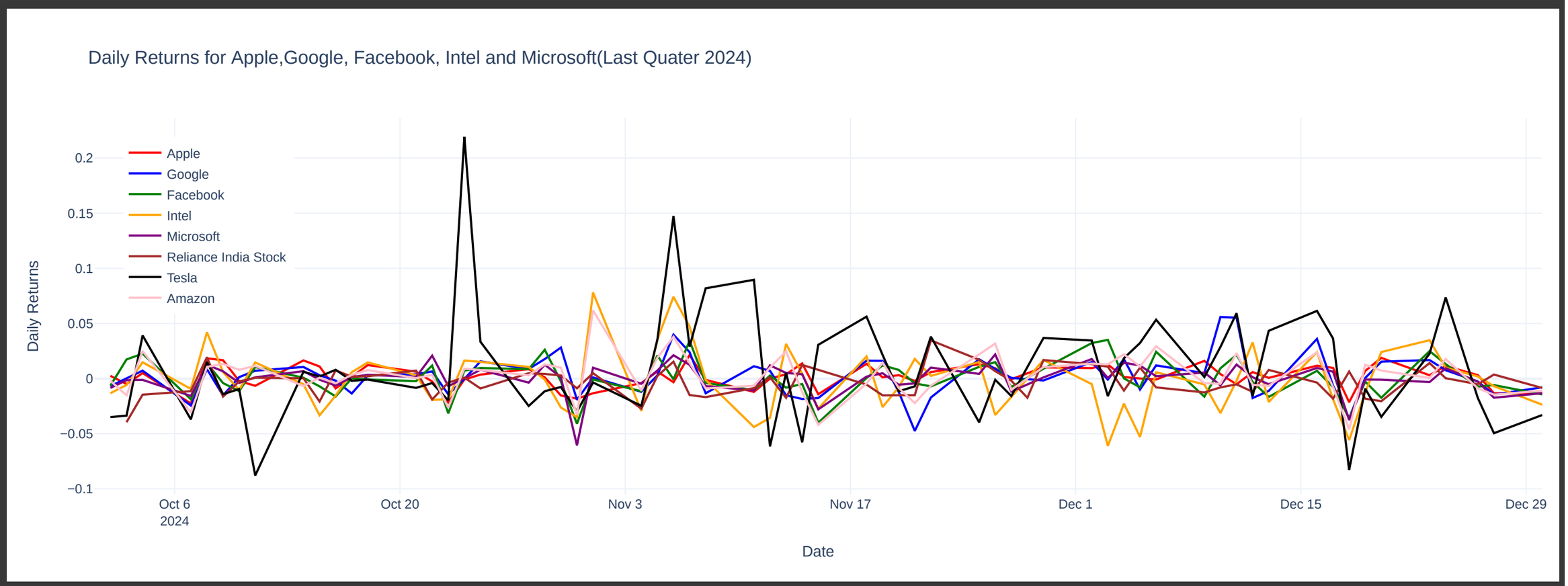
```
Intel_data=yf.download(tickers["Intel_ticker"],start=start_date, end=end_date)
microsoft_data=yf.download(tickers["microsoft_ticker"],start=start_date, end=end_date)
reliance_data=yf.download(tickers["reliance_ticker"],start=start_date, end=end_date)
tesla_data=yf.download(tickers["Tesla_ticker"],start=start_date, end=end_date)
amazon_data=yf.download(tickers["Amazon_ticker"],start=start_date, end=end_date)
```

```
YF.download() has changed argument auto_adjust default to True
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
```

```
apple_data['Daily_Return'] = apple_data['Close'].pct_change()
google_data['Daily_Return'] = google_data['Close'].pct_change()
facebook_data['Daily Return']=facebook_data['Close'].pct_change()
Intel_data['Daily Return']=Intel_data['Close'].pct_change()
microsoft_data['Daily Return']=microsoft_data['Close'].pct_change()
reliance_data['Daily Return']=reliance_data['Close'].pct_change()
tesla_data['Daily Return']=tesla_data['Close'].pct_change()
amazon_data['Daily Return']=amazon_data['Close'].pct_change()
```

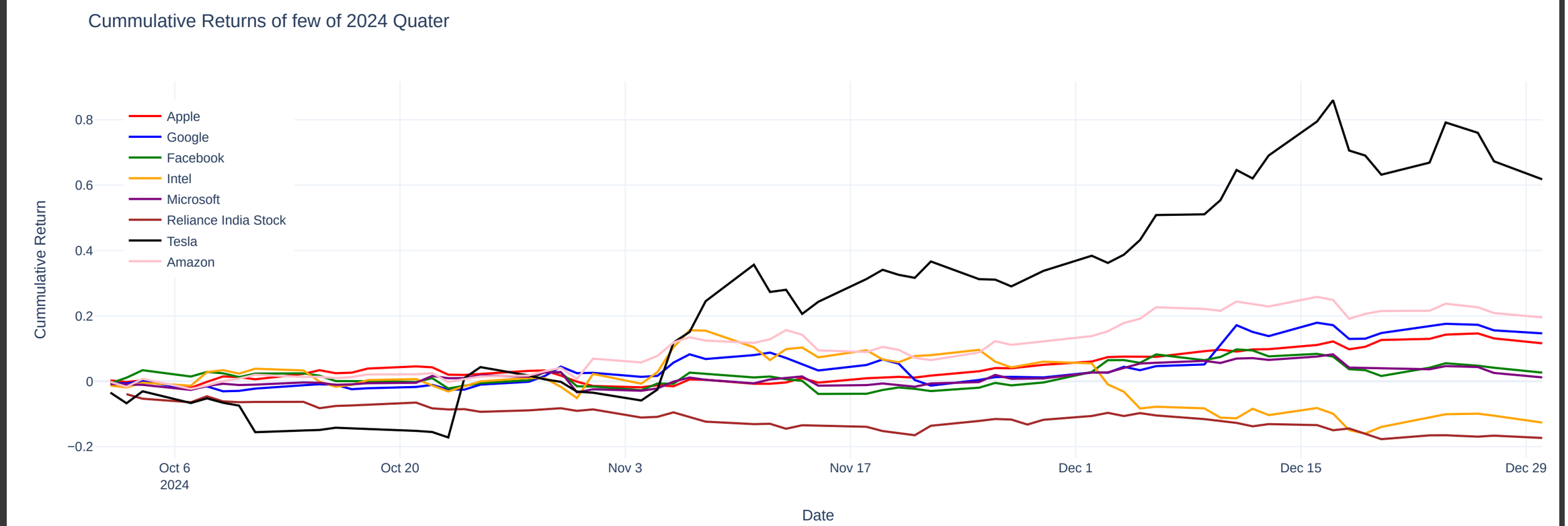
```
fig=go.Figure()

fig.add_trace(go.Scatter(x=apple_data.index,y=apple_data['Daily_Return'],mode='lines',name='Apple',
                         line=dict(color='red',width=2)))

fig.add_trace(go.Scatter(x=google_data.index,y=google_data['Daily_Return'],mode='lines',name='Google',
                         line=dict(color='blue',width=2)))
fig.add_trace(go.Scatter(x=facebook_data.index,y=facebook_data['Daily Return'],mode='lines',name='Facebook',
                         line=dict(color='green',width=2)))
fig.add_trace(go.Scatter(x=Intel_data.index,y=Intel_data['Daily Return'],mode='lines',name='Intel',
                         line=dict(color='orange',width=2)))
fig.add_trace(go.Scatter(x=microsoft_data.index,y=microsoft_data['Daily Return'],mode='lines',name='Microsoft',
                         line=dict(color='purple',width=2)))
fig.add_trace(go.Scatter(x=reliance_data.index,y=reliance_data['Daily Return'], mode='lines',name='Reliance India Stock',
                         line=dict(color='brown',width=2)))
fig.add_trace(go.Scatter(x=tesla_data.index,y=tesla_data['Daily Return'],mode='lines',name='Tesla',
                         line=dict(color='black',width=2)))
fig.add_trace(go.Scatter(x=amazon_data.index,y=amazon_data['Daily Return'],mode='lines',name='Amazon',
                         line=dict(color='pink',width=2)))

fig.update_layout(title='Daily Return of Apple, Google, Facebook, Intel and Microsoft (Last 3 years 2021)',xaxis_title='Date',yaxis_
```

```
fig.update_layout(title='Daily Returns for Apple,Google, Facebook, Intel and Microsoft(Last Quater 2024)',xaxis_title='Date',yaxi
                  legend=dict(x=0.02,y=0.95))

fig.show()
```



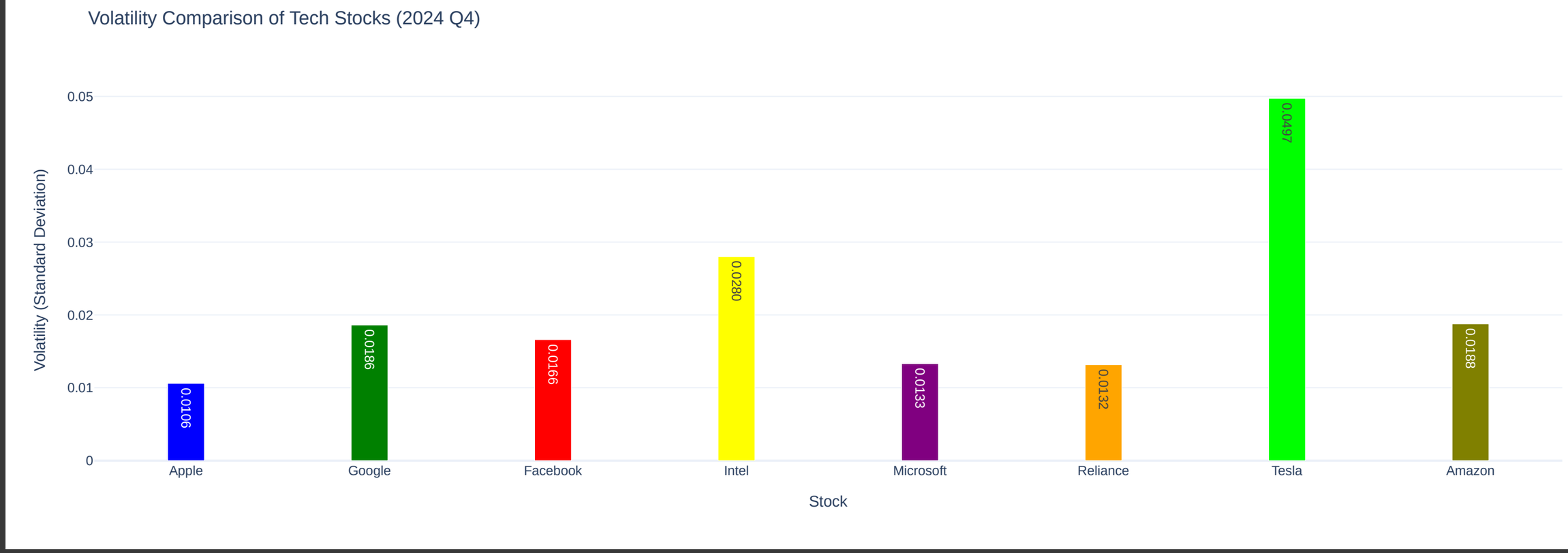

```
apple_cummulative_return=(1+apple_data['Daily_Return']).cumprod()-1
google_cummulative_return=(1+google_data['Daily_Return']).cumprod()-1
facebook_cummulative_return=(1+facebook_data['Daily Return']).cumprod()-1
Intel_cummulative_return=(1+Intel_data['Daily Return']).cumprod()-1
microsoft_cummulative_return=(1+microsoft_data['Daily Return']).cumprod()-1
reliance_cummulative_return=(1+reliance_data['Daily Return']).cumprod()-1
tesla_cummulative_return=(1+tesla_data['Daily Return']).cumprod()-1
amazon_cummulative_return=(1+amazon_data['Daily Return']).cumprod()-1

fig=go.Figure()

fig.add_trace(go.Scatter(x=apple_cummulative_return.index,y=apple_cummulative_return,
                         mode='lines',name='Apple',line=dict(color='red')))
fig.add_trace(go.Scatter(x=google_cummulative_return.index,y=google_cummulative_return,
                         mode='lines',name='Google',line=dict(color='blue')))
```

```
fig.add_trace(go.Scatter(x=facebook_cummulative_return.index,y=facebook_cummulative_return,
                         mode='lines',name='Facebook',line=dict(color='green')))
fig.add_trace(go.Scatter(x=Intel_cummulative_return.index,y=Intel_cummulative_return,
                         mode='lines',name='Intel',line=dict(color='orange')))
fig.add_trace(go.Scatter(x=microsoft_cummulative_return.index,y=microsoft_cummulative_return,
                         mode='lines',name='Microsoft',line=dict(color='purple')))
fig.add_trace(go.Scatter(x=reliance_cummulative_return.index,y=reliance_cummulative_return,
                         mode='lines',name='Reliance India Stock',line=dict(color='brown')))
fig.add_trace(go.Scatter(x=tesla_cummulative_return.index,y=tesla_cummulative_return,
                         mode='lines',name='Tesla',line=dict(color='black')))
fig.add_trace(go.Scatter(x=amazon_cummulative_return.index,y=amazon_cummulative_return,
                         mode='lines',name='Amazon',line=dict(color='pink')))

fig.update_layout(title='Cummulative Returns of few of 2024 Quater'
,xaxis_title='Date',yaxis_title='Cummulative Return', legend=dict(x=0.02,y=0.95))

fig.show()
```




```
apple_volatility=apple_data['Daily_Return'].std()
```

```
google_volatility=google_data['Daily_Return'].std()
facebook_volatility=facebook_data['Daily Return'].std()
Intel_volatility=Intel_data['Daily Return'].std()
microsoft_volatility=microsoft_data['Daily Return'].std()
reliance_volatility=reliance_data['Daily Return'].std()
tesla_volatility=tesla_data['Daily Return'].std()
amazon_volatility=amazon_data['Daily Return'].std()

data_dict = {
    'Apple': apple_volatility,
    'Google': google_volatility,
    'Facebook': facebook_volatility,
    'Intel': Intel_volatility,
    'Microsoft': microsoft_volatility,
    'Reliance': reliance_volatility,
    'Tesla': tesla_volatility,
    'Amazon': amazon_volatility
    # Add more like:
    # 'Tesla': tesla_data['Daily Return'].std(),
    # 'Amazon': amazon_data['Daily Return'].std()
}


colors = ['blue', 'green', 'red', 'yellow', 'purple', 'orange','lime','olive']

fig1 = go.Figure()
fig1.add_bar(
    x=list(data_dict.keys()),
    y=list(data_dict.values()),
    text=[f'{v:.4f}' for v in data_dict.values()],
    textposition='auto',
    marker=dict(color=colors[:len(data_dict)])
)

fig1.update_layout(
    title='Volatility Comparison of Tech Stocks (2024 Q4)',
    xaxis_title='Stock',
    yaxis_title='Volatility (Standard Deviation)',
    bargap=0.8
)

fig1.show()
```




```
market_data = yf.download('^GSPC', start=start_date, end=end_date)
market_data['Daily Return'] = market_data['Close'].pct_change()

# Create a dictionary to store the results
results = {}

# Loop through the tickers and calculate the beta and covariance
for company, ticker in tickers.items():
    # Download company data (if not already downloaded)
    if company.endswith("_ticker"):
        company_name = company[:-len("_ticker")]
        company_data = locals()[company_name.lower() + "_data" if company_name.lower() != "intel" else "Intel_data"]

        if 'Daily_Return' in company_data.columns and 'Daily Return' not in company_data.columns:
            company_data = company_data.rename(columns={'Daily_Return': 'Daily Return'})

        if 'Daily_Return' not in company_data.columns:
```

```
        if 'Daily Return' not in company_data.columns:
            company_data['Daily Return'] = company_data['Close'].pct_change()
    else:
        company_data = yf.download(ticker, start=start_date, end=end_date)
        company_data['Daily Return'] = company_data['Close'].pct_change()

    # Calculate covariance and beta
    cov = company_data['Daily Return'].cov(market_data['Daily Return'])
    beta = cov / market_data['Daily Return'].var()

    # Store the results
    results[company_name or company] = {'Covariance': cov, 'Beta': beta}  # Use company_name or company as key

# Display the results
for company, data in results.items():
    print(f"\n{company}:")
    print(f" Covariance with Market: {data['Covariance']:.6f}")
    print(f" Beta: {data['Beta']:.6f}\n")
    print("-" * 20)

# Find the company with the highest volatility (beta)
most_volatile_company = max(results, key=lambda company: results[company]['Beta'])

print(f"\n{most_volatile_company} has the highest volatility (beta) among the selected companies.")
```

```
[*********************100%***********************]  1 of 1 completed
apple:
 Covariance with Market: 0.000044
 Beta: 0.691758

--------------------

google:
 Covariance with Market: 0.000075
 Beta: 1.184915

--------------------

facebook:
 Covariance with Market: 0.000079
 Beta: 1.249585

--------------------

intel:
 Covariance with Market: 0.000135
 Beta: 2.138677

--------------------
```

```
microsoft:
 Covariance with Market: 0.000074
 Beta: 1.179948

--------------------

reliance:
 Covariance with Market: 0.000016
 Beta: 0.261628

--------------------

Tesla:
 Covariance with Market: 0.000188
 Beta: 2.986235

--------------------

Amazon:
 Covariance with Market: 0.000105
 Beta: 1.667240

--------------------

Tesla has the highest volatility (beta) among the selected companies.
```