Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1hMDbMa71t2ahwmeX7eHweQEGjbZihJvN"""

# Imports

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

import matplotlib.colors as mcolors

import random

from google.colab import drive

drive.mount('/content/drive')

df = pd.concat([pd.read\_csv("/content/drive/MyDrive/DAT565/Data/life-expectancy-at-birth-oecd.csv")])

years\_to\_filter = [2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022]

df\_2018 = df[df['Year'].isin(years\_to\_filter)]

df\_2018.count()

print(df\_2018)

**# Task 1**

import pandas as pd

import numpy as np

df = pd.read\_csv("life-expectancy-at-birth-oecd.csv")

filtered\_df = df[(df['Year'] >= 2015) & (df['Year'] <= 2022)]

LE\_mean = filtered\_df['Life expectancy'].mean()

LE\_std = filtered\_df['Life expectancy'].std()

one\_std\_above\_mean = LE\_mean + LE\_std

display("Mean", LE\_mean)

display("Standard deviation", LE\_std)

display("one standard deviation above mean", one\_std\_above\_mean)

# Load the data from the CSV file

df = pd.read\_csv("life-expectancy-at-birth-oecd.csv")

# Filter the data for the years 2015-2022

filtered\_df = df[(df['Year'] >= 2015) & (df['Year'] <= 2022)]

# Calculate the mean and standard deviation of 'Life expectancy' for the selected years

LE\_mean = filtered\_df['Life expectancy'].mean()

LE\_std = filtered\_df['Life expectancy'].std()

# Filter the data based on the condition (Life expectancy > mean + std)

Life\_Expectancy = filtered\_df[filtered\_df['Life expectancy'] > (LE\_mean + LE\_std)]

Life\_Expectancy.to\_csv('life\_expectancy\_above\_mean.csv', index=False)

# Display the countries with life expectancy higher than one standard deviation above the mean

display(Life\_Expectancy)

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Load the data from the CSV file

df = pd.read\_csv("life-expectancy-at-birth-oecd.csv")

df= df.dropna(subset =['Life expectancy'])

# Filter the data for the years 2015-2022

filtered\_df = df[(df['Year'] >= 2015) & (df['Year'] <= 2022)]

# Calculate the mean and standard deviation of 'Life expectancy' for the selected years

LE\_mean = filtered\_df['Life expectancy'].mean()

LE\_std = filtered\_df['Life expectancy'].std()

# Filter the data based on the condition (Life expectancy > mean + std)

above\_mean\_std = filtered\_df[filtered\_df['Life expectancy'] > (LE\_mean + LE\_std)]

# Filter the data based on the condition (Life expectancy < mean)

below\_mean = filtered\_df[filtered\_df['Life expectancy'] < LE\_mean]

# Order the data by life expectancy

above\_one\_std = above\_mean\_std.sort\_values(by='Life expectancy', ascending=False)

below\_mean = below\_mean.sort\_values(by='Life expectancy', ascending=False)

# Create a bar chart to visualize the data

plt.figure(figsize=(10, 6))

plt.barh(above\_mean\_std['Entity'], above\_mean\_std['Life expectancy'], color='green', label='Above Mean + 1 Std')

plt.barh(below\_mean['Entity'], below\_mean['Life expectancy'], color='Yellow', label='Below Mean')

plt.xlabel('Lfe expectancy')

plt.title('Life Expectancy Comparison (2015-2022)')

plt.legend()

plt.gca().invert\_yaxis() # Invert the y-axis for better visualization

plt.show()

**# Task 2**

years\_to\_filter = [2019] #Visable year in graph

df = pd.concat([pd.read\_csv("life-expectancy-at-birth-oecd.csv")])

df\_year = df[df['Year'].isin(years\_to\_filter)] #Life expectancy for chosen year

df2 = pd.concat([pd.read\_csv("national-gdp-penn-world-table.csv")])

df2\_year = df2[df2['Year'].isin(years\_to\_filter)] #GDP for chosen year

olC = set(df\_2021['Entity']) & set(df2\_2021['Entity']) #Finding which countries has both data types

dfc = df\_year[df\_year['Entity'].isin(olC)] #Life expectancy list for joint countries

df2c = df2\_year[df2\_year['Entity'].isin(olC)] #GDP list for joint countries

print(dfc.describe()) #Shows mean life expectancy

print(df2c.describe()) #Shows mean GDP

xValues = df2c['GDP (output, multiple price benchmarks)']

yValues = dfc['Period life expectancy at birth - Sex: all']

ax = plt.axes()

ax.set\_facecolor('#DCDCDC') #color of graph

plt.grid(True) #grid on

#scatterplot with joint variable countries combining life expectancy to gdp from the two data sets

for i, entity in enumerate(olC):

x = df2c[df2c['Entity'] == entity]['GDP (output, multiple price benchmarks)']

y = dfc[dfc['Entity'] == entity]['Period life expectancy at birth - Sex: all']

color = random.choice(list(mcolors.CSS4\_COLORS.keys())) #Random color choice from the CSS4 hex-list

plt.scatter(x, y, color=color, label=entity)

plt.title('Life expectancy vs GDP')

plt.xlabel('GDP [$]')

plt.ylabel('Life expectancy [Years of age]')

plt.xscale('log')

legend = plt.legend(loc='upper left', bbox\_to\_anchor=(1, 1), ncol = 3)

legend.get\_frame().set\_facecolor('#DCDCDC')

plt.show()

#Compute the Pearson Correlation coefficient for Question 3

PCC = np . corref ( xValues , yValues ) [0 , 1]

print ( “Pearson correlation coefficient : “ , PCC)

**#Task 4 graph with colors.**

import matplotlib.colors as mcolors

import random

years\_to\_filter = [2019] #Visable year in graph

df = pd.concat([pd.read\_csv("life-expectancy-at-birth-oecd.csv")])

df\_year = df[df['Year'].isin(years\_to\_filter)] #Life expectancy at birth for chosen year

df2 = pd.concat([pd.read\_csv("gdp-per-capita-penn-world-table.csv")])

df2\_year = df2[df2['Year'].isin(years\_to\_filter)] #GDP per capita for chosen year

olC = set(df\_2021['Entity']) & set(df2\_2021['Entity']) #Find all countries that has both Life expectancy data and gdp per capita data.

dfc = df\_2021[df\_2021['Entity'].isin(olC)] #List of the life expectancy for the common countries

df2c = df2\_2021[df2\_2021['Entity'].isin(olC)] #List of the GDP per capita for the common countries

xValues = df2c['GDP per capita (output, multiple price benchmarks)']

yValues = dfc['Period life expectancy at birth - Sex: all']

ax = plt.axes()

ax.set\_facecolor('#DCDCDC') #background color of graph

plt.grid(True) #grid on

#scatterplot with joint variable 'entities' = countries combining life expectancy to gdp per capita from the two data sets

for i, entity in enumerate(olC):

x = df2c[df2c['Entity'] == entity]['GDP per capita (output, multiple price benchmarks)']

y = dfc[dfc['Entity'] == entity]['Period life expectancy at birth - Sex: all']

color = random.choice(list(mcolors.CSS4\_COLORS.keys())) #pick a random color for the country in the graph from the CSS4 list

plt.scatter(x, y, color=color, label=entity)

plt.title('Life expectancy vs GDP per capita')

plt.xlabel('x')

plt.ylabel('y')

plt.xscale('log')

legend = plt.legend(loc='upper left', bbox\_to\_anchor=(1, 1), ncol = 3)

legend.get\_frame().set\_facecolor('#DCDCDC')

plt.show()

# Left over code

df = pd.concat([pd.read\_csv("/content/drive/MyDrive/DAT565/Data/gdp-per-capita-worldbank.csv")])

years\_to\_filter = [2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022]

df\_2018 = df[df['Year'].isin(years\_to\_filter)]

df\_2018.count()

print(df\_2018)

df = pd.read\_csv("/content/drive/MyDrive/DAT565/Data/life-expectancy-at-birth-oecd.csv")

Life\_Expectancy = (50,60.70)

Life\_Expectancy = df[df['Period life expectancy at birth - Sex: all'].isin(Life\_Expectancy)]

Life\_Expectancy.count()

print (Life\_Expectancy)