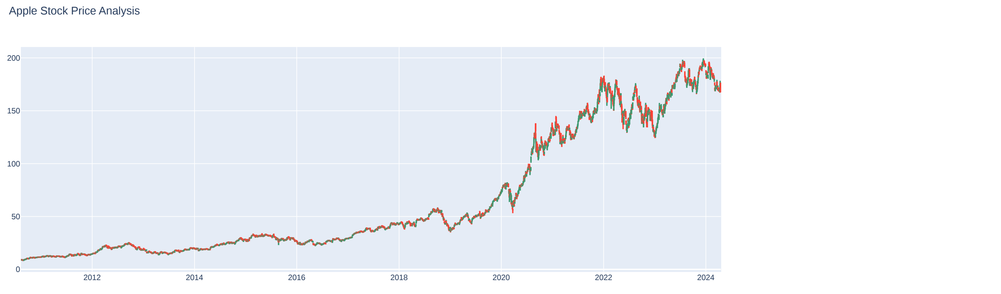
In [3]:
```
# Import Needed Liberaries:
import pandas as pd
import yfinance as yf
from datetime import date, timedelta
```

In [5]:
```
# Define end date as today
today = date.today()
end_date = today.strftime("%Y-%m-%d")

# Define start date as 5000 days before today
start_date = today - timedelta(days=5000)
start_date = start_date.strftime("%Y-%m-%d")

# Download stock data for AAPL
data = yf.download('AAPL', start=start_date, end=end_date, progress=False)

# Add 'Date' column to the DataFrame
data["Date"] = data.index

# Reorder columns
data = data[["Date", "Open", "High", "Low", "Close", "Adj Close", "Volume"]]

# Reset index
data.reset_index(drop=True, inplace=True)

# Display the last few rows of the DataFrame
print(data.tail())
```

```
           Date        Open        High         Low       Close   Adj Close  \
3440 2024-04-11  168.339996  175.460007  168.160004  175.039993  175.039993
3441 2024-04-12  174.259995  178.360001  174.210007  176.550003  176.550003
3442 2024-04-15  175.360001  176.630005  172.500000  172.690002  172.690002
3443 2024-04-16  171.750000  173.759995  168.270004  169.380005  169.380005
3444 2024-04-17  169.610001  170.649994  168.000000  168.000000  168.000000

        Volume
3440   91070300
3441  101593300
3442   73531800
3443   73711200
3444   50843400
```

In [6]:
```
import plotly.graph_objects as go

# Create a candlestick chart using Plotly
figure = go.Figure(data=[go.Candlestick(x=data["Date"],
                    open=data["Open"],
                    high=data["High"],
                    low=data["Low"],
                    close=data["Close"])])

# Update layout with title and hide range slider
figure.update_layout(title="Apple Stock Price Analysis",
                     xaxis_rangeslider_visible=False)

# Show the figure
figure.show()
```




In [7]:
```
correlation = data.corr()
print(correlation["Close"].sort_values(ascending=False))
```

```
Close        1.000000
Adj Close    0.999957
Low          0.999893
High         0.999889
Open         0.999767
Date         0.895984
Volume      -0.531932
Name: Close, dtype: float64
```

In [8]:
```
from sklearn.model_selection import train_test_split

# Define features (x) and target variable (y)
x = data[["Open", "High", "Low", "Volume"]].to_numpy()
y = data["Close"].to_numpy().reshape(-1, 1)

# Split the data into training and testing sets
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, random_state=42)
```

In [10]:
```
from keras.models import Sequential
from keras.layers import Dense, LSTM

# Define the model
model = Sequential()

# Add LSTM layers
model.add(LSTM(128, return_sequences=True, input_shape=(xtrain.shape[1], 1)))
model.add(LSTM(64, return_sequences=False))

# Add dense layers
model.add(Dense(25))
model.add(Dense(1))

# Print model summary
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 4, 128)            66560

 lstm_1 (LSTM)               (None, 64)                49408

 dense (Dense)               (None, 25)                1625

 dense_1 (Dense)             (None, 1)                 26

=================================================================
Total params: 117,619
Trainable params: 117,619
Non-trainable params: 0
_________________________________________________________________
```

In [11]:
```
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(xtrain, ytrain, batch_size=1, epochs=30)
```

```
Epoch 1/30
2756/2756 [==============================] - 27s 5ms/step - loss: 369.8870
Epoch 2/30
2756/2756 [==============================] - 18s 7ms/step - loss: 16.8737
Epoch 3/30
2756/2756 [==============================] - 16s 6ms/step - loss: 12.7744
Epoch 4/30
2756/2756 [==============================] - 14s 5ms/step - loss: 12.2676
Epoch 5/30
2756/2756 [==============================] - 14s 5ms/step - loss: 9.4833
Epoch 6/30
2756/2756 [==============================] - 14s 5ms/step - loss: 8.3336
Epoch 7/30
2756/2756 [==============================] - 14s 5ms/step - loss: 8.8354
Epoch 8/30
2756/2756 [==============================] - 15s 5ms/step - loss: 7.7743
Epoch 9/30
2756/2756 [==============================] - 15s 5ms/step - loss: 7.7091
Epoch 10/30
2756/2756 [==============================] - 15s 5ms/step - loss: 6.3113
Epoch 11/30
2756/2756 [==============================] - 15s 5ms/step - loss: 7.0718
Epoch 12/30
2756/2756 [==============================] - 14s 5ms/step - loss: 7.0340
Epoch 13/30
2756/2756 [==============================] - 15s 5ms/step - loss: 8.0890
Epoch 14/30
2756/2756 [==============================] - 14s 5ms/step - loss: 5.1880
Epoch 15/30
2756/2756 [==============================] - 15s 5ms/step - loss: 6.0934
Epoch 16/30
2756/2756 [==============================] - 15s 5ms/step - loss: 5.1978
Epoch 17/30
2756/2756 [==============================] - 15s 5ms/step - loss: 6.7035
Epoch 18/30
2756/2756 [==============================] - 15s 5ms/step - loss: 4.6020
Epoch 19/30
2756/2756 [==============================] - 15s 5ms/step - loss: 5.0566
Epoch 20/30
2756/2756 [==============================] - 15s 6ms/step - loss: 4.3835
Epoch 21/30
2756/2756 [==============================] - 16s 6ms/step - loss: 6.2698
Epoch 22/30
2756/2756 [==============================] - 15s 5ms/step - loss: 3.9047
Epoch 23/30
2756/2756 [==============================] - 15s 5ms/step - loss: 5.2889
Epoch 24/30
2756/2756 [==============================] - 15s 6ms/step - loss: 4.4810
Epoch 25/30
2756/2756 [==============================] - 15s 5ms/step - loss: 4.5561
Epoch 26/30
2756/2756 [==============================] - 15s 5ms/step - loss: 4.8701
Epoch 27/30
2756/2756 [==============================] - 15s 5ms/step - loss: 4.3413
Epoch 28/30
2756/2756 [==============================] - 15s 5ms/step - loss: 4.2482
Epoch 29/30
2756/2756 [==============================] - 15s 6ms/step - loss: 5.3666
Epoch 30/30
2756/2756 [==============================] - 15s 5ms/step - loss: 4.2227
```

Out[11]:
```
<keras.callbacks.History at 0x203145b18b0>
```

In [14]:
```
import numpy as np

# Define your features in the correct format
features = np.array([[177.070007, 180.419998, 177.089996, 179.0, 74919600]])

# Extract the relevant features for prediction
features_relevant = features[:, :-1]  # Exclude the last column 'Volume'

# Reshape the features to match the input shape expected by your model
features_reshaped = np.expand_dims(features_relevant, axis=-1)

# Predict using your model
prediction = model.predict(features_reshaped)
```

```
# Print the prediction
print("Predicted closing price:", prediction)
```

```
1/1 [==============================] - 2s 2s/step
Predicted closing price: [[5.187636]]
```