# -\*- coding: utf-8 -\*-

"""

Created on Mon Nov 6 20:18:27 2023

@author: Prudhvi vardhan

"""

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

# read data from path using pandas

df = pd.read\_csv("C:\\Users\\user\\Downloads\\suicides\_data.csv")

# gives first 5 rows and columns of data

df.head()

# gives last 5 rows and columns of data

df.tail()

# gives overview of the data

df.info()

# give statistical overview

df.describe()

# graph1 :Line chart

def generate\_suicide\_timeline(df):

"""

this function shows the total world suicides of all time ,by region

"""

# mapping each country and dividing in to regions

# country\_region = {'USA': 'Americas', 'France': 'Europe', ...}

country\_region = {

'USA': 'america',

'Canada': 'america',

'Brazil': 'america',

'France': 'europe',

'Germany': 'europe',

'Italy': 'europe',

'China': 'asia',

'Japan': 'asia',

'India': 'asia',

}

df['region'] = df['country'].map(country\_region)

# Now, we group the data by year and region

yearwise\_region = df.groupby(['year', 'region'])['suicides/100k pop'].sum().unstack().reset\_index()

# Here , we draw lines for different regions

plt.figure(figsize=(12, 8))

for region in yearwise\_region.columns[1:]:

plt.plot(yearwise\_region['year'], yearwise\_region[region], marker='\*', label=(region))

# adding labelling to data

plt.title(" Total world wide suicides per 100k Population by region ", fontsize=20)

plt.xlabel('Year', fontsize=16)

plt.ylabel('No. Suicides per 100k Pop', fontsize=16)

plt.legend(title='Region', loc='upper right')

plt.grid(True)

plt.xticks(rotation=60)

plt.tight\_layout()

plt.show()

# calling the function

generate\_suicide\_timeline(df)

# graph2 : bar chart

def chart\_suicide\_demographics(data, analysis\_year, palette=None):

"""

Displays a bar chart for suicide demographics based on age for a specified year within the dataset.

Parameters:

- data: DataFrame containing the suicide statistics.

- analysis\_year: The year for which the statistics are to be analyzed.

- palette: A list of colors for the bars. If None, default colors will be used.

This function filters the dataset for the given year, calculates the sum of suicides by age group,

sorts the age groups into a standard order, and then plots a bar chart.

"""

if palette is None:

palette = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b']

# Narrow down the dataset to the chosen year

data\_for\_year = data[data['year'] == analysis\_year]

# Collate the total number of suicides per age bracket

suicide\_tally\_by\_age = (

data\_for\_year.groupby('age', as\_index=False)['suicides\_no']

.sum()

.sort\_values(by='suicides\_no', ascending=False)

)

# Define the order of age categories assuming a certain set of age groups

ordered\_ages = ['5-14 years', '15-24 years', '25-34 years', '35-54 years', '55-74 years', '75+ years']

suicide\_tally\_by\_age['age'] = pd.Categorical(suicide\_tally\_by\_age['age'], categories=ordered\_ages, ordered=True)

# Charting

plt.figure(figsize=(12, 7))

bars = plt.bar(suicide\_tally\_by\_age['age'], suicide\_tally\_by\_age['suicides\_no'], color=palette)

# Customize the chart appearance

plt.title(f'Suicide Demographics in {analysis\_year}', fontsize=18)

plt.xlabel('Age Category', fontsize=16)

plt.ylabel('Count of Suicides', fontsize=16)

plt.xticks(rotation=45)

plt.yticks(fontsize=13)

# Add a legend if multiple colors are used

if len(set(palette)) > 1:

plt.legend(bars, ordered\_ages)

# Enhance layout for better presentation

plt.tight\_layout()

plt.grid(True, which='both', linestyle='--', linewidth=0.5)

plt.gca().set\_axisbelow(True)

# Show the resulting bar chart

plt.show()

# Running the function with the dataset for the year 2015

chart\_suicide\_demographics(df, 2015)

# graph 3: Pie Chart

def generate\_suicide\_statistics\_chart(dataframe, target\_year, chart\_colors=None):

"""

Generates a pie chart displaying the distribution of suicide numbers by gender for a specified year.

Parameters:

- dataframe: pandas DataFrame containing the suicide data.

- target\_year: Integer representing the year to filter the data by.

- chart\_colors: Optional list of colors for the chart. Default colors are used if not provided.

The function filters the data for the given year, calculates the sum of suicides by gender,

and then plots a pie chart with the aggregated data.

"""

# Ensure chart colors are provided, or set default colors

if not chart\_colors:

chart\_colors = ['blue', 'red']

# Filter the data for the specified year

yearly\_data = dataframe[dataframe['year'] == target\_year]

# Aggregate suicide numbers by gender

suicide\_stats\_by\_gender = yearly\_data.groupby('sex')['suicides\_no'].sum().reset\_index()

# Plot configuration

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

plt.pie(

suicide\_stats\_by\_gender['suicides\_no'],

labels=suicide\_stats\_by\_gender['sex'],

colors=chart\_colors,

autopct='%1.1f%%',

startangle=90

)

# Set the title of the plot

plt.title(f'Distribution of Suicides by Gender in {target\_year}', fontsize=16)

# Ensuring the pie chart is a circle

plt.axis('equal')

# Display the plot

plt.show()

# We'll test this function with the year 1987 and custom colors

generate\_suicide\_statistics\_chart(df , 1987, chart\_colors=['lightblue', 'lightcoral'])