Extracting and Visualizing Stock Data

Description

Extracting essential data from a dataset and displaying it is a necessary part of data science; therefore individuals can make correct decisions based on the data. In this assignment, you will extract some stock data, you will then display this data in a graph.

Table of Contents

Estimated Time Needed: **30 min**

__Note__:- If you are working Locally using anaconda, please uncomment the following code and execute it.

In [2]:
```
#!pip install yfinance==0.2.38
#!pip install pandas==2.2.2
#!pip install nbformat
```

In [3]:
```
!pip install yfinance
!pip install bs4
!pip install nbformat
```

```
Collecting yfinance
  Downloading yfinance-0.2.46-py2.py3-none-any.whl.metadata (13 kB)
Requirement already satisfied: pandas>=1.3.0 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from yfinance) (2.1.4)
Requirement already satisfied: numpy>=1.16.5 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from yfinance) (1.26.4)
Requirement already satisfied: requests>=2.31 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from yfinance) (2.32.2)
Collecting multitasking>=0.0.7 (from yfinance)
  Downloading multitasking-0.0.11-py3-none-any.whl.metadata (5.5 kB)
Requirement already satisfied: lxml>=4.9.1 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from yfinance) (4.9.3)
Requirement already satisfied: platformdirs>=2.0.0 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from yfinance) (3.10.0)
Requirement already satisfied: pytz>=2022.5 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from yfinance) (2024.1)
Collecting frozendict>=2.3.4 (from yfinance)
  Downloading frozendict-2.4.6-py311-none-any.whl.metadata (23 kB)
Collecting peewee>=3.16.2 (from yfinance)
  Downloading peewee-3.17.7.tar.gz (939 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 939.5/939.5 kB 56.9 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: beautifulsoup4>=4.11.1 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from yfinance) (4.12.3)
Collecting html5lib>=1.1 (from yfinance)
  Downloading html5lib-1.1-py2.py3-none-any.whl.metadata (16 kB)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from beautifulsoup4>=4.11.1->yfinance) (2.5)
Requirement already satisfied: six>=1.9 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from html5lib>=1.1->yfinance) (1.16.0)
Collecting webencodings (from html5lib>=1.1->yfinance)
  Downloading webencodings-0.5.1-py2.py3-none-any.whl.metadata (2.1 kB)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from pandas>=1.3.0->yfinance) (2.8.2)
Requirement already satisfied: tzdata>=2022.1 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from pandas>=1.3.0->yfinance) (2023.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from requests>=2.31->yfinance) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from requests>=2.31->yfinance) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from requests>=2.31->yfinance) (1.26.19)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from requests>=2.31->yfinance) (2024.8.30)
Downloading yfinance-0.2.46-py2.py3-none-any.whl (100 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 101.0/101.0 kB 26.7 MB/s eta 0:00:00
Downloading frozendict-2.4.6-py311-none-any.whl (16 kB)
Downloading html5lib-1.1-py2.py3-none-any.whl (112 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 112.2/112.2 kB 32.1 MB/s eta 0:00:00
Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Downloading webencodings-0.5.1-py2.py3-none-any.whl (11 kB)
Building wheels for collected packages: peewee
  Building wheel for peewee (pyproject.toml) ... done
  Created wheel for peewee: filename=peewee-3.17.7-cp311-cp311-linux_x86_64.whl size=300129 sha256=7b42f7ede75b4b182c78c3dd5c4b90198884119dc089e77a5dcd4507cb91
6067
  Stored in directory: /tmp/wsuser/.cache/pip/wheels/fd/28/34/9ba1363b76703fe35ae8296af28ea74578a41b83544bb9da65
Successfully built peewee
Installing collected packages: webencodings, peewee, multitasking, html5lib, frozendict, yfinance
Successfully installed frozendict-2.4.6 html5lib-1.1 multitasking-0.0.11 peewee-3.17.7 webencodings-0.5.1 yfinance-0.2.46
Collecting bs4
  Downloading bs4-0.0.2-py2.py3-none-any.whl.metadata (411 bytes)
Requirement already satisfied: beautifulsoup4 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from bs4) (4.12.3)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from beautifulsoup4->bs4) (2.5)
Downloading bs4-0.0.2-py2.py3-none-any.whl (1.2 kB)
Installing collected packages: bs4
Successfully installed bs4-0.0.2
```

```
Requirement already satisfied: nbformat in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (5.9.2)
Requirement already satisfied: fastjsonschema in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from nbformat) (2.16.2)
Requirement already satisfied: jsonschema>=2.6 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from nbformat) (4.19.2)
Requirement already satisfied: jupyter-core in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from nbformat) (5.5.0)
Requirement already satisfied: traitlets>=5.1 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from nbformat) (5.7.1)
Requirement already satisfied: attrs>=22.2.0 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from jsonschema>=2.6->nbformat) (23.1.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from jsonschema>=2.6->nbform
at) (2023.7.1)
Requirement already satisfied: referencing>=0.28.4 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from jsonschema>=2.6->nbformat) (0.30.2)
Requirement already satisfied: rpds-py>=0.7.1 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from jsonschema>=2.6->nbformat) (0.10.6)
Requirement already satisfied: platformdirs>=2.5 in /opt/conda/envs/Python-RT24.1/lib/python3.11/site-packages (from jupyter-core->nbformat) (3.10.0)
```

In [5]:
```
import yfinance as yf
import pandas as pd
import requests
from bs4 import BeautifulSoup
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

In Python, you can ignore warnings using the warnings module. You can use the filterwarnings function to filter or ignore specific warning messages or categories.

In [6]:
```
import warnings
# Ignore all warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

Define Graphing Function

In this section, we define the function `make_graph`. **You don't have to know how the function works, you should only care about the inputs. It takes a dataframe with stock data (dataframe must contain Date and Close columns), a dataframe with revenue data (dataframe must contain Date and Revenue columns), and the name of the stock.**

In [7]:
```
def make_graph(stock_data, revenue_data, stock):
    fig = make_subplots(rows=2, cols=1, shared_xaxes=True, subplot_titles=("Historical Share Price", "Historical Revenue"), vertical_spacing = .3)
    stock_data_specific = stock_data[stock_data.Date <= '2021--06-14']
    revenue_data_specific = revenue_data[revenue_data.Date <= '2021-04-30']
    fig.add_trace(go.Scatter(x=pd.to_datetime(stock_data_specific.Date), y=stock_data_specific.Close.astype("float"), name="Share Price"), row=1, col=1)
    fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific.Date), y=revenue_data_specific.Revenue.astype("float"), name="Revenue"), row=2, col=1)
    fig.update_xaxes(title_text="Date", row=1, col=1)
    fig.update_xaxes(title_text="Date", row=2, col=1)
    fig.update_yaxes(title_text="Price ($US)", row=1, col=1)
    fig.update_yaxes(title_text="Revenue ($US Millions)", row=2, col=1)
    fig.update_layout(showlegend=False,
    height=900,
    title=stock,
    xaxis_rangeslider_visible=True)
    fig.show()
```

Use the make_graph function that we've already defined. You'll need to invoke it in questions 5 and 6 to display the graphs and create the dashboard.

> **Note: You don't need to redefine the function for plotting graphs anywhere else in this notebook; just use the existing function.**

Question 1: Use yfinance to Extract Stock Data

Using the `Ticker` function enter the ticker symbol of the stock we want to extract data on to create a ticker object. The stock is Tesla and its ticker symbol is `TSLA`.

In [8]:
```
import yfinance as yf

# Create a Ticker object for Tesla using the symbol "TSLA"
tesla = yf.Ticker("TSLA")

# we can now access various data on Tesla, such as its stock price history
print(tesla.info)  # To get basic information about Tesla
```

{'address1': '1 Tesla Road', 'city': 'Austin', 'state': 'TX', 'zip': '78725', 'country': 'United States', 'phone': '512 516 8177', 'website': 'https://www.tesla.com', 'industry': 'Auto Manufacturers', 'industryKey': 'auto-manufacturers', 'industryDisp': 'Auto Manufacturers', 'sector': 'Consumer Cyclical', 'sectorKey': 'consumer-cyclical', 'sectorDisp': 'Consumer Cyclical', 'longBusinessSummary': 'Tesla, Inc. designs, develops, manufactures, leases, and sells electric vehicles, and energy generation and storage systems in the United States, China, and internationally. The company operates in two segments, Automotive, and Energy Generation and Storage. The Automotive segment offers electric vehicles, as well as sells automotive regulatory credits; and non-warranty after-sales vehicle, used vehicles, body shop and parts, supercharging, retail merchandise, and vehicle insurance services. This segment also provides sedans and sport utility vehicles through direct and used vehicle sales, a network of Tesla Superchargers, and in-app upgrades; purchase financing and leasing services; services for electric vehicles through its company-owned service locations and Tesla mobile service technicians; and vehicle limited warranties and extended service plans. The Energy Generation and Storage segment engages in the design, manufacture, installation, sale, and leasing of solar energy generation and energy storage products, and related services to residential, commercial, and industrial customers and utilities through its website, stores, and galleries, as well as through a network of channel partners; and provision of service and repairs to its energy product customers, including under warranty, as well as various financing options to its solar customers. The company was formerly known as Tesla Motors, Inc. and changed its name to Tesla, Inc. in February 2017. Tesla, Inc. was incorporated in 2003 and is headquartered in Austin, Texas.', 'fullTimeEmployees': 140473, 'companyOfficers': [{'maxAge': 1, 'name': 'Mr. Elon R. Musk', 'age': 51, 'title': 'Co-Founder, Technoking of Tesla, CEO & Director', 'yearBorn': 1972, 'fiscalYear': 2023, 'exercisedValue': 0, 'unexercisedValue': 0}, {'maxAge': 1, 'name': 'Mr. Vaibhav Taneja', 'age': 45, 'title': 'Chief Financial Officer', 'yearBorn': 1978, 'fiscalYear': 2023, 'totalPay': 278000, 'exercisedValue': 8517957, 'unexercisedValue': 202075632}, {'maxAge': 1, 'name': 'Mr. Xiaotong Zhu', 'age': 43, 'title': 'Senior Vice President of Automotive', 'yearBorn': 1980, 'fiscalYear': 2023, 'totalPay': 926877, 'exercisedValue': 0, 'unexercisedValue': 344144320}, {'maxAge': 1, 'name': 'Travis Axelrod', 'title': 'Head of Investor Relations', 'fiscalYear': 2023, 'exercisedValue': 0, 'unexercisedValue': 0}, {'maxAge': 1, 'name': 'Brian Scelfo', 'title': 'Senior Director of Corporate Development', 'fiscalYear': 2023, 'exercisedValue': 0, 'unexercisedValue': 0}, {'maxAge': 1, 'name': 'Mr. Franz von Holzhausen', 'title': 'Chief Designer', 'fiscalYear': 2023, 'exercisedValue': 0, 'unexercisedValue': 0}, {'maxAge': 1, 'name': 'Mr. John Walker', 'age': 60, 'title': 'Vice President of Sales - North America', 'yearBorn': 1963, 'fiscalYear': 2023, 'totalPay': 121550, 'exercisedValue': 0, 'unexercisedValue': 0}, {'maxAge': 1, 'name': 'Mr. Peter Bannon', 'title': 'Chip Architect', 'fiscalYear': 2023, 'exercisedValue': 0, 'unexercisedValue': 0}, {'maxAge': 1, 'name': 'Mr. Turner Caldwell', 'title': 'Engineering Manager', 'fiscalYear': 2023, 'exercisedValue': 0, 'unexercisedValue': 0}, {'maxAge': 1, 'name': 'Mr. Rodney D. Westmoreland Jr.', 'title': 'Director of Construction Management', 'fiscalYear': 2023, 'exercisedValue': 0, 'unexercisedValue': 0}], 'auditRisk': 7, 'boardRisk': 9, 'compensationRisk': 10, 'shareHolderRightsRisk': 9, 'overallRisk': 10, 'governanceEpochDate': 1727740800, 'compensationAsOfEpochDate': 1703980800, 'maxAge': 86400, 'priceHint': 2, 'previousClose': 218.85, 'open': 217.0636, 'dayLow': 215.26, 'dayHigh': 218.2, 'regularMarketPreviousClose': 218.85, 'regularMarketOpen': 217.0636, 'regularMarketDayLow': 215.26, 'regularMarketDayHigh': 218.2, 'beta': 2.297, 'trailingPE': 61.056023, 'forwardPE': 71.937294, 'volume': 42776597, 'regularMarketVolume': 42776597, 'averageVolume': 79394303, 'averageVolume10days': 69495490, 'averageDailyVolume10Day': 69495490, 'marketCap': 696335663104, 'fiftyTwoWeekLow': 138.8, 'fiftyTwoWeekHigh': 271.0, 'priceToSalesTrailing12Months': 7.305395, 'fiftyDayAverage': 227.7604, 'twoHundredDayAverage': 201.49934, 'currency': 'USD', 'enterpriseValue': 678925762560, 'profitMargins': 0.12996, 'floatShares': 2777644002, 'sharesOutstanding': 3194639872, 'sharesShort': 74332630, 'sharesShortPriorMonth': 78698016, 'sharesShortPreviousMonthDate': 1724976000, 'dateShortInterest': 1727654400, 'sharesPercentSharesOut': 0.0233, 'heldPercentInsiders': 0.12976, 'heldPercentInstitutions': 0.46936, 'shortRatio': 0.93, 'shortPercentOfFloat': 0.026700001, 'impliedSharesOutstanding': 3194639872, 'bookValue': 20.81, 'priceToBook': 10.474292, 'lastFiscalYearEnd': 1703980800, 'nextFiscalYearEnd': 1735603200, 'mostRecentQuarter': 1719705600, 'earningsQuarterlyGrowth': -0.453, 'netIncomeToCommon': 12389999616, 'trailingEps': 3.57, 'forwardEps': 3.03, 'pegRatio': 62.74, 'lastSplitFactor': '3:1', 'lastSplitDate': 1661385600, 'enterpriseToRevenue': 7.123, 'enterpriseToEbitda': 55.75, '52WeekChange': 0.030270219, 'SandP52WeekChange': 0.3982091, 'exchange': 'NMS', 'quoteType': 'EQUITY', 'symbol': 'TSLA', 'underlyingSymbol': 'TSLA', 'shortName': 'Tesla, Inc.', 'longName': 'Tesla, Inc.', 'firstTradeDateEpochUtc': 1277818200, 'timeZoneFullName': 'America/New_York', 'timeZoneShortName': 'EDT', 'uuid': 'ec367bc4-f92c-397c-ac81-bf7b43cffaf7', 'messageBoardId': 'finmb_27444752', 'gmtOffSetMilliseconds': -14400000, 'currentPrice': 217.97, 'targetHighPrice': 310.0, 'targetLowPrice': 24.86, 'targetMeanPrice': 210.94, 'targetMedianPrice': 221.0, 'recommendationMean': 2.8, 'recommendationKey': 'hold', 'numberOfAnalystOpinions': 45, 'totalCash': 30720000000, 'totalCashPerShare': 9.616, 'ebitda': 12177999872, 'totalDebt': 12515000320, 'quickRatio': 1.249, 'currentRatio': 1.911, 'totalRevenue': 95317999616, 'debtToEquity': 18.606, 'revenuePerShare': 29.932, 'returnOnAssets': 0.044159997, 'returnOnEquity': 0.20861, 'freeCashflow': -907249984, 'operatingCashflow': 11532000256, 'earningsGrowth': -0.462, 'revenueGrowth': 0.023, 'grossMargins': 0.17719999, 'ebitdaMargins': 0.12776, 'operatingMargins': 0.0858, 'financialCurrency': 'USD', 'trailingPegRatio': 6.8585}

Using the ticker object and the function `history` extract stock information and save it in a dataframe named `tesla_data`. Set the `period` parameter to `"max"` so we get information for the maximum amount of time.

In [9]:
```
import yfinance as yf

# Create a Ticker object for Tesla
tesla = yf.Ticker("TSLA")

# Extract stock data for the maximum available period and save it to a DataFrame
tesla_data = tesla.history(period="max")
```

```
# Check the first few rows of the data
print(tesla_data.head())
```

```
                               Open      High       Low     Close     Volume  \
Date
2010-06-29 00:00:00-04:00  1.266667  1.666667  1.169333  1.592667  281494500
2010-06-30 00:00:00-04:00  1.719333  2.028000  1.553333  1.588667  257806500
2010-07-01 00:00:00-04:00  1.666667  1.728000  1.351333  1.464000  123282000
2010-07-02 00:00:00-04:00  1.533333  1.540000  1.247333  1.280000   77097000
2010-07-06 00:00:00-04:00  1.333333  1.333333  1.055333  1.074000  103003500

                           Dividends  Stock Splits
Date
2010-06-29 00:00:00-04:00        0.0           0.0
2010-06-30 00:00:00-04:00        0.0           0.0
2010-07-01 00:00:00-04:00        0.0           0.0
2010-07-02 00:00:00-04:00        0.0           0.0
2010-07-06 00:00:00-04:00        0.0           0.0
```

Reset the index using the `reset_index(inplace=True)` function on the tesla_data DataFrame and display the first five rows of the `tesla_data` dataframe using the `head` function. Take a screenshot of the results and code from the beginning of Question 1 to the results below.

In [10]:
```
import yfinance as yf

# Step 1: Create a Ticker object for Tesla
tesla = yf.Ticker("TSLA")

# Step 2: Extract stock data for the maximum available period and save it to a DataFrame
tesla_data = tesla.history(period="max")

# Step 3: Reset the index of the tesla_data DataFrame
tesla_data.reset_index(inplace=True)

# Step 4: Display the first five rows of the tesla_data DataFrame
print(tesla_data.head())
```

```
                       Date      Open      High       Low     Close  \
0 2010-06-29 00:00:00-04:00  1.266667  1.666667  1.169333  1.592667
1 2010-06-30 00:00:00-04:00  1.719333  2.028000  1.553333  1.588667
2 2010-07-01 00:00:00-04:00  1.666667  1.728000  1.351333  1.464000
3 2010-07-02 00:00:00-04:00  1.533333  1.540000  1.247333  1.280000
4 2010-07-06 00:00:00-04:00  1.333333  1.333333  1.055333  1.074000

      Volume  Dividends  Stock Splits
0  281494500        0.0           0.0
1  257806500        0.0           0.0
2  123282000        0.0           0.0
3   77097000        0.0           0.0
4  103003500        0.0           0.0
```

Question 2: Use Webscraping to Extract Tesla Revenue Data

Use the `requests` library to download the webpage https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm Save the text of the response as a variable named `html_data`.

In [12]:
```
import requests

# URL of the webpage containing Tesla revenue data
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/revenue.htm"

# Send a GET request to download the webpage content
response = requests.get(url)

# Save the text of the response in a variable named html_data
html_data = response.text

# Output the first 500 characters to verify
print(html_data[:500])
```

```
<!DOCTYPE html>
<!--[if lt IE 7]>      <html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->
<!--[if IE 7]>         <html class="no-js lt-ie9 lt-ie8"> <![endif]-->
<!--[if IE 8]>         <html class="no-js lt-ie9"> <![endif]-->
<!--[if gt IE 8]><!--> <html class="no-js"> <!--<![endif]-->
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
                <link rel="canonical" href="https://www.macrotrends.net/stocks/charts/TSLA/tesla/revenue" />
```

Parse the html data using `beautiful_soup` using parser i.e `html5lib` or `html.parser`. Make sure to use the `html_data` with the content parameter as follow `html_data.content`.

In [13]:
```
from bs4 import BeautifulSoup

# Parse the HTML data using BeautifulSoup with the 'html.parser'
soup = BeautifulSoup(html_data, 'html.parser')

# Output a formatted version of the parsed HTML (optional for verification)
print(soup.prettify()[:500])  # Printing the first 500 characters of formatted HTML for verification
```

```
<!DOCTYPE html>
<!--[if lt IE 7]>      <html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->
<!--[if IE 7]>         <html class="no-js lt-ie9 lt-ie8"> <![endif]-->
<!--[if IE 8]>         <html class="no-js lt-ie9"> <![endif]-->
<!--[if gt IE 8]><!-->
<html class="no-js">
 <!--<![endif]-->
 <head>
  <meta charset="utf-8"/>
  <meta content="IE=edge,chrome=1" http-equiv="X-UA-Compatible"/>
  <link href="https://www.macrotrends.net/stocks/charts/TSLA/tesla/revenue" rel="canonical"/>
  <title>
   Te
```

Using `BeautifulSoup` or the `read_html` function extract the table with `Tesla Revenue` and store it into a dataframe named `tesla_revenue`. The dataframe should have columns `Date` and `Revenue`.

▶ Step-by-step instructions

▶ Click here if you need help locating the table

In [14]:
```
from bs4 import BeautifulSoup
import pandas as pd

# Parse the HTML using BeautifulSoup
soup = BeautifulSoup(html_data, 'html.parser')

# Find the table containing the revenue data (assume it's the first table)
table = soup.find('table')

# Initialize lists to store the table data
dates = []
revenues = []

# Loop through the table rows and extract the Date and Revenue columns
for row in table.find_all('tr')[1:]:  # Skipping the header row
    cols = row.find_all('td')
    dates.append(cols[0].text.strip())    # Date
    revenues.append(cols[1].text.strip()) # Revenue

# Create a DataFrame from the extracted data
tesla_revenue = pd.DataFrame({
    'Date': dates,
    'Revenue': revenues
})

# Display the first few rows of the tesla_revenue DataFrame
print(tesla_revenue.head())
```

```
   Date  Revenue
0  2021  $53,823
1  2020  $31,536
2  2019  $24,578
3  2018  $21,461
4  2017  $11,759
```

Execute the following line to remove the comma and dollar sign from the `Revenue` column.

In [18]:
```
tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace(',|\$',"", regex=True)
```

Execute the following lines to remove an null or empty strings in the Revenue column.

In [19]:
```
tesla_revenue.dropna(inplace=True)

tesla_revenue = tesla_revenue[tesla_revenue['Revenue'] != ""]
```

Display the last 5 row of the `tesla_revenue` dataframe using the `tail` function. Take a screenshot of the results.

In [20]:
```
# Display the last 5 rows of the tesla_revenue DataFrame
print(tesla_revenue.tail())
```

```
    Date Revenue
8   2013    2013
9   2012     413
10  2011     204
11  2010     117
12  2009     112
```

Question 3: Use yfinance to Extract Stock Data

Using the `Ticker` function enter the ticker symbol of the stock we want to extract data on to create a ticker object. The stock is GameStop and its ticker symbol is `GME`.

In [21]:
```
import yfinance as yf

# Create a Ticker object for GameStop
gamestop = yf.Ticker("GME")

# Access basic information about GameStop
print(gamestop.info)
```

{'address1': '625 Westport Parkway', 'city': 'Grapevine', 'state': 'TX', 'zip': '76051', 'country': 'United States', 'phone': '817 424 2000', 'website': 'http
s://www.gamestop.com', 'industry': 'Specialty Retail', 'industryKey': 'specialty-retail', 'industryDisp': 'Specialty Retail', 'sector': 'Consumer Cyclical', 's
ectorKey': 'consumer-cyclical', 'sectorDisp': 'Consumer Cyclical', 'longBusinessSummary': 'GameStop Corp., a specialty retailer, provides games and entertainme
nt products through its stores and ecommerce platforms in the United States, Canada, Australia, and Europe. The company sells new and pre-owned gaming platform
s; accessories, such as controllers, gaming headsets, and virtual reality products; new and pre-owned gaming software; and in-game digital currency, digital do
wnloadable content, and full-game downloads. It sells collectibles comprising apparel, toys, trading cards, gadgets, and other retail products for pop culture
and technology enthusiasts, as well as engages in the digital asset wallet and NFT marketplace activities. The company operates stores and ecommerce sites unde
r the GameStop, EB Games, and Micromania brands; and pop culture themed stores that sell collectibles, apparel, gadgets, electronics, toys, and other retail pr
oducts under the Zing Pop Culture brand, as well as offers Game Informer magazine, a print and digital gaming publication. The company was formerly known as GS
C Holdings Corp. GameStop Corp. was founded in 1996 and is headquartered in Grapevine, Texas.', 'fullTimeEmployees': 8000, 'companyOfficers': [{'maxAge': 1, 'n
ame': 'Mr. Ryan Cohen', 'age': 37, 'title': 'President, CEO & Executive Chairman', 'yearBorn': 1986, 'fiscalYear': 2023, 'exercisedValue': 0, 'unexercisedValu
e': 0}, {'maxAge': 1, 'name': 'Mr. Daniel William Moore', 'age': 40, 'title': 'Principal Accounting Officer & Principal Financial Officer', 'yearBorn': 1983,
'fiscalYear': 2023, 'totalPay': 277711, 'exercisedValue': 0, 'unexercisedValue': 0}, {'maxAge': 1, 'name': 'Mr. Mark Haymond Robinson', 'age': 45, 'title': 'Ge
neral Counsel & Secretary', 'yearBorn': 1978, 'fiscalYear': 2023, 'totalPay': 337657, 'exercisedValue': 0, 'unexercisedValue': 0}], 'auditRisk': 8, 'boardRis
k': 6, 'compensationRisk': 7, 'shareHolderRightsRisk': 3, 'overallRisk': 5, 'governanceEpochDate': 1727740800, 'compensationAsOfEpochDate': 1703980800, 'irWebs
ite': 'http://phx.corporate-ir.net/phoenix.zhtml?c=130125&p=irol-irhome', 'maxAge': 86400, 'priceHint': 2, 'previousClose': 20.7, 'open': 20.62, 'dayLow': 20.4
3, 'dayHigh': 20.93, 'regularMarketPreviousClose': 20.7, 'regularMarketOpen': 20.62, 'regularMarketDayLow': 20.43, 'regularMarketDayHigh': 20.93, 'exDividendDa
te': 1552521600, 'fiveYearAvgDividendYield': 9.52, 'beta': -0.162, 'trailingPE': 149.5, 'forwardPE': -2093.0, 'volume': 3063065, 'regularMarketVolume': 306306
5, 'averageVolume': 8454998, 'averageVolume10days': 4489400, 'averageDailyVolume10Day': 4489400, 'bid': 20.8, 'ask': 20.84, 'bidSize': 800, 'askSize': 1300, 'm
arketCap': 9345455104, 'fiftyTwoWeekLow': 9.95, 'fiftyTwoWeekHigh': 64.83, 'priceToSalesTrailing12Months': 2.0530438, 'fiftyDayAverage': 21.6872, 'twoHundredDa
yAverage': 19.31975, 'currency': 'USD', 'enterpriseValue': 5256145920, 'profitMargins': 0.00934, 'floatShares': 390217891, 'sharesOutstanding': 446510016, 'sha
resShort': 38133807, 'sharesShortPriorMonth': 35551855, 'sharesShortPreviousMonthDate': 1724976000, 'dateShortInterest': 1727654400, 'sharesPercentSharesOut':
0.0854, 'heldPercentInsiders': 0.08495, 'heldPercentInstitutions': 0.22136, 'shortRatio': 2.74, 'shortPercentOfFloat': 0.1024, 'impliedSharesOutstanding': 4465
10016, 'bookValue': 10.278, 'priceToBook': 2.0363884, 'lastFiscalYearEnd': 1706918400, 'nextFiscalYearEnd': 1738540800, 'mostRecentQuarter': 1722643200, 'netIn
comeToCommon': 42500000, 'trailingEps': 0.14, 'forwardEps': -0.01, 'pegRatio': 14.82, 'lastSplitFactor': '4:1', 'lastSplitDate': 1658448000, 'enterpriseToReven
ue': 1.155, 'enterpriseToEbitda': 111.359, '52WeekChange': 0.5098468, 'SandP52WeekChange': 0.3982091, 'lastDividendValue': 0.095, 'lastDividendDate': 155252160
0, 'exchange': 'NYQ', 'quoteType': 'EQUITY', 'symbol': 'GME', 'underlyingSymbol': 'GME', 'shortName': 'GameStop Corporation', 'longName': 'GameStop Corp.', 'fi
rstTradeDateEpochUtc': 1013610600, 'timeZoneFullName': 'America/New_York', 'timeZoneShortName': 'EDT', 'uuid': '8ded85bd-8171-3e2e-afa6-c81272285147', 'message
BoardId': 'finmb_1342560', 'gmtOffSetMilliseconds': -14400000, 'currentPrice': 20.93, 'targetHighPrice': 10.0, 'targetLowPrice': 5.75, 'targetMeanPrice': 7.88,
'targetMedianPrice': 7.88, 'recommendationMean': 4.5, 'recommendationKey': 'underperform', 'numberOfAnalystOpinions': 2, 'totalCash': 4204199936, 'totalCashPer
Share': 9.857, 'ebitda': 47200000, 'totalDebt': 533500000, 'quickRatio': 5.442, 'currentRatio': 6.233, 'totalRevenue': 4552000000, 'debtToEquity': 12.171, 'rev
enuePerShare': 13.97, 'returnOnAssets': 0.00043000001, 'returnOnEquity': 0.015039999, 'freeCashflow': -93387504, 'operatingCashflow': -33100000, 'revenueGrowt
h': -0.314, 'grossMargins': 0.26237, 'ebitdaMargins': 0.010369999, 'operatingMargins': -0.03558, 'financialCurrency': 'USD', 'trailingPegRatio': None}

Using the ticker object and the function `history` extract stock information and save it in a dataframe named `gme_data`. Set the `period` parameter to `"max"` so we get information for the maximum amount of time.

In [23]:
```
import yfinance as yf

# Create a Ticker object for GameStop
gamestop = yf.Ticker("GME")

# Extract stock data for the maximum available period and save it to a DataFrame
gme_data = gamestop.history(period="max")

# Reset the index
gme_data.reset_index(inplace=True)

# Display the first few rows of the gme_data DataFrame
print(gme_data.head())
```

```
                       Date      Open      High       Low     Close    Volume  \
0 2002-02-13 00:00:00-05:00  1.620129  1.693350  1.603296  1.691667  76216000
1 2002-02-14 00:00:00-05:00  1.712707  1.716073  1.670626  1.683250  11021600
2 2002-02-15 00:00:00-05:00  1.683251  1.687459  1.658002  1.674834   8389600
3 2002-02-19 00:00:00-05:00  1.666418  1.666418  1.578047  1.607504   7410400
4 2002-02-20 00:00:00-05:00  1.615920  1.662210  1.603296  1.662210   6892800

   Dividends  Stock Splits
0        0.0           0.0
1        0.0           0.0
2        0.0           0.0
3        0.0           0.0
4        0.0           0.0
```

Reset the index using the `reset_index(inplace=True)` function on the gme_data DataFrame and display the first five rows of the `gme_data` dataframe using the `head` function. Take a screenshot of the results and code from the beginning of Question 3 to the results below.

In [24]:
```
import yfinance as yf

# Step 1: Create a Ticker object for GameStop
gamestop = yf.Ticker("GME")

# Step 2: Extract stock data for the maximum available period and save it to a DataFrame
gme_data = gamestop.history(period="max")

# Step 3: Reset the index of the gme_data DataFrame
gme_data.reset_index(inplace=True)

# Step 4: Display the first five rows of the gme_data DataFrame
print(gme_data.head())
```

```
                       Date      Open      High       Low     Close    Volume  \
0 2002-02-13 00:00:00-05:00  1.620128  1.693350  1.603296  1.691666  76216000
1 2002-02-14 00:00:00-05:00  1.712707  1.716074  1.670626  1.683250  11021600
2 2002-02-15 00:00:00-05:00  1.683250  1.687458  1.658001  1.674834   8389600
3 2002-02-19 00:00:00-05:00  1.666417  1.666417  1.578047  1.607504   7410400
4 2002-02-20 00:00:00-05:00  1.615920  1.662210  1.603296  1.662210   6892800

   Dividends  Stock Splits
0        0.0           0.0
1        0.0           0.0
2        0.0           0.0
3        0.0           0.0
4        0.0           0.0
```

Question 4: Use Webscraping to Extract GME Revenue Data

Use the `requests` library to download the webpage https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html. Save the text of the response as a variable named `html_data_2` .

In [25]:
```
import requests

# URL of the webpage containing stock data
url_2 = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html"

# Send a GET request to download the webpage content
response_2 = requests.get(url_2)

# Save the text of the response in a variable named html_data_2
html_data_2 = response_2.text

# Output the first 500 characters to verify
print(html_data_2[:500])  # Printing the first 500 characters for verification
```

```
<!DOCTYPE html>
<!-- saved from url=(0105)https://web.archive.org/web/20200814131437/https://www.macrotrends.net/stocks/charts/GME/gamestop/revenue -->
<html class=" js flexbox canvas canvastext webgl no-touch geolocation postmessage websqldatabase indexeddb hashchange history draganddrop websockets rgba hsla
multiplebgs backgroundsize borderimage borderradius boxshadow textshadow opacity cssanimations csscolumns cssgradients cssreflections csstransforms csstransfor
ms3d csstransitions fontface g
```

Parse the html data using `beautiful_soup` using parser i.e `html5lib` or `html.parser`.

In [26]:
```
from bs4 import BeautifulSoup

# Parse the HTML data using BeautifulSoup with the 'html.parser'
soup_2 = BeautifulSoup(html_data_2, 'html.parser')

# Output a formatted version of the parsed HTML (optional for verification)
print(soup_2.prettify()[:500])  # Print the first 500 characters of the formatted HTML for verification
```

```
<!DOCTYPE html>
<!-- saved from url=(0105)https://web.archive.org/web/20200814131437/https://www.macrotrends.net/stocks/charts/GME/gamestop/revenue -->
<html class="js flexbox canvas canvastext webgl no-touch geolocation postmessage websqldatabase indexeddb hashchange history draganddrop websockets rgba hsla m
ultiplebgs backgroundsize borderimage borderradius boxshadow textshadow opacity cssanimations csscolumns cssgradients cssreflections csstransforms csstransform
s3d csstransitions fontface ge
```

Using `BeautifulSoup` or the `read_html` function extract the table with `GameStop Revenue` and store it into a dataframe named `gme_revenue`. The dataframe should have columns `Date` and `Revenue`. Make sure the comma and dollar sign is removed from the `Revenue` column.

> **Note: Use the method similar to what you did in question 2.**

▶ Click here if you need help locating the table

In [27]:
```
from bs4 import BeautifulSoup
import pandas as pd

# Parse the HTML using BeautifulSoup
soup_2 = BeautifulSoup(html_data_2, 'html.parser')

# Find the table containing the revenue data (assuming it's the first table)
table = soup_2.find('table')
```

```
# Initialize lists to store the table data
dates = []
revenues = []

# Loop through the table rows and extract the Date and Revenue columns
for row in table.find_all('tr')[1:]:  # Skip the header row
    cols = row.find_all('td')
    dates.append(cols[0].text.strip())    # Date
    revenues.append(cols[1].text.strip())  # Revenue

# Create a DataFrame from the extracted data
gme_revenue = pd.DataFrame({
    'Date': dates,
    'Revenue': revenues
})

# Remove the dollar signs and commas from the 'Revenue' column
gme_revenue['Revenue'] = gme_revenue['Revenue'].replace({'\$': '', ',': ''}, regex=True)

# Convert the Revenue column to numeric
gme_revenue['Revenue'] = pd.to_numeric(gme_revenue['Revenue'], errors='coerce')

# Display the first few rows of the gme_revenue DataFrame
print(gme_revenue.head())
```

```
  Date  Revenue
0  2020     6466
1  2019     8285
2  2018     8547
3  2017     7965
4  2016     9364
```

Display the last five rows of the `gme_revenue` dataframe using the `tail` function. Take a screenshot of the results.

In [28]:
```
import requests
import pandas as pd
from bs4 import BeautifulSoup

# Step 1: Download the webpage
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0220EN-SkillsNetwork/labs/project/stock.html"
response = requests.get(url)
html_data_2 = response.text

# Step 2: Extract the table with GameStop revenue data using pandas.read_html
tables = pd.read_html(html_data_2)
gme_revenue = tables[0]  # Assuming the relevant table is the first one

# Rename columns to 'Date' and 'Revenue'
gme_revenue.columns = ['Date', 'Revenue']

# Step 3: Remove the dollar signs and commas from the 'Revenue' column
gme_revenue['Revenue'] = gme_revenue['Revenue'].replace({'\$': '', ',': ''}, regex=True)
```

```
# Convert the Revenue column to numeric
gme_revenue['Revenue'] = pd.to_numeric(gme_revenue['Revenue'], errors='coerce')

# Step 4: Display the last five rows of the gme_revenue DataFrame
print(gme_revenue.tail())
```

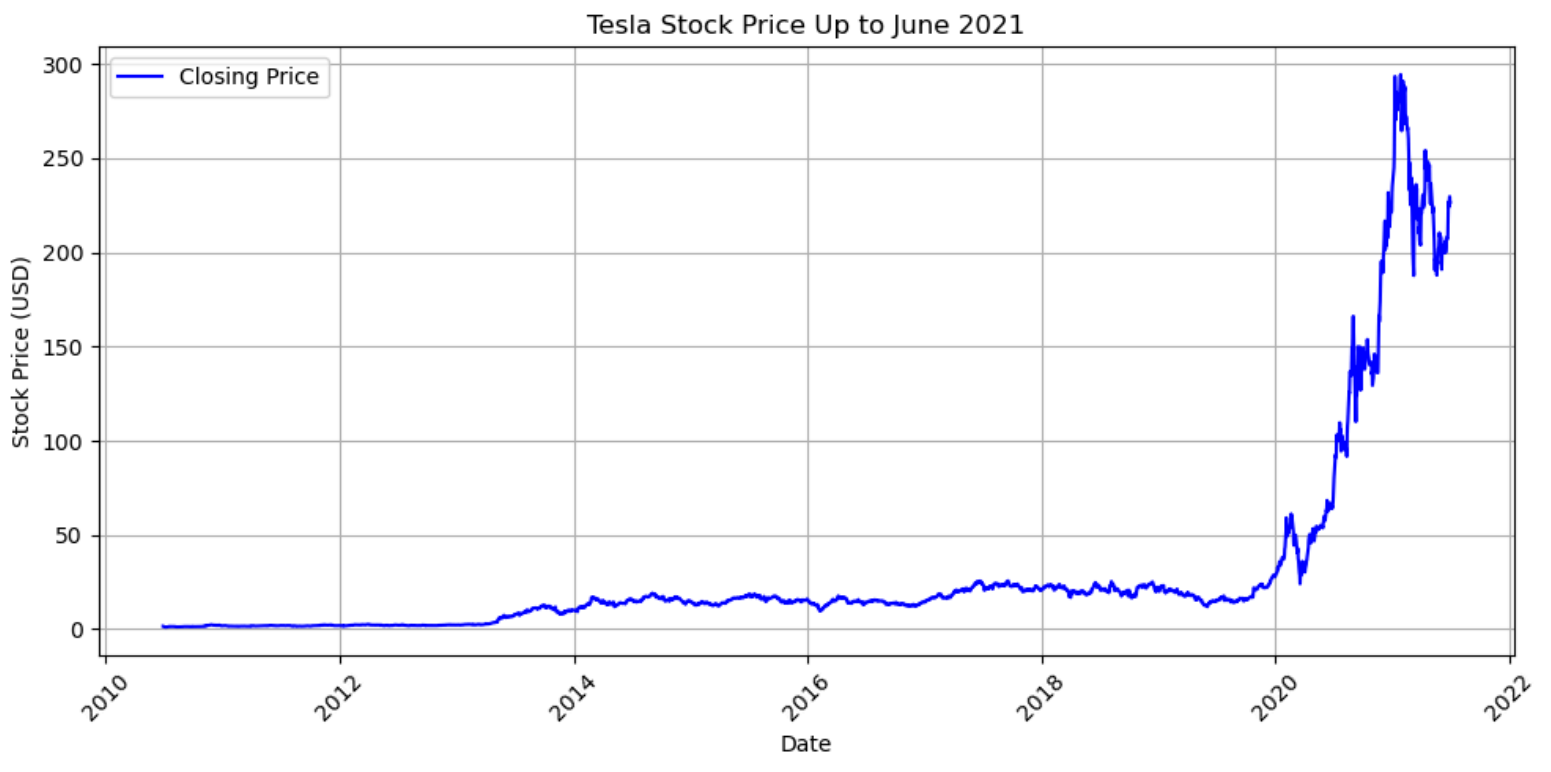
```
    Date  Revenue
11  2009     8806
12  2008     7094
13  2007     5319
14  2006     3092
15  2005     1843
```

Question 5: Plot Tesla Stock Graph

Use the `make_graph` function to graph the Tesla Stock Data, also provide a title for the graph. Note the graph will only show data upto June 2021.

▶ Hint

In [29]:
```
import matplotlib.pyplot as plt

# Step 1: Filter the Tesla stock data up to June 2021
tesla_data_filtered = tesla_data[tesla_data['Date'] <= '2021-06-30']

# Step 2: Define the make_graph function
def make_graph(data, title):
    plt.figure(figsize=(10, 5))
    plt.plot(data['Date'], data['Close'], label='Closing Price', color='blue')
    plt.title(title)
    plt.xlabel('Date')
    plt.ylabel('Stock Price (USD)')
    plt.xticks(rotation=45)
    plt.legend()
    plt.grid()
    plt.tight_layout()
    plt.show()

# Step 3: Call the make_graph function with the filtered data and a title
make_graph(tesla_data_filtered, "Tesla Stock Price Up to June 2021")
```



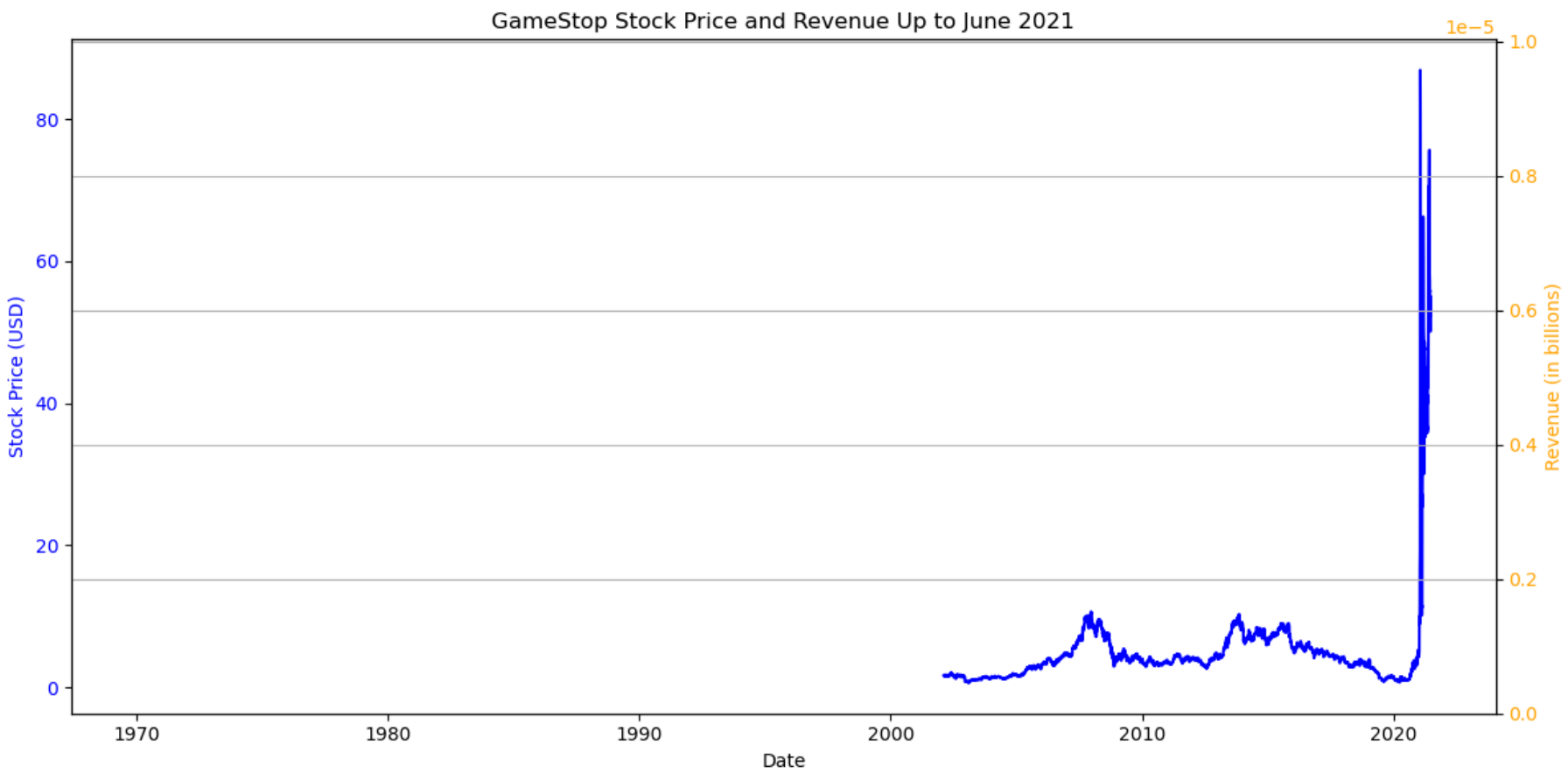
Question 6: Plot GameStop Stock Graph

Use the `make_graph` function to graph the GameStop Stock Data, also provide a title for the graph. The structure to call the `make_graph` function is `make_graph(gme_data, gme_revenue, 'GameStop')` . Note the graph will only show data upto June 2021.

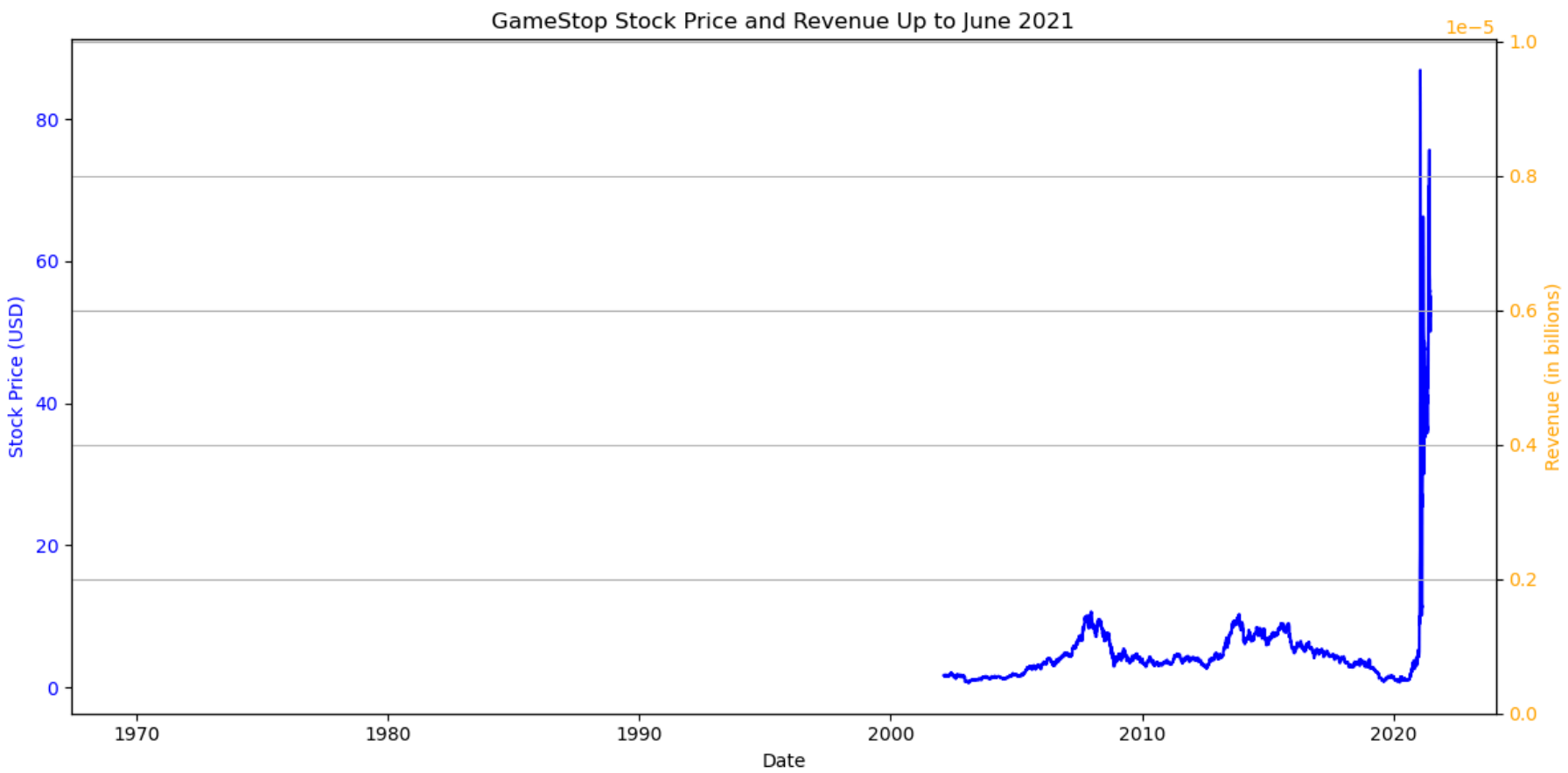
In [46]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Check data types
print(gme_data.dtypes)
print(gme_revenue.dtypes)

# Convert Date columns to datetime if needed
gme_data['Date'] = pd.to_datetime(gme_data['Date'])
gme_revenue['Date'] = pd.to_datetime(gme_revenue['Date'])

# Define the make_graph function
def make_graph(stock_data, revenue_data, title):
    # Filter the data to only include up to June 2021
    stock_data_filtered = stock_data[stock_data['Date'] <= '2021-06-30']
    revenue_data_filtered = revenue_data[revenue_data['Date'] <= '2021-06-30']
```

```
    # Create a figure and axis
    fig, ax1 = plt.subplots(figsize=(12, 6))

    # Plot stock prices
    ax1.set_xlabel('Date')
    ax1.set_ylabel('Stock Price (USD)', color='blue')
    ax1.plot(stock_data_filtered['Date'], stock_data_filtered['Close'], label='Closing Price', color='blue')
    ax1.tick_params(axis='y', labelcolor='blue')

    # Create a second y-axis for revenue
    ax2 = ax1.twinx()
    ax2.set_ylabel('Revenue (in billions)', color='orange')
    ax2.bar(revenue_data_filtered['Date'], revenue_data_filtered['Revenue'] / 1e9, width=10, alpha=0.3, color='orange', label='Revenue')
    ax2.tick_params(axis='y', labelcolor='orange')

    # Add title and grid
    plt.title(title + " Stock Price and Revenue Up to June 2021")
    fig.tight_layout()  # Adjust layout to prevent overlap
    plt.grid()
    plt.show()

# Call the make_graph function to plot the GameStop data
make_graph(gme_data, gme_revenue, 'GameStop')
```

```
Date            datetime64[ns, America/New_York]
Open                                     float64
High                                     float64
Low                                      float64
Close                                    float64
Volume                                     int64
Dividends                                float64
Stock Splits                             float64
dtype: object
Date       datetime64[ns]
Revenue             int64
dtype: object
```



In [45]:
```
make_graph(gme_data, gme_revenue, 'GameStop')
```



About the Authors:

Joseph Santarcangelo has a PhD in Electrical Engineering, his research focused on using machine learning, signal processing, and computer vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.


```
toggle
```

```
toggle
```

```
toggle
```

```
toggle
```

```
toggle
```

```
toggle
```