Question 1:Use the yfinance to Extract Stock Data

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

In [1]:
```
!pip install yfinance
!mamba install bs4
!pip install nbformat
```

```
Requirement already satisfied: yfinance in c:\users\satta\anaconda3\lib\site-packages
(0.2.4)
Requirement already satisfied: pandas>=1.3.0 in c:\users\satta\anaconda3\lib\site-pac
kages (from yfinance) (1.5.3)
Requirement already satisfied: numpy>=1.16.5 in c:\users\satta\anaconda3\lib\site-pac
kages (from yfinance) (1.24.3)
Requirement already satisfied: requests>=2.26 in c:\users\satta\anaconda3\lib\site-pa
ckages (from yfinance) (2.29.0)
Requirement already satisfied: multitasking>=0.0.7 in c:\users\satta\anaconda3\lib\si
te-packages (from yfinance) (0.0.11)
Requirement already satisfied: lxml>=4.9.1 in c:\users\satta\anaconda3\lib\site-packa
ges (from yfinance) (4.9.2)
Requirement already satisfied: appdirs>=1.4.4 in c:\users\satta\anaconda3\lib\site-pa
ckages (from yfinance) (1.4.4)
Requirement already satisfied: pytz>=2022.5 in c:\users\satta\anaconda3\lib\site-pack
ages (from yfinance) (2022.7)
Requirement already satisfied: frozendict>=2.3.4 in c:\users\satta\anaconda3\lib\site
-packages (from yfinance) (2.3.8)
Requirement already satisfied: cryptography>=3.3.2 in c:\users\satta\anaconda3\lib\si
te-packages (from yfinance) (39.0.1)
Requirement already satisfied: beautifulsoup4>=4.11.1 in c:\users\satta\anaconda3\lib
\site-packages (from yfinance) (4.12.2)
Requirement already satisfied: html5lib>=1.1 in c:\users\satta\anaconda3\lib\site-pac
kages (from yfinance) (1.1)
Requirement already satisfied: soupsieve>1.2 in c:\users\satta\anaconda3\lib\site-pac
kages (from beautifulsoup4>=4.11.1->yfinance) (2.4)
Requirement already satisfied: cffi>=1.12 in c:\users\satta\anaconda3\lib\site-packag
es (from cryptography>=3.3.2->yfinance) (1.15.1)
Requirement already satisfied: six>=1.9 in c:\users\satta\anaconda3\lib\site-packages
(from html5lib>=1.1->yfinance) (1.16.0)
Requirement already satisfied: webencodings in c:\users\satta\anaconda3\lib\site-pack
ages (from html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\satta\anaconda3\lib
\site-packages (from pandas>=1.3.0->yfinance) (2.8.2)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\satta\anaconda3\l
ib\site-packages (from requests>=2.26->yfinance) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\satta\anaconda3\lib\site-pack
ages (from requests>=2.26->yfinance) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\satta\anaconda3\lib
\site-packages (from requests>=2.26->yfinance) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\satta\anaconda3\lib\sit
e-packages (from requests>=2.26->yfinance) (2023.5.7)
Requirement already satisfied: pycparser in c:\users\satta\anaconda3\lib\site-package
s (from cffi>=1.12->cryptography>=3.3.2->yfinance) (2.21)
'mamba' is not recognized as an internal or external command,
operable program or batch file.
```

```
Requirement already satisfied: nbformat in c:\users\satta\anaconda3\lib\site-packages
(5.7.0)
Requirement already satisfied: fastjsonschema in c:\users\satta\anaconda3\lib\site-pa
ckages (from nbformat) (2.16.2)
Requirement already satisfied: jsonschema>=2.6 in c:\users\satta\anaconda3\lib\site-p
ackages (from nbformat) (4.17.3)
Requirement already satisfied: jupyter-core in c:\users\satta\anaconda3\lib\site-pack
ages (from nbformat) (5.3.0)
Requirement already satisfied: traitlets>=5.1 in c:\users\satta\anaconda3\lib\site-pa
ckages (from nbformat) (5.7.1)
Requirement already satisfied: attrs>=17.4.0 in c:\users\satta\anaconda3\lib\site-pac
kages (from jsonschema>=2.6->nbformat) (22.1.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in c:\us
ers\satta\anaconda3\lib\site-packages (from jsonschema>=2.6->nbformat) (0.18.0)
Requirement already satisfied: platformdirs>=2.5 in c:\users\satta\anaconda3\lib\site
-packages (from jupyter-core->nbformat) (2.5.2)
Requirement already satisfied: pywin32>=300 in c:\users\satta\anaconda3\lib\site-pack
ages (from jupyter-core->nbformat) (305.1)
```

In [2]:
```
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 [3]:
```
def make_graph(stock_data, revenue_data, stock):
    fig = make_subplots(rows=2, cols=1, shared_xaxes=True, subplot_titles=("Historical
    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, infer_datetime
    fig.add_trace(go.Scatter(x=pd.to_datetime(revenue_data_specific.Date, infer_dateti
    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()
```

In [4]:
```
tesla = yf.Ticker("TSLA")
```

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 [6]:
```
tesla_data = tesla.history(period="max")
```

In [7]:
```
tesla_data.reset_index(inplace=True)
tesla_data.head()
```

Out[7]:

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2010-06-29 00:00:00-04:00	1.266667	1.666667	1.169333	1.592667	281494500	0.0	0.0
1	2010-06-30 00:00:00-04:00	1.719333	2.028000	1.553333	1.588667	257806500	0.0	0.0
2	2010-07-01 00:00:00-04:00	1.666667	1.728000	1.351333	1.464000	123282000	0.0	0.0
3	2010-07-02 00:00:00-04:00	1.533333	1.540000	1.247333	1.280000	77097000	0.0	0.0
4	2010-07-06 00:00:00-04:00	1.333333	1.333333	1.055333	1.074000	103003500	0.0	0.0

Question 2: Use Webscraping to Extract Tesla Revenue Data

In [8]:
```
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloper
html_data = requests.get(url).text
```

Parse the html data using beautiful_soup.

In [9]:
```
beautiful_soup = BeautifulSoup(html_data, "html.parser")
beautiful_soup.find_all('title')
```

Out[9]:
```
[<title>Tesla Revenue 2010-2022 | TSLA | MacroTrends</title>]
```

In [10]:
```
tesla_revenue = pd.DataFrame(columns=["Date", "Revenue"])

for row in beautiful_soup.find("tbody").find_all('tr'):
    col =row.find_all("td")
    date= col[0].text
    revenue = col[1].text

tesla_revenue = tesla_revenue.append({"Date":date,"Revenue":revenue}, ignore_index=Tru
```

```
C:\Users\satta\AppData\Local\Temp\ipykernel_3232\3386551880.py:8: FutureWarning: The
frame.append method is deprecated and will be removed from pandas in a future versio
n. Use pandas.concat instead.
  tesla_revenue = tesla_revenue.append({"Date":date,"Revenue":revenue}, ignore_index=
True)
```

In [11]:
```
tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace("$", "").str.replace('
```

```
C:\Users\satta\AppData\Local\Temp\ipykernel_3232\409785084.py:1: FutureWarning: The d
efault value of regex will change from True to False in a future version. In additio
n, single character regular expressions will *not* be treated as literal strings when
regex=True.
  tesla_revenue["Revenue"] = tesla_revenue['Revenue'].str.replace("$", "").str.replac
e(",", "")
```

In [12]:
```
tesla_revenue.dropna(inplace=True)

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

In [13]:
```
tesla_revenue.tail(5)
```

Out[13]:

	Date	Revenue
0	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 [14]:
```
GameStop = yf.Ticker("GME")
```

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 [19]:
```
gme_data = GameStop.history(period="max")
```

In [20]:
```
gme_data.reset_index(inplace=True)
gme_data.head()
```

Out[20]:

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2002-02-13 00:00:00-05:00	1.620129	1.693350	1.603296	1.691667	76216000	0.0	0.0
1	2002-02-14 00:00:00-05:00	1.712707	1.716074	1.670626	1.683250	11021600	0.0	0.0
2	2002-02-15 00:00:00-05:00	1.683250	1.687458	1.658001	1.674834	8389600	0.0	0.0
3	2002-02-19 00:00:00-05:00	1.666418	1.666418	1.578047	1.607504	7410400	0.0	0.0
4	2002-02-20 00:00:00-05:00	1.615920	1.662210	1.603296	1.662210	6892800	0.0	0.0

Question 4: Use Webscraping to Extract GME Revenue Data

In [21]:
```
url ="https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperS

html_data = requests.get(url).text
```

Parse the html data using beautiful_soup.

In [22]:
```
beautiful_soup = BeautifulSoup(html_data, "html.parser")
```

In [23]:
```
gme_revenue = pd.DataFrame(columns = ["Date","Revenue"])
for row in beautiful_soup.find("tbody").find_all('tr'):
    col = row.find_all("td")
    date = col[0].text
```

```
    revenue = col[1].text

gme_revenue= gme_revenue.append({"Date":date, "Revenue":revenue}, ignore_index=True)

gme_revenue["Revenue"] = gme_revenue["Revenue"].str.replace("$", "").str.replace(",",
```

```
C:\Users\satta\AppData\Local\Temp\ipykernel_3232\947018535.py:7: FutureWarning: The f
rame.append method is deprecated and will be removed from pandas in a future version.
Use pandas.concat instead.
  gme_revenue= gme_revenue.append({"Date":date, "Revenue":revenue}, ignore_index=Tru
e)
C:\Users\satta\AppData\Local\Temp\ipykernel_3232\947018535.py:9: FutureWarning: The d
efault value of regex will change from True to False in a future version. In additio
n, single character regular expressions will *not* be treated as literal strings when
regex=True.
  gme_revenue["Revenue"] = gme_revenue["Revenue"].str.replace("$", "").str.replace
(",", "")
```

In [24]:
```
tesla_revenue.dropna(inplace=True)
tesla_revenue = tesla_revenue[tesla_revenue['Revenue'] != ""]
gme_revenue.tail()
```

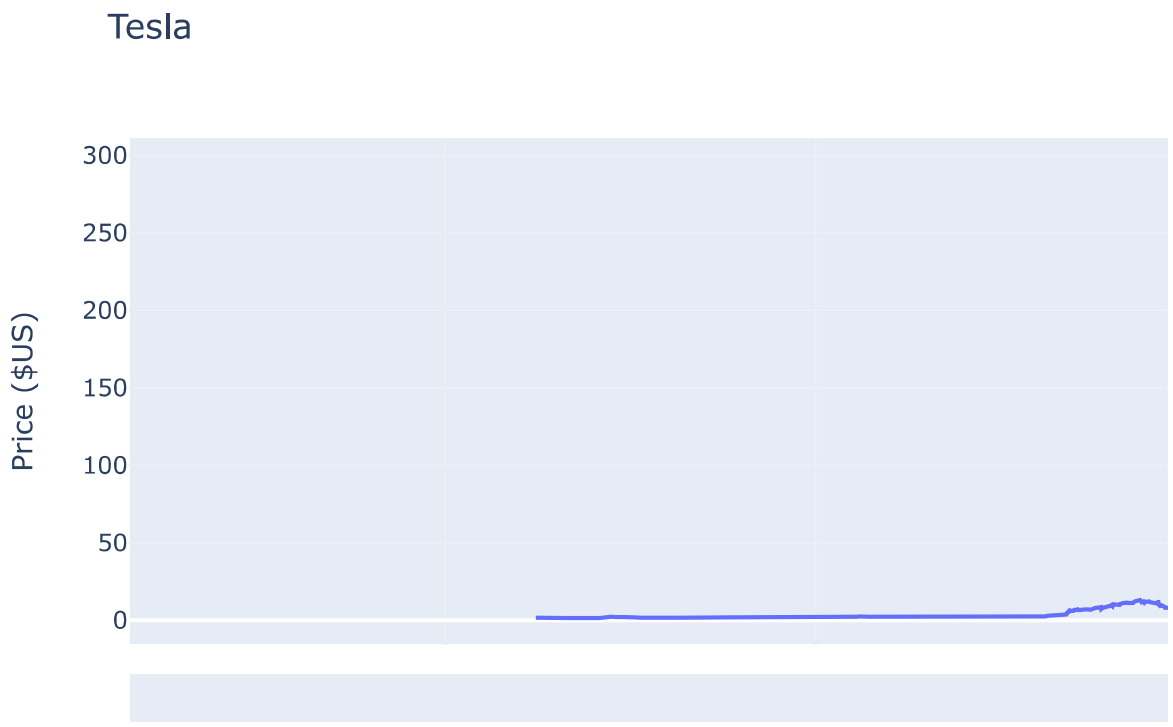
Out[24]:

	Date	Revenue
0	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. The structure to call the make_graph function is make_graph(tesla_data, tesla_revenue, 'Tesla'). Note the graph will only show data upto June 2021.

In [25]:
```
make_graph(tesla_data, tesla_revenue, 'Tesla')
```



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 [26]:
```
make_graph(gme_data, gme_revenue, 'GameStop')
```

In [27]:
```
import pandas as pd
import plotly.express as px
```

In [32]:
```
tesla_data = pd.read_csv('tesla_stock.csv')
tesla_revenue = pd.read_csv('tesla_revenue.csv')
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[32], line 1
----> 1 tesla_data = pd.read_csv('tesla_stock.csv')
      2 tesla_revenue = pd.read_csv('tesla_revenue.csv')

File ~\anaconda3\Lib\site-packages\pandas\util\_decorators.py:211, in deprecate_kwar
g.<locals>._deprecate_kwarg.<locals>.wrapper(*args, **kwargs)
    209     else:
    210         kwargs[new_arg_name] = new_arg_value
--> 211 return func(*args, **kwargs)

File ~\anaconda3\Lib\site-packages\pandas\util\_decorators.py:331, in deprecate_nonke
yword_arguments.<locals>.decorate.<locals>.wrapper(*args, **kwargs)
    325 if len(args) > num_allow_args:
    326     warnings.warn(
    327         msg.format(arguments=_format_argument_list(allow_args)),
    328         FutureWarning,
    329         stacklevel=find_stack_level(),
    330     )
--> 331 return func(*args, **kwargs)

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:950, in read_csv(file
path_or_buffer, sep, delimiter, header, names, index_col, usecols, squeeze, prefix, m
angle_dupe_cols, dtype, engine, converters, true_values, false_values, skipinitialspa
ce, skiprows, skipfooter, nrows, na_values, keep_default_na, na_filter, verbose, skip
_blank_lines, parse_dates, infer_datetime_format, keep_date_col, date_parser, dayfirs
t, cache_dates, iterator, chunksize, compression, thousands, decimal, lineterminator,
quotechar, quoting, doublequote, escapechar, comment, encoding, encoding_errors, dial
ect, error_bad_lines, warn_bad_lines, on_bad_lines, delim_whitespace, low_memory, mem
ory_map, float_precision, storage_options)
    935 kwds_defaults = _refine_defaults_read(
    936     dialect,
    937     delimiter,
   (...)
    946     defaults={"delimiter": ","},
    947 )
    948 kwds.update(kwds_defaults)
--> 950 return _read(filepath_or_buffer, kwds)

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:605, in _read(filepat
h_or_buffer, kwds)
    602 _validate_names(kwds.get("names", None))
    604 # Create the parser.
--> 605 parser = TextFileReader(filepath_or_buffer, **kwds)
    607 if chunksize or iterator:
    608     return parser

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:1442, in TextFileRead
er.__init__(self, f, engine, **kwds)
   1439     self.options["has_index_names"] = kwds["has_index_names"]
   1441 self.handles: IOHandles | None = None
-> 1442 self._engine = self._make_engine(f, self.engine)

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:1735, in TextFileRead
er._make_engine(self, f, engine)
   1733     if "b" not in mode:
   1734         mode += "b"
-> 1735 self.handles = get_handle(
   1736     f,
```

```
   1737     mode,
   1738     encoding=self.options.get("encoding", None),
   1739     compression=self.options.get("compression", None),
   1740     memory_map=self.options.get("memory_map", False),
   1741     is_text=is_text,
   1742     errors=self.options.get("encoding_errors", "strict"),
   1743     storage_options=self.options.get("storage_options", None),
   1744 )
   1745 assert self.handles is not None
   1746 f = self.handles.handle

File ~\anaconda3\Lib\site-packages\pandas\io\common.py:856, in get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text, errors, storage_options)
    851 elif isinstance(handle, str):
    852     # Check whether the filename is to be opened in binary mode.
    853     # Binary mode does not support 'encoding' and 'newline'.
    854     if ioargs.encoding and "b" not in ioargs.mode:
    855         # Encoding
--> 856         handle = open(
    857             handle,
    858             ioargs.mode,
    859             encoding=ioargs.encoding,
    860             errors=errors,
    861             newline="",
    862         )
    863     else:
    864         # Binary mode
    865         handle = open(handle, ioargs.mode)

FileNotFoundError: [Errno 2] No such file or directory: 'tesla_stock.csv'
```

In [ ]: