```
import pandas as pd
from numpy import random
import datetime
from datetime import date, timedelta

import plotly.graph_objects as go
import plotly.express as px
import plotly.io as pio

pio.templates.default = "plotly_white"

import statistics as st
from statsmodels.stats import weightstats as stests
from scipy.stats import ttest_1samp

import scipy.stats
import io
```

Task 1

```
# 1)  Read the data and display the first 100 rows from the data
data = pd.read_excel("/content/TWTR.xlsx")
print(data.head(100))
```

```
         Date       Open       High        Low      Close  Adj Close  \
0  2013-11-07  45.099998  50.090000  44.000000  44.900002  44.900002
1  2013-11-08  45.930000  46.939999  40.685001  41.650002  41.650002
2  2013-11-11  40.500000  43.000000  39.400002  42.900002  42.900002
3  2013-11-12  43.660000  43.779999  41.830002  41.900002  41.900002
4  2013-11-13  41.029999  42.869999  40.759998  42.599998  42.599998
..        ...        ...        ...        ...        ...        ...
95 2014-03-27  45.090000  46.400002  43.310001  46.320000  46.320000
96 2014-03-28  46.650002  47.340000  45.700001  47.299999  47.299999
97 2014-03-31  47.549999  47.750000  46.430000  46.669998  46.669998
98 2014-04-01  46.709999  47.590000  46.180000  46.980000  46.980000
99 2014-04-02  47.400002  47.439999  45.509998  45.730000  45.730000

          Volume
0   117701670.0
1    27925307.0
2    16113941.0
3     6316755.0
4     8688325.0
..          ...
95   15507597.0
96    9610491.0
97    5794497.0
98    6916147.0
99    7911260.0

[100 rows x 7 columns]
```

```
# 2)    Give the column insights
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2264 entries, 0 to 2263
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       2264 non-null   datetime64[ns]
 1   Open       2259 non-null   float64
 2   High       2259 non-null   float64
 3   Low        2259 non-null   float64
 4   Close      2259 non-null   float64
 5   Adj Close  2259 non-null   float64
 6   Volume     2259 non-null   float64
dtypes: datetime64[ns](1), float64(6)
memory usage: 123.9 KB
None
```

```
# 3)    Check whether this dataset contains any null values or not if it is there then remove the null values from it

print(data.isnull().sum())
```

```
Date         0
Open         5
High         5
Low          5
Close        5
Adj Close    5
Volume       5
dtype: int64
```

```
# Drop Null values from data

print(data.dropna(inplace= True))
```

```
None
```

```
# 4)    Find the statistical description of the data.

print(data.describe())
```

```
              Open         High          Low        Close    Adj Close  \
count  2259.000000  2259.000000  2259.000000  2259.000000  2259.000000
mean     36.020286    36.699881    35.339465    36.003625    36.003625
std      14.118463    14.372057    13.828724    14.089989    14.089989
min      13.950000    14.220000    13.725000    14.010000    14.010000
25%      25.550000    26.215001    24.912501    25.410000    25.410000
50%      35.419998    36.099998    34.820000    35.490002    35.490002
75%      44.205000    45.015000    43.327501    44.135000    44.135000
max      78.360001    80.750000    76.050003    77.629997    77.629997

             Volume
count  2.259000e+03
mean   2.175186e+07
std    1.909988e+07
min    0.000000e+00
25%    1.233530e+07
50%    1.691305e+07
75%    2.428082e+07
max    2.692131e+08
```

```
# 5)    Find the missing values in the data
print(data.isna())
```

```
          Date   Open   High    Low  Close  Adj Close  Volume
0     False  False  False  False  False      False   False
1     False  False  False  False  False      False   False
2     False  False  False  False  False      False   False
3     False  False  False  False  False      False   False
4     False  False  False  False  False      False   False
...     ...    ...    ...    ...    ...        ...     ...
2254  False  False  False  False  False      False   False
2255  False  False  False  False  False      False   False
2256  False  False  False  False  False      False   False
2257  False  False  False  False  False      False   False
2258  False  False  False  False  False      False   False

[2259 rows x 7 columns]
```

Task 2

```
# 6)    Give me the Z-test O/R T-test over High, low, and close columns and see if the null hypothesis gets rejected or accepted

# Z-Test

high = data['High']
print("Data =", high)

high_mean = st.mean(high)
print("Mean Data :", high_mean)

high_stdv = st.stdev(high)
print(high_stdv)


ztest, pval = stests.ztest(high, value = 30)
print("Z-Test Score :", ztest)

print("P-Value :", pval)


if pval<0.05:
  print("Reject Null Hypothesis")
else:
  print("Accept Null Hypothesis")
```

```
Data = 0       50.090000
1       46.939999
2       43.000000
3       43.779999
4       42.869999
          ...
2254    50.750000
2255    51.860001
2256    53.180000
2257    53.500000
2258    54.000000
Name: High, Length: 2259, dtype: float64
Mean Data : 36.69988069278442
14.372056692309659
Z-Test Score : 22.15676048174423
P-Value : 8.978177265940086e-109
Reject Null Hypothesis
```

```
# T-Test

high = data['High']
print("Data =", high)

high_mean = st.mean(high)
print("Mean Data :", high_mean)

high_stdv = st.stdev(high)
print(high_stdv)


t_test, pval = ttest_1samp(high,50)
print("T-test Score :", t_test)

print("P-Value", pval)


if pval<0.05:
  print("Reject Null Hypothesis")
else:
  print("Accept Null Hypothesis")
```

```
Data = 0       50.090000
1       46.939999
2       43.000000
3       43.779999
4       42.869999
          ...
2254    50.750000
2255    51.860001
2256    53.180000
2257    53.500000
2258    54.000000
Name: High, Length: 2259, dtype: float64
Mean Data : 36.69988069278442
14.372056692309659
T-test Score : -43.984000817502285
P-Value 9.208373050426324e-306
Reject Null Hypothesis
```

```
# 7)    By using ANOVA find the Fvalue and Pvalue from the data and see its Acceptance and rejection of the Null hypothesis

grp1 = data['High']
grp2 = data['Low']

print(grp1.head(5))
print(grp2.head(5))

Fval, Pval = scipy.stats.f_oneway(grp1,grp2)

print("F-Value", Fval)
print("P-Value", Pval)

if Pval<0.05:
  print("Reject Null Hypothesis")
else:
  print("Accept Null Hypothesis")
```

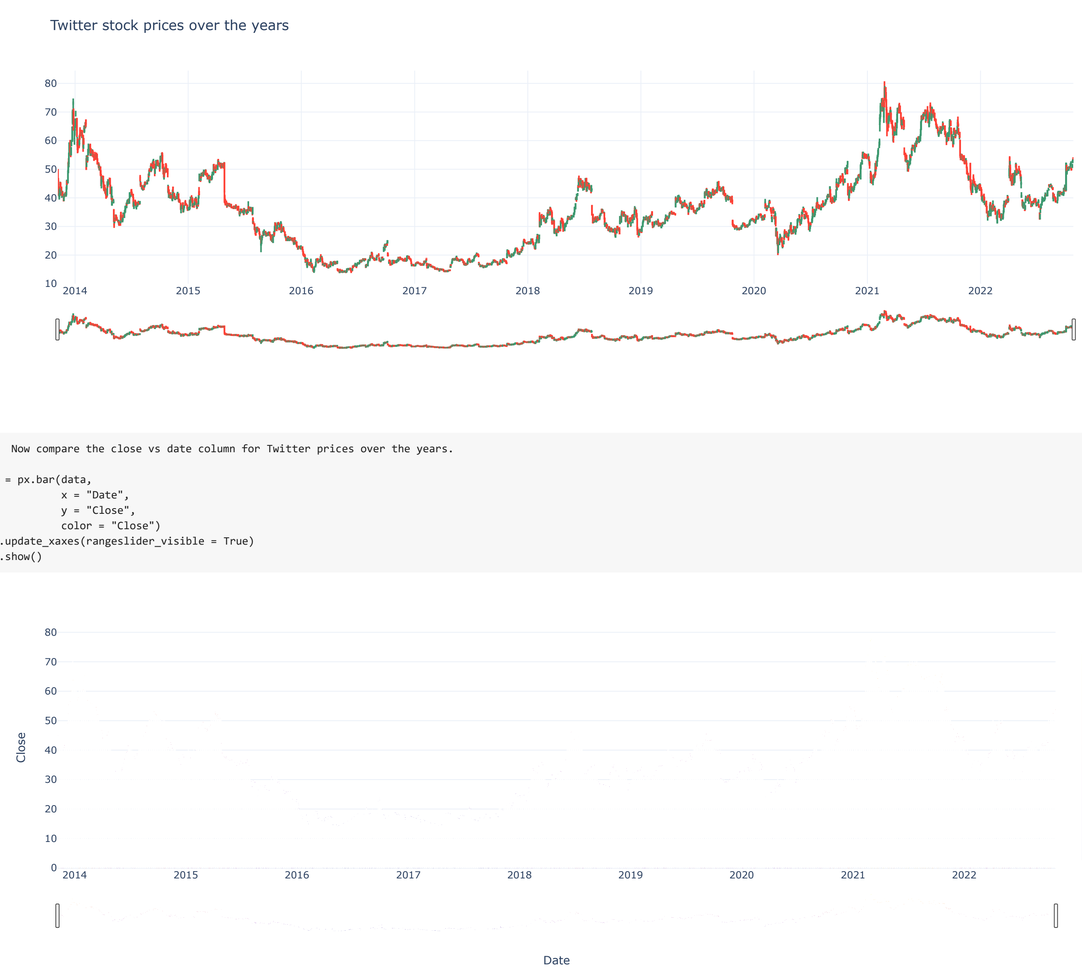
```
0    50.090000
1    46.939999
2    43.000000
3    43.779999
4    42.869999
Name: High, dtype: float64
0    44.000000
1    40.685001
2    39.400002
3    41.830002
4    40.759998
Name: Low, dtype: float64
F-Value 10.510084060032649
P-Value 0.0011958722702637046
Reject Null Hypothesis
```

```
# 8)    Check if the data is dependent or independent by using the chi-square method.


datas = {'High':data['High'], 'Low':data['Low']}
print(datas)

chisq , pval = scipy.stats.chisquare(datas['Low'])

alpha = 0.05

print("Chi-Square:", chisq)
print("P-Value", pval)

if pval<alpha:
  print("Depedent")
else:
  print("Indepedent")
```

```
{'High': 0       50.090000
1       46.939999
2       43.000000
3       43.779999
4       42.869999
          ...
2254    50.750000
2255    51.860001
2256    53.180000
2257    53.500000
2258    54.000000
Name: High, Length: 2259, dtype: float64, 'Low': 0       44.000000
1       40.685001
2       39.400002
3       41.830002
4       40.759998
          ...
2254    49.549999
2255    50.520000
2256    52.200001
2257    52.770000
2258    53.700001
Name: Low, Length: 2259, dtype: float64}
Chi-Square: 12218.788864355833
P-Value 0.0
Depedent
```

Task 3

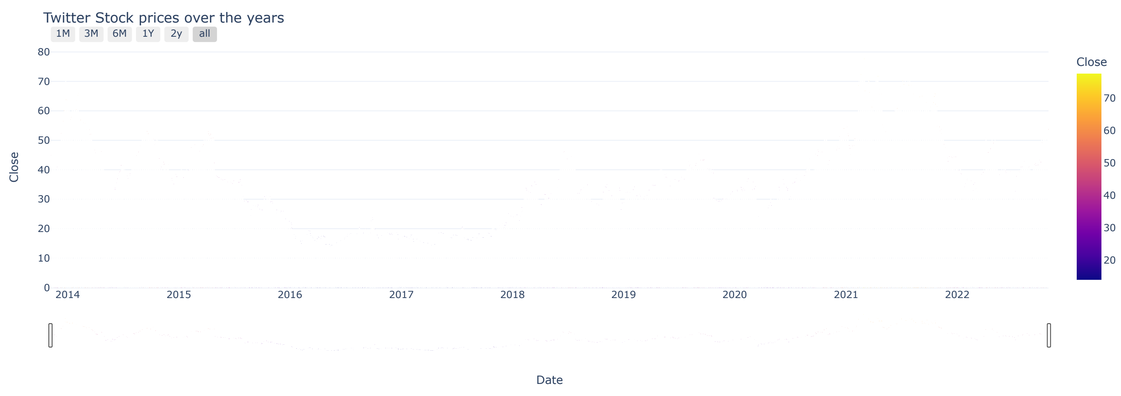
```
# 11)   Show the Twitter stock prices over the years  and give a conclusion.

figure = go.Figure(data=[go.Candlestick(x= data["Date"],
                                        open= data['Open'],
                                        high= data['High'],
                                        low = data['Low'],
                                        close = data['Close'])])

figure.update_layout(title = "Twitter stock prices over the years", xaxis_rangeslider_visible = True)
figure.show()
```



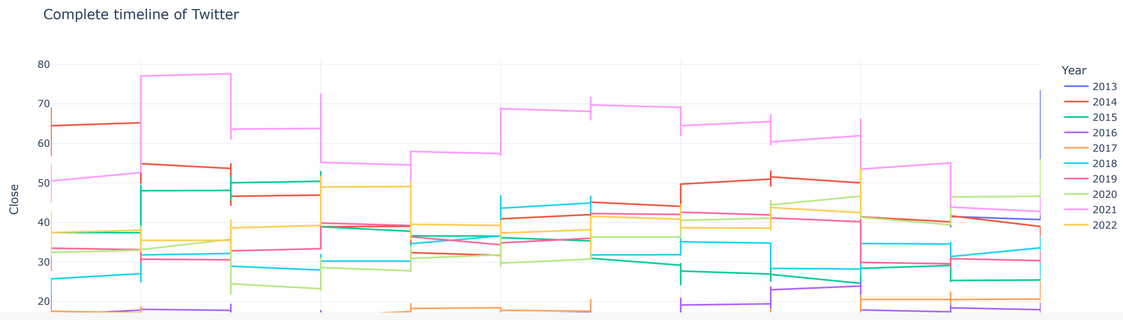

```
# 12)   Now compare the close vs date column for Twitter prices over the years.

figure = px.bar(data,
                x = "Date",
                y = "Close",
                color = "Close")
figure.update_xaxes(rangeslider_visible = True)
figure.show()
```



```
# 13)   Assign buttons to control time periods.  Add the  buttons to analyze the stock prices of Twitter in different time periods:

figure = px.bar(data, x = "Date", y = "Close", color = "Close")

figure.update_xaxes(rangeslider_visible = True)

figure.update_layout(title = "Twitter Stock prices over the years")

figure.update_xaxes(
    rangeselector = dict(
        buttons = list(
            [
                dict(count = 1, label = "1M", step = "month", stepmode = "backward"),
                dict(count = 3, label = "3M", step = "month", stepmode = "backward"),
                dict(count = 6, label = "6M", step = "month", stepmode = "backward"),
                dict(count = 1, label = "1Y", step = "year", stepmode = "backward"),
                dict(count = 2, label = "2y", step = "year", stepmode = "backward"),
                dict(step = "all")
            ]
        )
    )
)
```




```
# 14)   Give the complete timeline of Twitter in the stock market. (Line Graph)

data["Date"] = pd.to_datetime(data["Date"], format = '%Y-%m-%d')
data["Year"] = data['Date'].dt.year
data["Month"] = data['Date'].dt.month

figure = px.line(data,
                 x= "Month",
                 y = "Close",
                 color = "Year",
                 title = "Complete timeline of Twitter")
figure.show()
```




```
figure = px.box(data, y = "High")
figure.show()
```