PeerReview_TSLA_GME

November 2, 2025

[22]:
```
# Setup
!pip -q install yfinance pandas lxml html5lib plotly
import yfinance as yf
import pandas as pd
import plotly.express as px
pd.options.display.float_format = '{:,.6g}'.format
```

0.1 Frage 1 – Extracting Tesla Stock Data using yfinance

[23]:
```
tesla = yf.Ticker("TSLA")
tesla_data = tesla.history(period="max").reset_index()
tesla_data.head()
```

[23]:

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2010-06-29 00:00:00-04:00	1.26667	1.66667	1.16933	1.59267	281494500	0	0
1	2010-06-30 00:00:00-04:00	1.71933	2.028	1.55333	1.58867	257806500	0	0
2	2010-07-01 00:00:00-04:00	1.66667	1.728	1.35133	1.464	123282000	0	0
3	2010-07-02 00:00:00-04:00	1.53333	1.54	1.24733	1.28	77097000	0	0
4	2010-07-06 00:00:00-04:00	1.33333	1.33333	1.05533	1.074	103003500	0	0

0.2 Frage 2 – Extracting Tesla Revenue Data using Webscraping

[43]:
```
# --- START: Corrected Code for QUESTION 2 (Tesla Revenue) ---

# 1) Set the URL for Tesla revenue
url_tesla = "https://www.macrotrends.net/stocks/charts/TSLA/tesla/revenue"

# 2) Set a User-Agent header to pretend I'm a browser
headers = {
    "User-Agent": (
        "Mozilla/5.0 (Windows NT 10.0; Win64; x64) "
```

```
        "AppleWebKit/537.36 (KHTML, like Gecko) "
        "Chrome/124.0.0.0 Safari/537.36"
    )
}

# 3) Download the page HTML and parse it with BeautifulSoup
html_text = requests.get(url_tesla, headers=headers, timeout=30).text
soup = BeautifulSoup(html_text, "html.parser")

# 4) Use pandas to read all tables. I'll use StringIO to avoid the␣
 ↪FutureWarning.
all_tables = pd.read_html(StringIO(str(soup)))

# 5) Select the correct table.
#    From inspecting the page, I know:
#    - all_tables[0] is the ANNUAL revenue
#    - all_tables[1] is the QUARTERLY revenue
#    I need the QUARTERLY data for the graph.
tesla_revenue = all_tables[1]

# 6) Rename columns for clarity
tesla_revenue.columns = ["Date", "Revenue"]

# 7) Clean the 'Revenue' column: remove '$' and ','
tesla_revenue["Revenue"] = (
    tesla_revenue["Revenue"]
    .astype(str)
    .str.replace(r"[\$,]", "", regex=True)
)

# 8) Remove any empty or invalid rows
tesla_revenue.dropna(inplace=True)
tesla_revenue = tesla_revenue[tesla_revenue['Revenue'] != ""]

# 9) Convert columns to the correct data types
tesla_revenue["Revenue"] = pd.to_numeric(tesla_revenue["Revenue"])
tesla_revenue["Date"] = pd.to_datetime(tesla_revenue["Date"])

# 10) Sort the data by date to make sure it's chronological
tesla_revenue = tesla_revenue.sort_values("Date").reset_index(drop=True)

# 11) Display the last 5 rows to confirm the data is correct
print("Tesla Revenue Data (Corrected Version):")
print(tesla_revenue.tail())

# --- END: Corrected Code for QUESTION 2 ---
```

```
Tesla Revenue Data (Corrected Version):
```

```
         Date  Revenue
54 2024-06-30    25500
55 2024-09-30    25182
56 2024-12-31    25707
57 2025-03-31    19335
58 2025-06-30    22496
```

0.3 Frage 3 –Extracting GameStop Stock Data using yfinance

[44]:
```
import yfinance as yf
import pandas as pd

# 1) Create a ticker object for GameStop
#    The ticker symbol "GME" identifies GameStop on the stock exchange.
gme = yf.Ticker("GME")

# 2) Download the full historical stock data
#    'period="max"' retrieves all available data for GameStop.
gme_data = gme.history(period="max")

# 3) Reset the index so that 'Date' becomes a regular column instead of the␣
 ↪index.
gme_data.reset_index(inplace=True)

# 4) Display the first five rows to verify that the data was loaded correctly.
print("\nGameStop Stock Data (gme_data):")
print(gme_data.head())
```

```
GameStop Stock Data (gme_data):
                       Date    Open    High     Low   Close    Volume  \
0 2002-02-13 00:00:00-05:00 1.62013 1.69335  1.6033 1.69167  76216000
1 2002-02-14 00:00:00-05:00 1.71271 1.71607 1.67063 1.68325  11021600
2 2002-02-15 00:00:00-05:00 1.68325 1.68746   1.658 1.67483   8389600
3 2002-02-19 00:00:00-05:00 1.66642 1.66642 1.57805  1.6075   7410400
4 2002-02-20 00:00:00-05:00 1.61592 1.66221  1.6033 1.66221   6892800

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

0.4 Frage 4 – Extracting GameStop Revenue Data using Webscraping

[48]:
```
# --- START: Corrected Code for QUESTION 4 (GameStop Revenue) ---

# 1) Set the URL for GameStop revenue
url_gme = "https://www.macrotrends.net/stocks/charts/GME/gamestop/revenue"

# 2) Set a User-Agent header (same as before)
headers = {
    "User-Agent": (
        "Mozilla/5.0 (Windows NT 10.0; Win64; x64) "
        "AppleWebKit/537.36 (KHTML, like Gecko) "
        "Chrome/124.0.0.0 Safari/537.36"
    )
}

# 3) Download the page HTML and parse it
# (Make sure 'requests', 'BeautifulSoup', 'StringIO', 'pd' are imported)
html_text = requests.get(url_gme, headers=headers, timeout=30).text
soup = BeautifulSoup(html_text, "html.parser")

# 4) Use pandas to read all tables. Use StringIO to avoid the warning.
all_tables = pd.read_html(StringIO(str(soup)))

# 5) Select the correct table.
#    - all_tables[0] is the ANNUAL revenue
#    - all_tables[1] is the QUARTERLY revenue
#    I need the QUARTERLY data for the graph.
gme_revenue = all_tables[1]

# 6) Rename columns for clarity
gme_revenue.columns = ["Date", "Revenue"]

# 7) Clean the 'Revenue' column: remove '$' and ','
gme_revenue["Revenue"] = (
    gme_revenue["Revenue"]
    .astype(str)
    .str.replace(r"[\$,]", "", regex=True)
)

# 8) Remove any empty or invalid rows
gme_revenue.dropna(inplace=True)
gme_revenue = gme_revenue[gme_revenue['Revenue'] != ""]

# 9) Convert columns to the correct data types
gme_revenue["Revenue"] = pd.to_numeric(gme_revenue["Revenue"])
gme_revenue["Date"] = pd.to_datetime(gme_revenue["Date"])
```

```
# 10) Sort the data by date
gme_revenue = gme_revenue.sort_values("Date").reset_index(drop=True)

# 11) Display the last 5 rows to confirm the data is correct
print("GameStop Revenue Data (Corrected Version):")
print(gme_revenue.tail())

# --- END: Corrected Code for QUESTION 4 ---
```

```
GameStop Revenue Data (Corrected Version):
         Date  Revenue
54 2024-07-31      798
55 2024-10-31      860
56 2025-01-31     1283
57 2025-04-30      732
58 2025-07-31      972
```

0.5 Frage 5 & 6: Create Tesla and GameStop Dashboards

[27]:
```
from plotly.subplots import make_subplots
import plotly.graph_objects as go
import pandas as pd

def make_graph(stock_data, revenue_data, stock_name):
    """
    Plot a 2-row dashboard: (1) stock Close price, (2) revenue.
    Expects:
      - stock_data: DataFrame with columns ['Date', 'Close'] (Date is␣
 ↪datetime-like)
      - revenue_data: DataFrame with columns ['Date', 'Revenue'] (Date is␣
 ↪datetime-like)
      - stock_name: string for plot titles
    """

    # --- Make defensive copies and enforce types ---
    sd = stock_data.copy()
    rd = revenue_data.copy()

    # Ensure Date columns are datetime for both dataframes
    sd["Date"] = pd.to_datetime(sd["Date"], errors="coerce")
    rd["Date"] = pd.to_datetime(rd["Date"], errors="coerce")

    # Clean and convert revenue to numeric (remove $, commas, whitespaces)
    rd["Revenue"] = (
        rd["Revenue"]
        .astype(str)
```

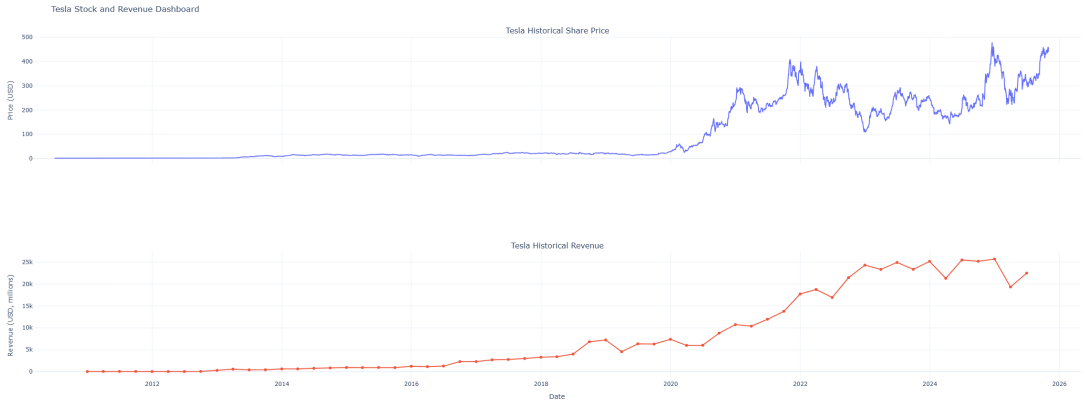
```
        .str.replace(r"[\$,]", "", regex=True)
        .str.replace(r"\s+", "", regex=True)
    )
    rd["Revenue"] = pd.to_numeric(rd["Revenue"], errors="coerce")

    # Drop invalid rows and sort chronologically
    sd = sd.dropna(subset=["Date", "Close"]).sort_values("Date")
    rd = rd.dropna(subset=["Date", "Revenue"]).sort_values("Date")

    # --- Create the figure with 2 subplots ---
    fig = make_subplots(
        rows=2, cols=1, shared_xaxes=True,
        vertical_spacing=0.1,
        subplot_titles=(f"{stock_name} Historical Share Price",
                        f"{stock_name} Historical Revenue")
    )

    # Stock price trace (row 1)
    fig.add_trace(
        go.Scatter(
            x=sd["Date"], y=sd["Close"],
            name="Share Price", mode="lines", line=dict(width=2)
        ),
        row=1, col=1
    )

    # Revenue trace (row 2)
    fig.add_trace(
        go.Scatter(
            x=rd["Date"], y=rd["Revenue"],
            name="Revenue", mode="lines", line=dict(width=2)
        ),
        row=2, col=1
    )

    # --- Axis labels and layout ---
    fig.update_yaxes(title_text="Price (USD)", row=1, col=1)
    fig.update_yaxes(title_text="Revenue (USD, millions)", row=2, col=1)
    fig.update_xaxes(title_text="Date", row=2, col=1)

    fig.update_layout(
        title_text=f"{stock_name} Stock and Revenue Dashboard",
        height=900, template="plotly_white",
        showlegend=False, margin=dict(l=50, r=20, t=70, b=40)
    )

    fig.show()
```

```
print("Function 'make_graph' is defined and ready to use.")
```

```
Function 'make_graph' is defined and ready to use.
```

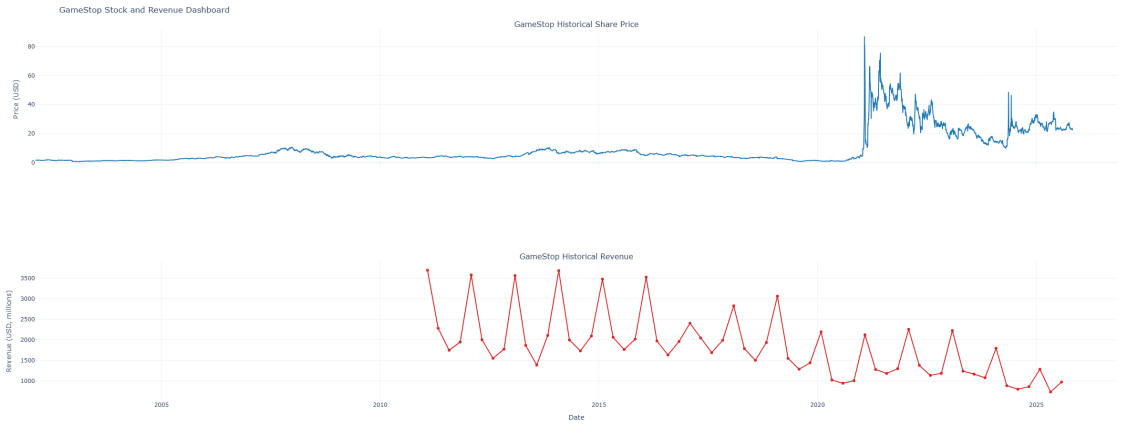
0.6 Frage 5 – Tesla Dashboard

[45]:
```
# Calling the function for Tesla
make_graph(tesla_data, tesla_revenue, 'Tesla')
```



0.7 Frage 6 – GameStop Dashboard

[49]:
```
# Calling the function for GameStop1
make_graph(gme_data, gme_revenue, 'GameStop')
```



[ ]:
```

```