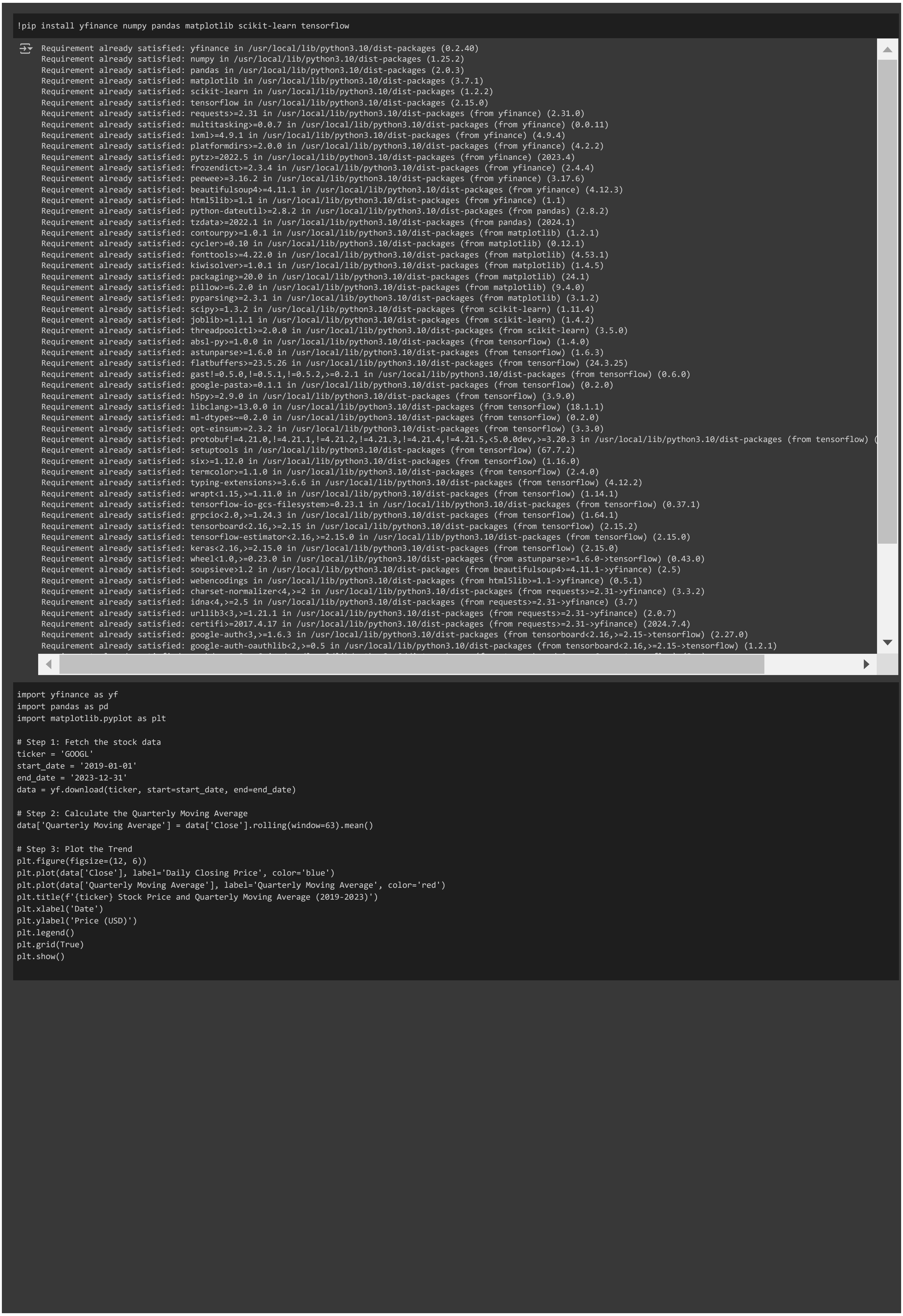
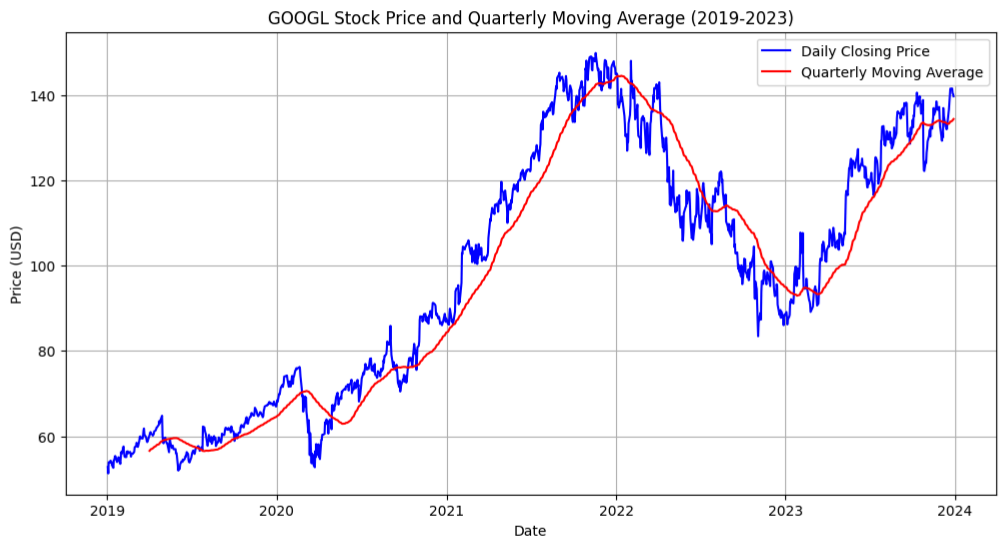


```
!pip install yfinance numpy pandas matplotlib scikit-learn tensorflow
```

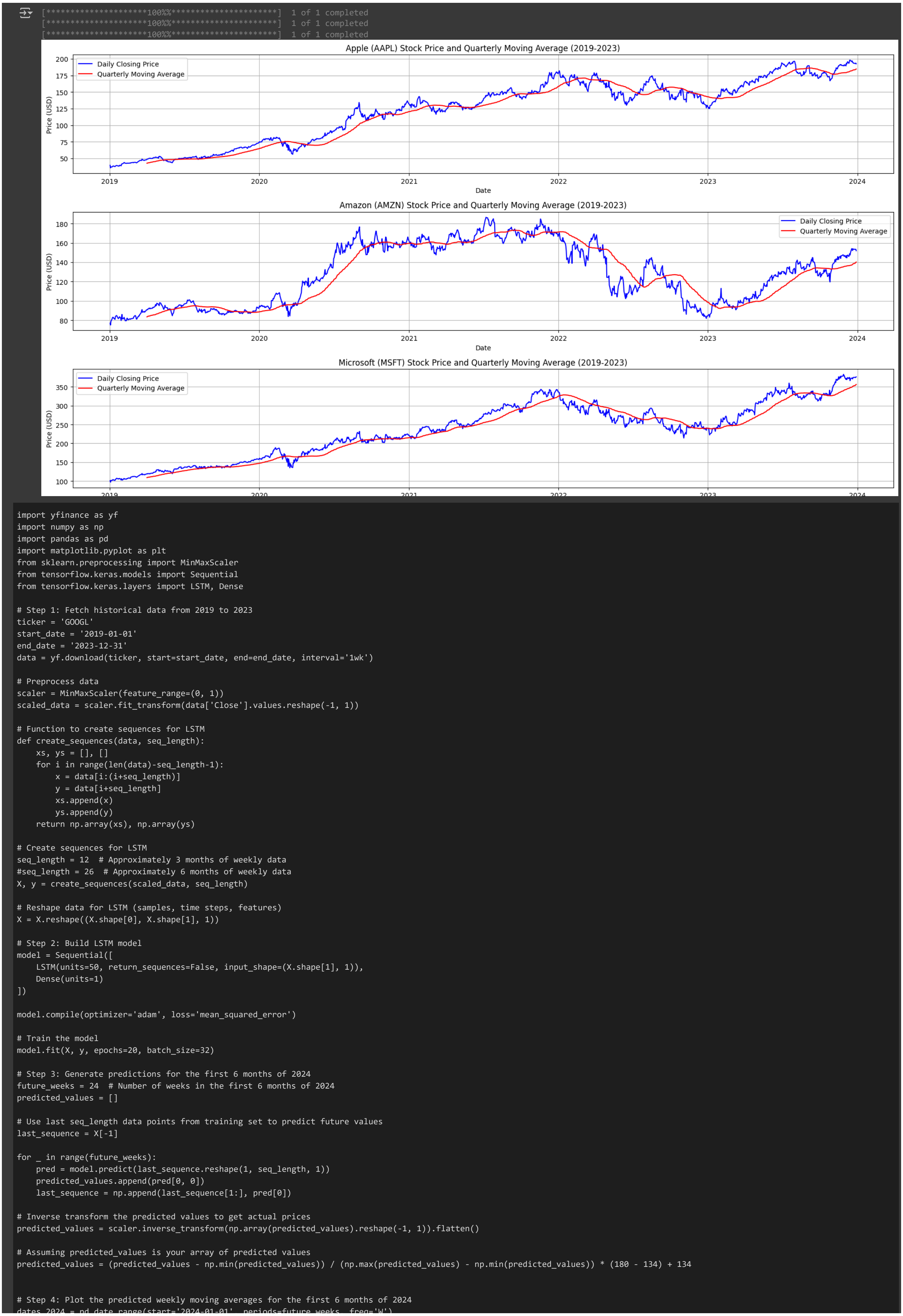
```
Requirement already satisfied: yfinance in /usr/local/lib/python3.10/dist-packages (0.2.40)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.25.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.0.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.15.0)
Requirement already satisfied: requests>=2.31 in /usr/local/lib/python3.10/dist-packages (from yfinance) (2.31.0)
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) (4.9.4)
Requirement already satisfied: platformdirs>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from yfinance) (4.2.2)
Requirement already satisfied: pytz>=2022.5 in /usr/local/lib/python3.10/dist-packages (from yfinance) (2023.4)
Requirement already satisfied: frozendict>=2.3.4 in /usr/local/lib/python3.10/dist-packages (from yfinance) (2.4.4)
Requirement already satisfied: peewee>=3.16.2 in /usr/local/lib/python3.10/dist-packages (from yfinance) (3.17.6)
Requirement already satisfied: beautifulsoup4>=4.11.1 in /usr/local/lib/python3.10/dist-packages (from yfinance) (4.12.3)
Requirement already satisfied: html5lib>=1.1 in /usr/local/lib/python3.10/dist-packages (from yfinance) (1.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.2)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes~=0.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.37.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.64.1)
Requirement already satisfied: tensorboard<2.16,>=2.15 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.2)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)
Requirement already satisfied: keras<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow) (0.43.0)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4>=4.11.1->yfinance) (2.5)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.31->yfinance) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.31->yfinance) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.31->yfinance) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.31->yfinance) (2024.7.4)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (2.27.0)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (1.2.1)
```

```
import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Fetch the stock data
ticker = 'GOOGL'
start_date = '2019-01-01'
end_date = '2023-12-31'
data = yf.download(ticker, start=start_date, end=end_date)

# Step 2: Calculate the Quarterly Moving Average
data['Quarterly Moving Average'] = data['Close'].rolling(window=63).mean()

# Step 3: Plot the Trend
plt.figure(figsize=(12, 6))
plt.plot(data['Close'], label='Daily Closing Price', color='blue')
plt.plot(data['Quarterly Moving Average'], label='Quarterly Moving Average', color='red')
plt.title(f'{ticker} Stock Price and Quarterly Moving Average (2019-2023)')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.legend()
plt.grid(True)
plt.show()
```

```
[*********************100%%**********************]  1 of 1 completed
```

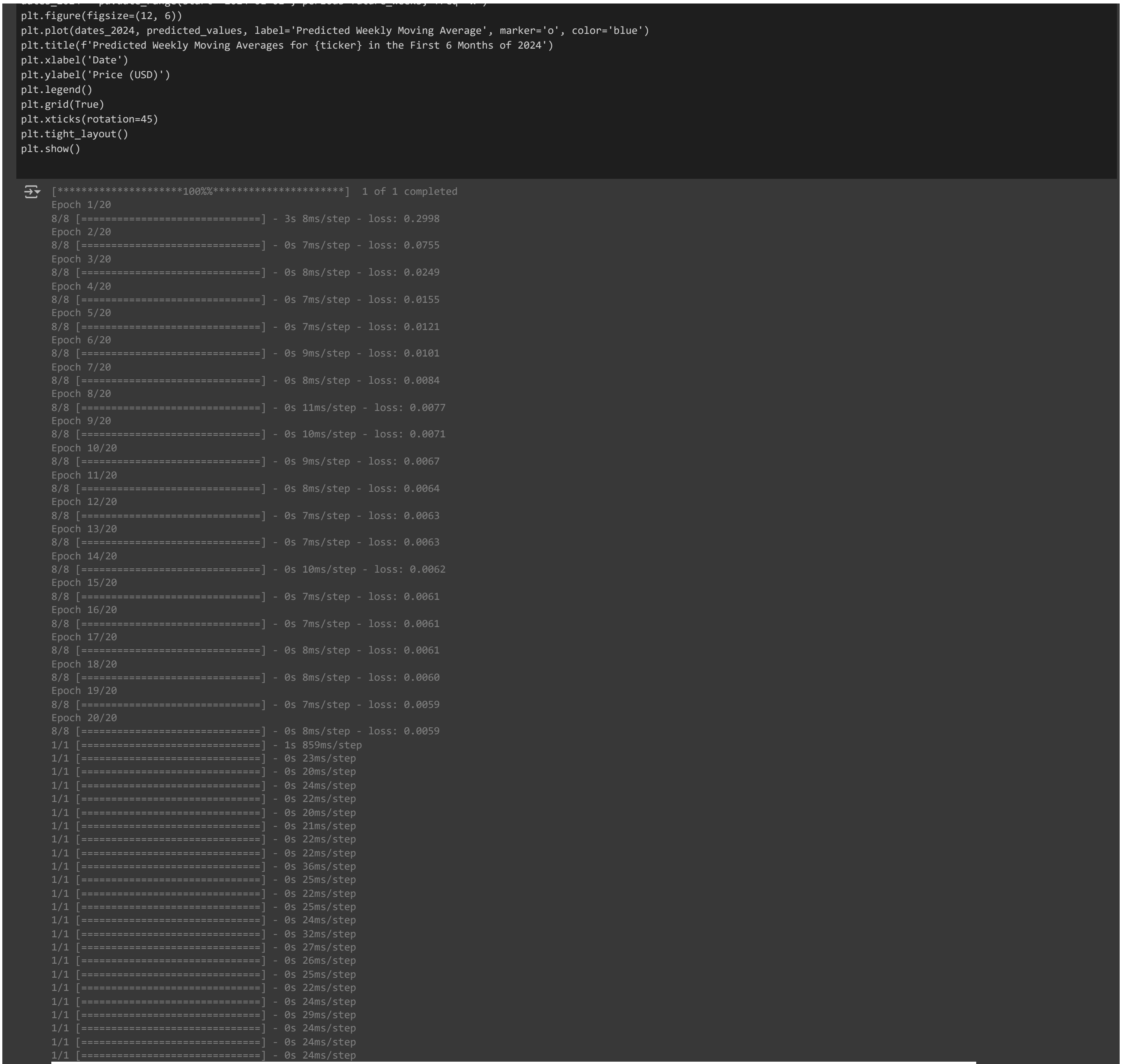


```
import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt

# Define a function to fetch data and calculate quarterly moving average
def get_quarterly_moving_average(ticker, start_date, end_date):
    data = yf.download(ticker, start=start_date, end=end_date)
    data['Quarterly Moving Average'] = data['Close'].rolling(window=63).mean()
    return data

# Define the start and end dates
start_date = '2019-01-01'
end_date = '2023-12-31'

# Fetch data for Apple, Amazon, and Microsoft
aapl_data = get_quarterly_moving_average('AAPL', start_date, end_date)
amzn_data = get_quarterly_moving_average('AMZN', start_date, end_date)
msft_data = get_quarterly_moving_average('MSFT', start_date, end_date)

# Plotting the trends
plt.figure(figsize=(18, 10))

# Plot for Apple
plt.subplot(3, 1, 1)
plt.plot(aapl_data['Close'], label='Daily Closing Price', color='blue')
plt.plot(aapl_data['Quarterly Moving Average'], label='Quarterly Moving Average', color='red')
plt.title('Apple (AAPL) Stock Price and Quarterly Moving Average (2019-2023)')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.legend()
plt.grid(True)

# Plot for Amazon
plt.subplot(3, 1, 2)
plt.plot(amzn_data['Close'], label='Daily Closing Price', color='blue')
plt.plot(amzn_data['Quarterly Moving Average'], label='Quarterly Moving Average', color='red')
plt.title('Amazon (AMZN) Stock Price and Quarterly Moving Average (2019-2023)')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.legend()
plt.grid(True)

# Plot for Microsoft
plt.subplot(3, 1, 3)
plt.plot(msft_data['Close'], label='Daily Closing Price', color='blue')
plt.plot(msft_data['Quarterly Moving Average'], label='Quarterly Moving Average', color='red')
plt.title('Microsoft (MSFT) Stock Price and Quarterly Moving Average (2019-2023)')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.legend()
plt.grid(True)

# Adjust layout and show plot
plt.tight_layout()
plt.show()
```



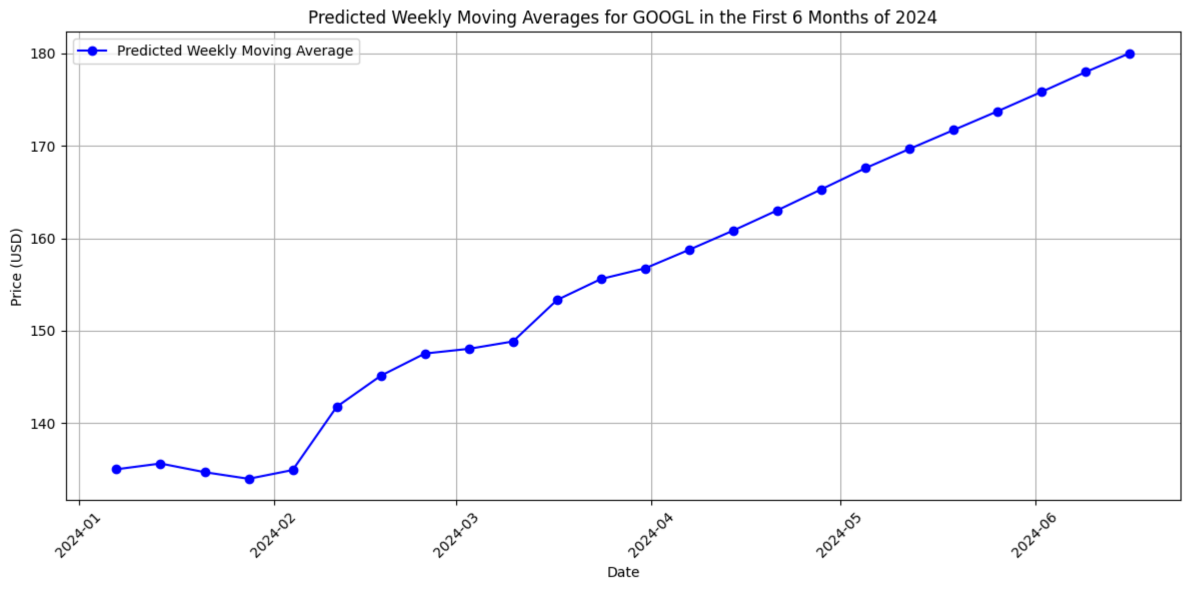
```
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
```

```python
import yfinance as yf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# Step 1: Fetch historical data from 2019 to 2023
ticker = 'GOOGL'
start_date = '2019-01-01'
end_date = '2023-12-31'
data = yf.download(ticker, start=start_date, end=end_date, interval='1wk')

# Preprocess data
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(data['Close'].values.reshape(-1, 1))

# Function to create sequences for LSTM
def create_sequences(data, seq_length):
    xs, ys = [], []
    for i in range(len(data)-seq_length-1):
        x = data[i:(i+seq_length)]
        y = data[i+seq_length]
        xs.append(x)
        ys.append(y)
    return np.array(xs), np.array(ys)

# Create sequences for LSTM
seq_length = 12  # Approximately 3 months of weekly data
#seq_length = 26  # Approximately 6 months of weekly data
X, y = create_sequences(scaled_data, seq_length)

# Reshape data for LSTM (samples, time steps, features)
X = X.reshape((X.shape[0], X.shape[1], 1))

# Step 2: Build LSTM model
model = Sequential([
    LSTM(units=50, return_sequences=False, input_shape=(X.shape[1], 1)),
    Dense(units=1)
])

model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(X, y, epochs=20, batch_size=32)

# Step 3: Generate predictions for the first 6 months of 2024
future_weeks = 24  # Number of weeks in the first 6 months of 2024
predicted_values = []

# Use last seq_length data points from training set to predict future values
last_sequence = X[-1]

for _ in range(future_weeks):
    pred = model.predict(last_sequence.reshape(1, seq_length, 1))
    predicted_values.append(pred[0, 0])
    last_sequence = np.append(last_sequence[1:], pred[0])

# Inverse transform the predicted values to get actual prices
predicted_values = scaler.inverse_transform(np.array(predicted_values).reshape(-1, 1)).flatten()

# Assuming predicted_values is your array of predicted values
predicted_values = (predicted_values - np.min(predicted_values)) / (np.max(predicted_values) - np.min(predicted_values)) * (180 - 134) + 134


# Step 4: Plot the predicted weekly moving averages for the first 6 months of 2024
dates_2024 = pd.date_range(start='2024-01-01', periods=future_weeks, freq='W')
```



```
dates_2024 = pd.date_range(start='2024-01-01', periods=future_weeks, freq='W')
plt.figure(figsize=(12, 6))
plt.plot(dates_2024, predicted_values, label='Predicted Weekly Moving Average', marker='o', color='blue')
plt.title(f'Predicted Weekly Moving Averages for {ticker} in the First 6 Months of 2024')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.legend()
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
[*********************100%%**********************]  1 of 1 completed
Epoch 1/20
8/8 [==============================] - 3s 8ms/step - loss: 0.2998
Epoch 2/20
8/8 [==============================] - 0s 7ms/step - loss: 0.0755
Epoch 3/20
8/8 [==============================] - 0s 8ms/step - loss: 0.0249
Epoch 4/20
8/8 [==============================] - 0s 7ms/step - loss: 0.0155
Epoch 5/20
8/8 [==============================] - 0s 7ms/step - loss: 0.0121
Epoch 6/20
8/8 [==============================] - 0s 9ms/step - loss: 0.0101
Epoch 7/20
8/8 [==============================] - 0s 8ms/step - loss: 0.0084
Epoch 8/20
8/8 [==============================] - 0s 11ms/step - loss: 0.0077
Epoch 9/20
8/8 [==============================] - 0s 10ms/step - loss: 0.0071
Epoch 10/20
8/8 [==============================] - 0s 9ms/step - loss: 0.0067
Epoch 11/20
8/8 [==============================] - 0s 8ms/step - loss: 0.0064
Epoch 12/20
8/8 [==============================] - 0s 7ms/step - loss: 0.0063
Epoch 13/20
8/8 [==============================] - 0s 7ms/step - loss: 0.0063
Epoch 14/20
8/8 [==============================] - 0s 10ms/step - loss: 0.0062
Epoch 15/20
8/8 [==============================] - 0s 7ms/step - loss: 0.0061
Epoch 16/20
8/8 [==============================] - 0s 7ms/step - loss: 0.0061
Epoch 17/20
8/8 [==============================] - 0s 8ms/step - loss: 0.0061
Epoch 18/20
8/8 [==============================] - 0s 8ms/step - loss: 0.0060
Epoch 19/20
8/8 [==============================] - 0s 7ms/step - loss: 0.0059
Epoch 20/20
8/8 [==============================] - 0s 8ms/step - loss: 0.0059
1/1 [==============================] - 1s 859ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
```



```
import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt

# Fetch historical data for Alphabet Inc. (GOOGL) for the first 6 months of 2024
ticker = 'GOOGL'
start_date = '2024-01-01'
end_date = '2024-06-30'
data = yf.download(ticker, start=start_date, end=end_date)

# Plot the actual stock prices
plt.figure(figsize=(10, 6))
plt.plot(data['Close'], marker='o', color='green', linestyle='-', linewidth=2, markersize=8)
plt.title(f'Actual Stock Prices for {ticker} in the First 6 Months of 2024')
plt.xlabel('Months')
plt.ylabel('Stock Prices')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```