```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


from datetime import datetime  # to access datetime
import scipy.stats as stats

import plotly.express as px # for interactive plotting
import plotly.graph_objects as go # for interactive plotting

# set the graphics style initially to defaul
plt.style.use('default')
```

The imported dataset 'EthnicDist" shows the distribution of population by gender and ethnic groups for counties in the United States

```python
df2=pd.read_csv('EthnicDist.csv')
df2.head()
```

```
    FIPS   STNAME          CTYNAME  TOT_POP  TOT_MALE  TOT_FEMALE
WA_MALE  \
0  18049  Indiana    Fulton County    20737     10369       10368
9985
1  18051  Indiana    Gibson County    33458     16642       16816
15873
2  18053  Indiana     Grant County    69330     33282       36048
29587
3  18055  Indiana    Greene County    32940     16479       16461
16179
4  18057  Indiana  Hamilton County   289495    141103      148392
125675

   WA_FEMALE  NHWA_MALE  NHWA_FEMALE  NHWhite_Alone  Not_NHWhite_Alone
\
0      10020       9561         9627          19188               1549

1      16117      15648        15955          31603               1855

2      32460      28353        31398          59751               9579

3      16167      16029        15999          32028                912

4     131785     120979       127105         248084              41411


  MinorityMinority MinorityPCT  Black BlackPCT  Hispanic HispanicPCT
0               No       7.47%    170       1%       965       4.65%
1               No       5.54%    667       2%       476       1.42%
```

```
2                    No      13.82%   4936         7%       2656          3.83%
3                    No       2.77%     82         0%        351          1.07%
4                    No      14.30%  11332         4%      10548          3.64%
```

Q1. Create two new variables in the dataframe that meaures males and females as a percentage of total population

```
df2['PCTMALE']=df2['TOT_MALE']/df2['TOT_POP']
df2['PCTFEMALE']=df2['TOT_FEMALE']/df2['TOT_POP']
df2.head()
```

```
    FIPS   STNAME          CTYNAME  TOT_POP  TOT_MALE  TOT_FEMALE
WA_MALE  \
0  18049  Indiana    Fulton County    20737     10369       10368
9985
1  18051  Indiana    Gibson County    33458     16642       16816
15873
2  18053  Indiana     Grant County    69330     33282       36048
29587
3  18055  Indiana    Greene County    32940     16479       16461
16179
4  18057  Indiana  Hamilton County   289495    141103      148392
125675

   WA_FEMALE  NHWA_MALE  NHWA_FEMALE  NHWhite_Alone  Not_NHWhite_Alone
\
0      10020       9561         9627          19188               1549

1      16117      15648        15955          31603               1855

2      32460      28353        31398          59751               9579

3      16167      16029        15999          32028                912

4     131785     120979       127105         248084              41411


  MinorityMinority MinorityPCT  Black BlackPCT  Hispanic
HispanicPCT  \
0                No       7.47%    170       1%       965          4.65%

1                No       5.54%    667       2%       476          1.42%

2                No      13.82%   4936       7%      2656          3.83%

3                No       2.77%     82       0%       351          1.07%

4                No      14.30%  11332       4%     10548          3.64%
```

```
   PCTMALE  PCTFEMALE
0  0.500024   0.499976
1  0.497400   0.502600
2  0.480052   0.519948
3  0.500273   0.499727
4  0.487411   0.512589
```

Q2. Create a new dataframe, df2MaFe, that calculates the average percentage distribution of males and females by States. Reset the index.

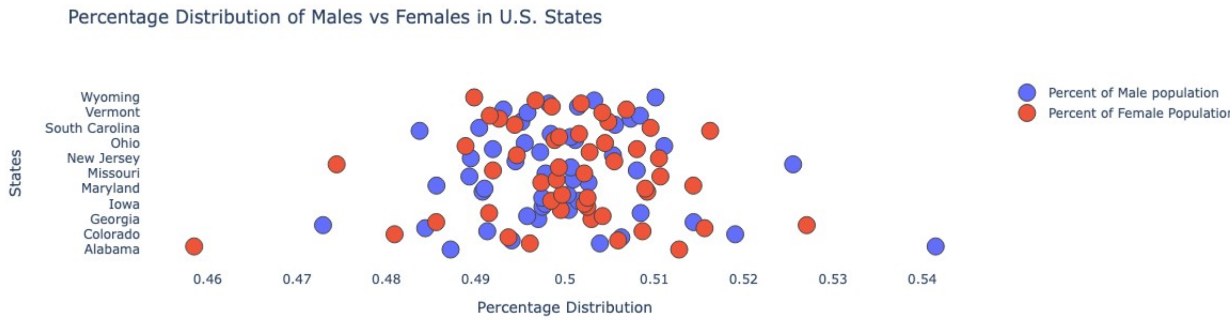
```
df2MaFe=df2.groupby(['STNAME'])
[['PCTMALE','PCTFEMALE']].mean().reset_index()
df2MaFe.head()
```

```
       STNAME   PCTMALE  PCTFEMALE
0     Alabama  0.487206   0.512794
1      Alaska  0.541509   0.458491
2     Arizona  0.503916   0.496084
3    Arkansas  0.494047   0.505953
4  California  0.506320   0.493680
```

Q3. Using Plotly graph-object, create a scatterplot of distribution of Males and Females in each State as shown below. Include the Title, axes-labels and legends. Make the graph background white.

```
fig = go.Figure()

# add scatter dots for percentage of voting age population
fig.add_trace(go.Scatter(
  x=df2MaFe['PCTMALE'],
  y=df2MaFe['STNAME'], name='Percent of Male population',
  ))

fig.add_trace(go.Scatter(
     x=df2MaFe['PCTFEMALE'],
    y=df2MaFe['STNAME'],
     name='Percent of Female Population',
    ))

fig.update_traces(mode='markers',
                 marker=dict(line_width=1,symbol='circle',size=16))

fig.update_layout(
        plot_bgcolor='white',
)

fig.update_layout(title="Percentage Distribution of Males vs Females
in U.S. States",
                  xaxis_title="Percentage Distribution",
```

```
                 yaxis_title="States")

fig.show()
```



The datasets apples3 and Google 3 show the Date and adjusted closing prices for Apple and Google stocks.

```
Apple=pd.read_excel('apple3.xlsx',parse_dates=['Date'])
Apple.head()
```

	Date	Adj Close	Volume
0	2021-01-04	127.331726	143301900
1	2021-01-05	128.905975	97664900
2	2021-01-06	124.566818	155088000
3	2021-01-07	128.817459	109578200
4	2021-01-08	129.929291	105158200

```
Google=pd.read_excel('Google3.xlsx',parse_dates=['Date'])
Google.head()
```

	Date	Adj Close	Volume
0	2021-01-04	86.412003	38038000
1	2021-01-05	87.045998	22906000
2	2021-01-06	86.764503	52042000
3	2021-01-07	89.362503	45300000
4	2021-01-08	90.360497	41012000

Q4. Merge the apple and google datasets

```
Stocks=Apple.merge(Google,on='Date',suffixes=("_aapl","_goog"))
Stocks.head()
```

	Date	Adj Close_aapl	Volume_aapl	Adj Close_goog	Volume_goog
0	2021-01-04	127.331726	143301900	86.412003	38038000
1	2021-01-05	128.905975	97664900	87.045998	22906000
2	2021-01-06	124.566818	155088000	86.764503	52042000
3	2021-01-07	128.817459	109578200	89.362503	45300000
4	2021-01-08	129.929291	105158200	90.360497	41012000

Q5.Generate the summary statistics for both Apple and Google closing prices

```
Stocks[['Adj Close_goog','Adj Close_aapl']].describe()
```

	Adj Close_goog	Adj Close_aapl
count	253.000000	253.000000
mean	125.607935	139.368922
std	18.369859	14.888355
min	86.412003	114.662361
25%	111.277496	126.908607
50%	129.177002	139.778534
75%	142.414993	147.223602
max	150.709000	180.190964

Q6. Slice out Apple stocks with adjusted closing prices less than $150

```
Stocks.loc[(Stocks['Adj Close_aapl'] < 150)].head()
```

	Date	Adj Close_aapl	Volume_aapl	Adj Close_goog	Volume_goog
0	2021-01-04	127.331726	143301900	86.412003	38038000
1	2021-01-05	128.905975	97664900	87.045998	22906000
2	2021-01-06	124.566818	155088000	86.764503	52042000
3	2021-01-07	128.817459	109578200	89.362503	45300000
4	2021-01-08	129.929291	105158200	90.360497	41012000

Q7.Using Plotly, generate line graphs for Apple and Google adjusted closing prices. Show the rangeslider, but don't show the gridlines.Label the graph as shown below

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=Stocks['Date'], y=Stocks['Adj
Close_aapl'],mode='lines', name='Apple'))

fig.add_trace(go.Scatter(x=Stocks['Date'], y=Stocks['Adj Close_goog'],
                    mode='lines',
                    name='Google'))

fig.update_xaxes(rangeslider_visible = True)

fig.update_layout(xaxis=dict(showline=True,showgrid=False),
  yaxis=dict(
        showgrid=False,
        showline=False,
        showticklabels=False),
  legend=dict(title='Stocks'),)


fig.update_layout(title= 'Apple vs Google Closing Prices',
                  xaxis_title='Day',
                  yaxis_title='Price')
```

```
fig.show()
```

The dataset AIrlines3 shows the delay time for departures for United (UA) and American Airlines (AA) for select days in June 2023

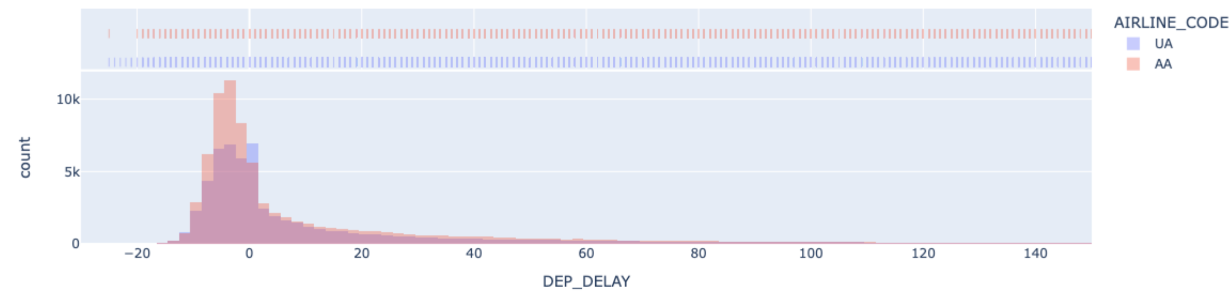
```
FL=pd.read_excel('Airline3.xlsx')
FL.head()

     FL_DATE AIRLINE_CODE  FL_NUMBER ORIGIN        DEST_CITY
DEP_DELAY
0 2023-06-27           UA       2098    IAD      Orlando, FL
329
1 2023-06-27           UA       2097    SFO       Newark, NJ
0
2 2023-06-27           UA       2096    LAX       Newark, NJ
0
3 2023-06-27           UA       2096    SFO  Los Angeles, CA
4
4 2023-06-27           UA       2095    ORD  Los Angeles, CA
20
```

Q8.Calculate the summary statistics for the departure delay times for United and American Airline

```
FLSUMM=FL.groupby('AIRLINE_CODE')['DEP_DELAY'].describe()
FLSUMM

                count       mean        std   min  25%  50%   75%
max
AIRLINE_CODE

AA            80416.0  26.439154  98.341958 -25.0 -5.0  0.0  21.0
3695.0
UA            62395.0  24.716644  73.648722 -25.0 -4.0  0.0  21.0
1549.0
```

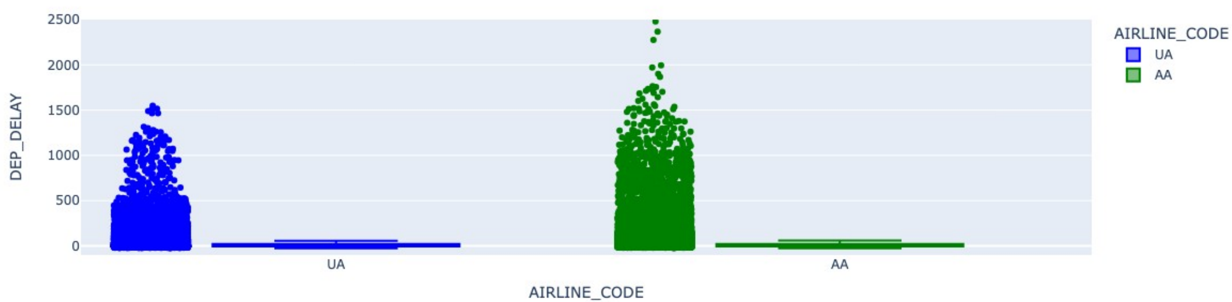
Q9. Using Plotly, generate a histogram for the departure delays for American and United airlines and include the 'rug' plot. Set the opacity to 0.35 and the x-axis range from -30 to 150

```
#Histogram with rug plot
fig = px.histogram(FL, x="DEP_DELAY",
color="AIRLINE_CODE",marginal='rug') #or box, violin
fig.update_layout(barmode='overlay')
fig.update_traces(opacity=0.35)
fig.update_xaxes(range=[-30, 150])

fig.show()
```



Q10. Using Plotly generate box-plots to show the departure delay times for American and United Airlines. Include all data-points and use 'Blue' and 'Green'to distinguish the two airlines. Set the y-axis range to -100 to 2500.

```
import plotly.express as px
fig = px.box(FL, x="AIRLINE_CODE", y="DEP_DELAY", points='all',
color="AIRLINE_CODE", color_discrete_sequence=['blue', 'green'])
fig.update_yaxes(range=[-100, 2500])

fig.show()
```



Q.11 Is there a statistical difference between average departure dela time for American and United Airlines? Run a two-sample t-test.

```
FL1=FL[FL['AIRLINE_CODE']=='UA']['DEP_DELAY']
FL2=FL[FL['AIRLINE_CODE']=='AA']['DEP_DELAY']

stats.ttest_ind(a=FL1, b=FL2,equal_var=True)

TtestResult(statistic=-3.652126687277148,
pvalue=0.0002601703909797832, df=142809.0)
```

Q12. If Federal Aviation was considering imposing a penalty on the airlines for any departure delays more than 50 minutes, what proportion of American and United fights will be penalized? Create a new variable, 'PenDel' that would classify a fight as being 'Penalized' or non penalized ('Non-Pen').

```
FL['PenDel']=['Penalized' if i >=50  else 'Non-Pen' for i in
FL['DEP_DELAY']]
FL.head()

     FL_DATE AIRLINE_CODE  FL_NUMBER ORIGIN        DEST_CITY
DEP_DELAY  \
0 2023-06-27           UA       2098    IAD      Orlando, FL
329
1 2023-06-27           UA       2097    SFO       Newark, NJ
0
2 2023-06-27           UA       2096    LAX       Newark, NJ
0
3 2023-06-27           UA       2096    SFO  Los Angeles, CA
4
4 2023-06-27           UA       2095    ORD  Los Angeles, CA
20

      PenDel
0  Penalized
1    Non-Pen
2    Non-Pen
3    Non-Pen
4    Non-Pen

FL['PenDel'].value_counts(normalize=True)

PenDel
Non-Pen      0.850761
Penalized    0.149239
Name: proportion, dtype: float64
```

The dataset WHR2 is drawn from the World Happiness Report.

```
whr=pd.read_excel('WHR2.xlsx')
whr.head()
```

```
   Unnamed: 0 Country name  Year  Life Ladder  Log GDP per capita  \
0           0  Afghanistan  2008     3.723590            7.168690   
1           1  Afghanistan  2009     4.401778            7.333790   
2           2  Afghanistan  2010     4.758381            7.386629   
3           3  Afghanistan  2011     3.831719            7.415019   
4           4  Afghanistan  2012     3.782938            7.517126   

   Social support  Healthy life expectancy at birth  \
0        0.450662                         50.799999   
1        0.552308                         51.200001   
2        0.539075                         51.599998   
3        0.521104                         51.919998   
4        0.520637                         52.240002   

   Freedom to make life choices  Generosity  Perceptions of corruption
\
0                      0.718114    0.177889                   0.881686

1                      0.678896    0.200178                   0.850035

2                      0.600127    0.134353                   0.706766

3                      0.495901    0.172137                   0.731109

4                      0.530935    0.244273                   0.775620


   Positive affect  Negative affect  Confidence in national government
\
0         0.517637         0.258195                           0.612072

1         0.583926         0.237092                           0.611545

2         0.618265         0.275324                           0.299357

3         0.611387         0.267175                           0.307386

4         0.710385         0.267919                           0.435440


   Democratic Quality  Delivery Quality  \
0           -1.929690         -1.655084   
1           -2.044093         -1.635025   
2           -1.991810         -1.617176   
3           -1.919018         -1.616221   
4           -1.842996         -1.404078   

   Standard deviation of ladder by country-year  \
0                                      1.774662   
1                                      1.722688   
2                                      1.878622
```

```
3                                     1.785360
4                                     1.798283

   Standard deviation/Mean of ladder by country-year
0                                          0.476600
1                                          0.391362
2                                          0.394803
3                                          0.465942
4                                          0.475367
```

```
whr.columns
```

```
Index(['Unnamed: 0', 'Country name', 'Year', 'Life Ladder',
       'Log GDP per capita', 'Social support',
       'Healthy life expectancy at birth', 'Freedom to make life
choices',
       'Generosity', 'Perceptions of corruption', 'Positive affect',
       'Negative affect', 'Confidence in national government',
       'Democratic Quality', 'Delivery Quality',
       'Standard deviation of ladder by country-year',
       'Standard deviation/Mean of ladder by country-year'],
      dtype='object')
```

Q13.Extract a subset of variables from the dataframe to include 'Life Ladder', 'Log GDP per capita','Healthy life expectancy at birth','Generosity','Democratic Quality'and store them in a new dataframe whr_core

```
whr_core=whr[['Life Ladder', 'Log GDP per capita','Healthy life
expectancy at birth','Generosity','Democratic Quality']]
whr_core.head()
```

```
   Life Ladder  Log GDP per capita  Healthy life expectancy at
birth  \
0     3.723590            7.168690                         50.799999

1     4.401778            7.333790                         51.200001

2     4.758381            7.386629                         51.599998

3     3.831719            7.415019                         51.919998

4     3.782938            7.517126                         52.240002


   Generosity  Democratic Quality
0    0.177889           -1.929690
1    0.200178           -2.044093
2    0.134353           -1.991810
3    0.172137           -1.919018
4    0.244273           -1.842996
```

Q14. In the World Happiness Report, the Cantril 'life ladder' represents a measure of 'happiness' where top of the ladder represents the best possible life for a country's citizen and the bottom of the ladder represents the worst possible life. What are the factors that determine "happiness? Run a multiple regression model to test if Life Ladder (the dependent variable) is affected by 'Log GDP per capita','Healthy life expectancy at birth','Generosity','Democratic Quality.

```
import statsmodels.api as sm

explanatory_var=['Log GDP per capita','Healthy life expectancy at
birth', 'Generosity','Democratic Quality']
x=whr[explanatory_var]
y=whr['Life Ladder']
x=sm.add_constant(x)
model=sm.OLS(y,x,missing='drop')
results=model.fit()
results.params
print(results.summary())
```

```
                            OLS Regression Results

======================================================================
========
Dep. Variable:            Life Ladder   R-squared:
0.678
Model:                            OLS   Adj. R-squared:
0.678
Method:                 Least Squares   F-statistic:
780.2
Date:                Mon, 04 Dec 2023   Prob (F-statistic):
0.00
Time:                        18:00:45   Log-Likelihood:
-1437.3
No. Observations:                1484   AIC:
2885.
Df Residuals:                    1479   BIC:
2911.
Df Model:                           4

Covariance Type:            nonrobust

======================================================================
==============================
                                       coef    std err          t
P>|t|      [0.025      0.975]
----------------------------------------------------------------------
------------------------------
const                               -1.0228      0.184     -5.554
0.000      -1.384      -0.662
Log GDP per capita                   0.4701      0.027     17.414
0.000       0.417       0.523
```

```
Healthy life expectancy at birth     0.0339      0.004      8.483
0.000       0.026       0.042
Generosity                           1.2044      0.102     11.753
0.000       1.003       1.405
Democratic Quality                   0.1774      0.026      6.947
0.000       0.127       0.227
======================================================================
========
Omnibus:                        8.574   Durbin-Watson:
0.543
Prob(Omnibus):                  0.014   Jarque-Bera (JB):
8.702
Skew:                          -0.183   Prob(JB):
0.0129
Kurtosis:                       2.917   Cond. No.
716.
======================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

Q15.Identify which variables are significant at an alpha of 0.05.

Q16. Based on your model, what is the effect on the Life Ladder if Generosity increased by 1 unit?