In [1]:
```
# Importar las librerías necesarias
import yfinance as yf

# Descargar los datos bursátiles de Tesla (TSLA)
tesla_data = yf.download('TSLA')

# Restablecer el índice del DataFrame
tesla_data_reset = tesla_data.reset_index()

# Mostrar las primeras 5 filas del DataFrame restablecido
print(tesla_data_reset.head())

# Guardar los primeros 5 datos en un archivo CSV (opcional)
tesla_data_reset.head().to_csv('tesla_data_head.csv', index=False)
```

```
[*********************100%***********************]  1 of 1 completed
        Date      Open      High       Low     Close  Adj Close     Volume
0 2010-06-29  1.266667  1.666667  1.169333  1.592667   1.592667  281494500
1 2010-06-30  1.719333  2.028000  1.553333  1.588667   1.588667  257806500
2 2010-07-01  1.666667  1.728000  1.351333  1.464000   1.464000  123282000
3 2010-07-02  1.533333  1.540000  1.247333  1.280000   1.280000   77097000
4 2010-07-06  1.333333  1.333333  1.055333  1.074000   1.074000  103003500
```

In [2]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd
```

In [16]:
```
url = 'https://finance.yahoo.com/quote/TSLA/financials?p=TSLA'
```

In [5]:
```
response = requests.get(url)
```

In [19]:
```
if response.status_code == 200:
    soup = BeautifulSoup(response.text, 'html.parser')
```

In [21]:
```
table = soup.find_all('table')[0]
```

In [22]:
```
headers = [header.text for header in table.find_all('th')]
```

In [34]:
```
data = []
for row in table.find_all('tr')[1:]:
    cols = row.find_all('td')
    if cols:
        data.append([col.text.strip() for col in cols])
```

In [32]:
```
tesla_revenue = pd.DataFrame(data, columns=['Breakdown', 'TTM', '12/31/2023', '1
```

In [33]:
```
print(tesla_revenue.tail())
```

```
Empty DataFrame
Columns: [Breakdown, TTM, 12/31/2023, 12/31/2022, 12/31/2021, 12/31/2020, 12/31/2
021]
Index: []
```

In [38]:
```
import yfinance as yf

# Fetch GME stock data
```

```
gme_data = yf.download("GME", start="2020-01-01", end="2023-09-26")

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

# Save the dataframe (optional, as a CSV for example)
gme_data.to_csv("gme_data.csv", index=False)

# Display the first five rows
print(gme_data.head())
```

```
[*********************100%***********************]  1 of 1 completed
        Date    Open    High     Low   Close  Adj Close    Volume
0 2020-01-02  1.5350  1.6175  1.5175  1.5775     1.5775  17814400
1 2020-01-03  1.5525  1.5625  1.4600  1.4700     1.4700  14175600
2 2020-01-06  1.4500  1.4775  1.4000  1.4625     1.4625  13579200
3 2020-01-07  1.4425  1.4575  1.3600  1.3800     1.3800  20912000
4 2020-01-08  1.3725  1.4625  1.3525  1.4300     1.4300  22517600
```

In [39]:
```
pip install requests beautifulsoup4 pandas
```

```
Requirement already satisfied: requests in /opt/conda/lib/python3.11/site-package
s (2.31.0)
Requirement already satisfied: beautifulsoup4 in /opt/conda/lib/python3.11/site-p
ackages (4.12.3)
Requirement already satisfied: pandas in /opt/conda/lib/python3.11/site-packages
(2.2.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/conda/lib/python
3.11/site-packages (from requests) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.11/site-pac
kages (from requests) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/lib/python3.11/si
te-packages (from requests) (2.2.1)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.11/si
te-packages (from requests) (2024.6.2)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.11/site-pa
ckages (from beautifulsoup4) (2.5)
Requirement already satisfied: numpy>=1.23.2 in /opt/conda/lib/python3.11/site-pa
ckages (from pandas) (2.1.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/conda/lib/python3.1
1/site-packages (from pandas) (2.9.0)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.11/site-pac
kages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.11/site-p
ackages (from pandas) (2024.2)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.11/site-package
s (from python-dateutil>=2.8.2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [42]:
```
import requests
import pandas as pd
from bs4 import BeautifulSoup

# URL of the financial page containing GME revenue data
url = 'https://es.finance.yahoo.com/quote/GME/'

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

# Parse the HTML content
```

```
soup = BeautifulSoup(response.text, 'html.parser')

# Find the relevant table (assumes the revenue data is in a specific table)
table = soup.find('table')

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

# Extract table rows
for row in table.find_all('tr')[1:]:
    cols = row.find_all('td')
    if len(cols) >= 2:
        dates.append(cols[0].text.strip())
        revenues.append(cols[1].text.strip().replace('$', '').replace(',', ''))

# Create a DataFrame
gme_revenue = pd.DataFrame({
    'Date': pd.to_datetime(dates),
    'Revenue': pd.to_numeric(revenues, errors='coerce')
})

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

```
Empty DataFrame
Columns: [Date, Revenue]
Index: []
```

In [44]:

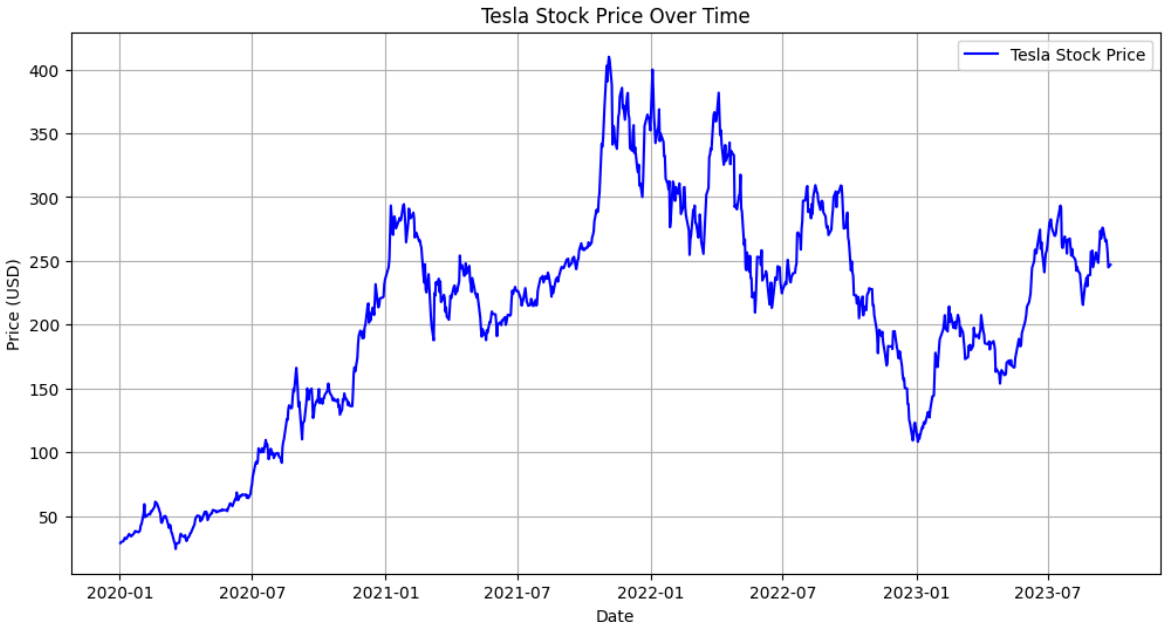
```
import yfinance as yf
import matplotlib.pyplot as plt

tesla_data = yf.download("TSLA", start="2020-01-01", end="2023-09-26")

def make_graph(data):
    plt.figure(figsize=(12, 6))
    plt.plot(data['Close'], label='Tesla Stock Price', color='blue')
    plt.title('Tesla Stock Price Over Time')
    plt.xlabel('Date')
    plt.ylabel('Price (USD)')
    plt.legend()
    plt.grid()
    plt.show()

make_graph(tesla_data)
```

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



In [46]:
```
import yfinance as yf
import matplotlib.pyplot as plt

gme_data = yf.download("GME", start="2020-01-01", end="2023-09-26")

def make_graph(data):
    plt.figure(figsize=(12, 6))
    plt.plot(data['Close'], label='GameStop Stock Price', color='green')
    plt.title('GameStop Stock Price Over Time')
    plt.xlabel('Date')
    plt.ylabel('Price (USD)')
    plt.legend()
    plt.grid()
    plt.show()

make_graph(gme_data)
```

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