Programming Project Report

Team Members: Pilar Guerrero, Emilia Granja,
Alexía Chacón and Tessa Correig
Industry: Business
Area of Study: Customer Targeting/Segmentation
Type of Data: Demographic Data

Table of Contents

Abstract

This project aims to develop a software tool that generates a range of visualizations derived from provided data. Our program inputs three datasets containing demographic, human capital, and world population data from multiple countries and regions to generate the chosen visual representations the user selects. Our objective is to assist businesses that are looking to expand their operations to new areas, and for them to be able to compare the available data of each country or region to properly select the best possible country for their customer targets and back up their decisions with a more thorough market study. The user is able to select different countries and make comparisons within the different data available, thus providing the user with insights focused on real life applications. The outcome will offer a structured comparative analysis, guiding and empowering businesses with insights, towards a more informed decision regarding their targeted expansion strategies into new markets.

Keywords

Data visualizations, Demographics, Human Capital, World Population, Comparative analysis

Introduction

In today's global markets, strategic decision-making for business development and expansion rely heavily on data-driven insights. Comprehending the role between demographics and the human capital environment, in our current world population, is crucial for businesses aiming to penetrate and expand in new markets successfully.

For this project, we decided to build our program focused on demographic, human capital, and world population data. These types of data play crucial roles in guiding businesses on potential growth strategies as they are able to provide more conceptual and factual evidence to make proper predictions and decisions. Demographic data is excellent for target audience identification as it provides valuable cognizance of the age, gender, education, as well as other traits that support the identification and targeting of the best audiences for certain businesses. Market segmentation is also important in regards to demographic data as it helps segment markets based on their criteria, thus allowing businesses to custom their products to more specific groups. Another aspect is consumer behavior analysis which demographics portray through the understanding of buyer preferences and tendencies, allowing companies to align accordingly. In addition, risk assessment and decision-making are other reasons why these types of data are vital, as they provide an understanding into the economic stability, workforce and cultural environments, as well as the potential challenges a business may face in a new market. This supports decision making by analyzing risks and making informed choices regarding expansion strategies. Lastly, the data visualized supports businesses to have a competitive advantage, as they are able to perform in-depth analysis and understandings of the target market better than their competitors to excel accordingly and have a more focused expansion into new markets. Therefore, demographic, human capital, and world population data serve marvelously to empower companies to make proper decisions, while minimizing their potential risks, and evaluating their strategies to target their audience effectively in new areas.

The purpose of our program is to allow businesses to compare the available data of each country and region, to analyze and select the area that is most suitable for their customer targeting, thus assisting them in their marketing expansion strategies. Businesses that are looking to expand their operations into new markets benefit most from our program, as it is purposely guided to provide these businesses with visual insights of two main data sections for each country and region.

Objectives

Objectives for the Program

The main objective of the python program is to create a graphical user interface (GUI) application that will allow users to explore data and insights as they choose. The program loads data from Excel and Csv files that contain datasets found online that include demographics, human capital and the world population data and will allow visualizations of age distribution, education enrollment rates, life expectancy, and population density. The program presents a series of graphic menus and plots based on user selections and aims to get insights into patterns and trends.
The program needs the following libraries; 'pandas', 'openpyxl', 'matplotlib', and 'graphics' and aims to offer an accessible tool for getting valuable insights from data.

Constraints and presumptions:

Constraints and limitations:
The dataset might have missing and erroneous data so we need to consider that and implement a data validation and cleaning process.
The dataset might be large so we need to consider that the program will still be scalable and efficient
The program is designed for a single user interaction at a time and this might be confusing for users and they might violate this indicator, so we might need to clarify that simultaneous multi user interaction is not supported.
A limitation is that we cannot show every country in the world so we had to limit it to the ones we considered the most relevant for the purpose of our program.
Another limitation for the graphs we will do with regions is the generalization of results, the insights that the program will provide will be based on generalized trends and so it does not capture individual countries
Finally another limitation is that the interpretation of our data does require understanding of statistical concepts so it requires some user expertise.

Presumptions:
There are some presumptions that we need to consider. Firstly we must presume that users will interact directly with the program and will understand the dynamics. Moving on we also need to assume users have suitable hardware and software and can run the program. Adding up, we must presume the information is representative to the population. Finally we must presume that data will be consistent and over time data will remain the same.

Methodology

Program Specifications

Description:
This program is like a digital tool that helps you see and understand information about different regions and countries, like their populations and human capital statistics. You use a dashboard to pick which regions and plots you want to look at, and the program shows you graphs and charts to make it easier to understand.

Inputs:

1. 3 Datasets: The program takes as input two excel sheets, one with information regarding region demographics and one with information on human capital per region. It also takes a third CSV dataset containing data on population size evolution of different countries from 1970- 2022

Excel Dataset:

Contains Different countries and regions as observations. We will use two sheets of these excel sheets. The one with insights on human capital and the one with insights on demographics. We will use these insights per region to show a general overview.
Source:
https://www.unfpa.org/modules/custom/unfpa_global_sowp_portal/data-file/SWOP-Data-2023.xlsx

Csv Dataset:

This dataset contains data on the evolution of population size from 1970-2022 for a range of countries. We will not use all countries as there are too many.
Source:
https://www.kaggle.com/datasets/iamsouravbanerjee/world-population-dataset/download?datasetVersionNumber=

1) *User Dashboard Input:* Ideally the user will interact with a dashboard created with the graphics library. The dashboard :
   - Choose the types of plots they want to visualize among the options given as buttons through the dashboard
   - Select regions from the excel dataset when applicable for a plot.
   - Select countries for the csv dataset on population evolution of countries when applicable for the plot
   - Allows users to go back to the menu or quit when the finish visualizing the plot.

Outputs:

1) *Visualizations:* The program generates a range of visualizations based on the user selections and the available options. The main visualization plots include: A pie chart based on the age distribution of a specific region, one enrollment rate to different education levels for a selected region, A bar chart displaying the population density across all regions, a stacked bar chart displaying the life expectancy of male and female for each region. And finally a line plot of the

evolution of a population in size for a selected country between the years of 1970-2022.

2) *User Interface*: As the user interacts with the interface and selects new options new windows will appear with new buttons depending on the options selected. For example if the user selects to plot a pie chart of the population for the population distribution of a region the interface will automatically display an interface with buttons to allow the user to select a region.
After displaying each plot

Relationships:

The input dataset serves as the source of information, while user inputs and selections determine how that data is processed and presented to the user. The user is allowed to choose the plot he wants to display as well as the region/ country if applicable and the program uses the matplotlib library to display the selected choices.

Pseudo Code

Create function LoadData()
Load demographics dataset
Load human capital dataset
Load CountryPopulation dataset
Return the three datasets

Create function startButton()
Takes the window as an input
Display the main title called InisghtGRAPHIX
Set the title size to 30
Set the title font to Bold courier style
Draw the title inside the window

Display another title called Application
Set the title size to 20
Set the title font to courier style
Draw the title inside the window

Create a rectangle that will serve as a button
Make the button gray
Draw the button inside the window

Write the word START inside the button
Set the text size to 30
Set the title font to courier style
Draw the text inside the window

Create a while loop waiting for a mouse click.
If a mouse click is detected:
Check if the click is within the boundaries of the "START" button.
If it is, undraw all GUI elements and exit the loop.

Create function named create_button

(This function will be used throughout all the program to create buttons in menus displayed)

Create a rectangle using the provided values p1 and p2 to serve as a button
Make the button gray
Draw the button inside the window

Calculate the midpoint of the button for proper placement of the label text.
Create a text label with the specified content label
Set the text font to courier style
Set the text size to 15
Draw the text label in the window.
Return both the created button and text label.

Create function called undrawbuttons that takes graphical objects as a variable

Create for loop that takes every element and:

Undraw each element.

(This function will be used throughout all the program to delete buttons in menus displayed to switch from window to window)

Create a function called DisplayMenu()

Create a title and call it SELECT AN OPTION
Set the title size to 30
Set the title font to Bold courier style
Draw the title inside the window

Create a button called Plots per Regions
Create another button called Country populations

Create indefinite loop

If a mouse click is detected:

Check if the click is within the boundaries of the "Plots per Regions" button.

If it is, undraw all GUI elements and return region

Otherwise, check if the click point is within the "Country Populations" button.

If true, undraw the buttons, labels, and title, then return "country".
If no button is clicked, continue waiting for a valid click.

This function allows the user to choose between exploring plots for regions, or populations of countries by displaying two buttons respectively and waiting for the user to click

Create function called RegionMenu()

Create a title and call it PLOT OPTIONS
Set the title size to 30
Set the title font to Bold courier style
Draw the title inside the window

Create a button called Age distribution /region
Create a button called Level of Education /region
Create a button called Life expectancy /region
Create a button called Population Density /region

Enter indefinite loop
Check for mouse clicks
If there's a mouse click check if the click point is within the boundaries of the "Age Distribution /region" button.
If true, undraw the buttons, labels, and title, then prompt the user to select a region and return (1, region).
Check if the click point is within the "Level of Education /region" button.
If true, undraw the buttons, labels, and title, then prompt the user to select a region and return (2, region).
Check if the click point is within the "Life Expectancy /region" button.
If true, undraw the buttons, labels, and title, then return (3, None).
Check if the click point is within the "Population Density /region" button.
If true, undraw the buttons, labels, and title, then return (4, None).

This function is to show a menu with region plot options as buttons. It will wait for the user to click a button of a plot option and return the plot choice.

Next we will define a function so the user can select a region.

We call the function select_region ():

We create a title text with the according points that says SELECT REGION and is the button that will allow users to select a region from which they want to see graphs.
We select the size
We select the font type
And we make the style equal to bold to make the text thicker and darker
Then we will create buttons for different regions and give them specific coordinates:
First we create a button with the "Arab States" region
We create the button with "Asia and the pacific" region
We create a button for "Eastern Europe & CentralAsia"
We create button for "Latin American & Caribbean"
We create button for "East Southern Africa"
Finally we create a button for "West and Central Africa"

Then we need an infinite loop to handle user input until a region is selected meaning the loop will continue until the user clicks on one of the buttons.
Inside the loop we get mouse click and then we add if statements to check if the click is inside any of the buttons by checking within the coordinates of each button.
It will check if the mouse click is in among the Arab States button and if it is it will undraw graphical elements and return the name.

It repeat the process with each of the region buttons:

- Checks if click is on the Asia and the pacific to return the name
- Checks if click is on Eastern Europe & CentralAsia to return the name
- Checks if click is on Latin American & Caribbean to return the name
- Checks if click is on East Southern Africa to return the name

This function will wait for the user to click a region and will return the region, this will be saved to then call the plot.

We will define a function to input a title by creating a graphical user interface.
We call the function inputTitle ():

First we create initial text elements. The first one is “Choose titles” and we give the text coordinates.
Next we give size to text.
We set the font type to “Courier”
We set the style to bold to make the text bold.
We draw the title.
Next for the second text element we would add a text that displays that if no title is provided the app will set default title and we give it coordinates.
We five it a size
We set the font type
We draw the text.

Next we will create a “Done” button by using the create_button function created before with specific coordinates.

Later on we will create and display an entry widget for title input. Here the text input field will be created with coordinates and width of 50 characters. This will allow the user to input a desired title.

Next we will wait for user input in a loop that will wait for mouse clicks with the win.getMouse() function.
It will check if the mouse click occurred within the coordinates of the Done button created before.
If the done button is clicked it will undraw the button, its label, the initial elements and the title input field, and will return the text entered by the user in the title input field so that the title the user chose is displayed.

This function will wait for the user to select a title and click done to submit the input. It will return the title inputted, this will be saved for the title of the graph.

Next we create a function population_piechart()

The function takes as input the data and the title
We will first assign the specified region (by users) to the selected region.
We then filter the dataframe to include only rows corresponding to the selected region.
Next we will select the relevant population columns; in this case we select the column related to the population distribution across different age groups because we want a pie chart demonstrating that.
Then we create a new dataframe that will contain only the selected population columns for the specified region.

Then we extract values from the first row of the dataframe and convert them to a list
We specify colors for each segment of pie chart
We specify labels for each segment corresponding to different age groups
Then we create a Pie Chart using Matplotlib, we set the colors and labels to the assigned colors and labels specified before, we put percentage display, counterclockwise rotation and without shadow.
Finally we will set the title; we use the title set by the user but if no title is provided we generate a default title based on the region.
We plot the title
We display the pie chart

This function aims to display a pie chart to visualize the distribution of population across different age groups for specific regions (specified by user).

We Create a function for the education_graph

The function takes as input the data and the title
We will first assign the specified region (by users) to the selected region.
We then filter the dataframe to include only rows corresponding to the selected region.
Next we will select the relevant population columns; in this case we select the column related to net enrollment rates for different levels of education because we want a graph demonstrating that.
Then we create a new dataframe that will contain only the selected population columns for the specified region.
Next rename columns for better readability by specifying the education levels
Next the data frame must be transposed to be easier to plott. (Making education levels in the x-axis and regions in the y-axis)
Use matplotlib to create a stacked bar chart with specified parameters.
Set the labels for the x and y axes.
Set title to title provided in inputTitle function and if no title is provided generate a default title based on the region selected
Next adjust x-axis label for better readability
Finally display the bar chart generated.

This function is designed to demonstrate net enrollment rates for different levels of education in a specified region by returning a stacked bar chart.

We create a function **total_population**

The function will take as input the data and the title:
We then assign to regions the data from summary indicators
We then assign to totalpop the data from Total population in millions

After that, we create a new figure with a given color
We create a bar plot with the regions and the totalpop data
We create the x variable label as 'Regions' and y variable label as 'Total Population'

We create an if loop where if the title is empty:
We set the title as 'Population by Regions Bar Chart' (which is our predetermined title) This allows us to set a default title if there is not one inputted by the user,

Then, we create bars using the plt.bar with the regions and the totalpop data

We then rotate the x variable ticks and adjust the subplot bottom for the plot to show in a clearer way
And we show the created plot

We create a function called CountryMenu(win):
With this we create a text object called "SELECT COUNTRY" with the appropriate coordinates and with a given size, and draw it

We create the buttons and the labels for the different countries: Argentina, Brazil, France, India, China, and Russia

We create a while loop that while it is true:
It will get the mouse click with the given coordinates. This pauses the program until the user clicks.

When we have a click we check if a country button is clicked:
For this we create an if loop where if the Argentina button is clicked:
It will undraw buttons, labels, and title
And return "Argentina"
If the Brazil button is clicked:
It will undraw buttons, labels, and title
And return "Brazil"
And so on with all the given country options (France, China, Russia, and India)
In this way the country Menu program will end by returning the country selected by the user through the interface.

We will create a function called country_population
The function will take df, country, and title as inputs.
First we will extract the data for the given country
After, we select the population columns for the different years
And it will plot a line graph with the values against the years, using the markers for the data points

For this we set the x variable label as 'Year' and y variable label as 'Population'

We create an if loop where if the title is empty:
It will set the title as 'Population Over Time - {country}' (our predetermined title)
This allows us to set a default title if the user doenst input any.

It will then define a custom formatter function to avoid scientific notation in plots
Set the x and y axis
Rotate the y variable ticks for the plot to show in a more clearer way

Finally, it will show the given plot

We create a function called End_or_Repeat(win):
Then create a text object called "SELECTED PLOT HAS BEEN PLOTTED" with the appropriate coordinates and with a chosen size, and draw it on the window

After, we create the buttons and the labels for the "Back to Menu" and the "Quit" options

We create a while loop where while true:
It will get the mouse click given coordinates

Then it will check if the 'Back to Menu' button is clicked:
And it will proceed to undraw the buttons, labels, and title
And finally return 1

Then check if the 'Quit' button is clicked:
And undraw the buttons, labels, and title
And finally return 2

We create our main() function:
Where it will create a window called "Regions Insights" with the chosen size and set the appropriate coordinates

It will set the background color
And it will load the necessary data tables such as the: demographics_table, humancapital_table, and the country_data

After this, it will display the start button and enter the main loop

We created a while loop that while true:
It will get the user's selection from the display menu

Next we created an if loop where if the selection is the region:
It will get the user's plot and the region choice from the region menu available

We created an if loop where if the plot choice is 1:
It will get the title input from the user
And it will generate a population pie chart for the selected region with the demographics_table and the provided title given before
If the plot choice is the second one:
It will get the title input from the user
And it will generate an education graph for the selected region with the humancapital_table and the provided title given before
If the plot choice is the third one:
It will get the title input from the user
And it will generate a life expectancy plot for all the regions with the demographics_table and the title given above
If the plot choice is the fourth one:
It will get a title input from the user
And it will therefore generate a total population bar chart for all the regions with the demographics_table and the provided title

Alternatively if the user selection from display meny is the country:
It will get the user's country choice from the country menu available
It will also get the title input from the user

And finally generate a country population plot for the selected country using the country_data and the title given before

Finally the program will then check the user's choice to continue or quit
We created an if loop where if the choice is to continue:
It will continue in the while loop of main
And if not: (else)
It will break the loop and exit the program

Finally outside of any function, The program will call the main function to execute the whole program

Functions

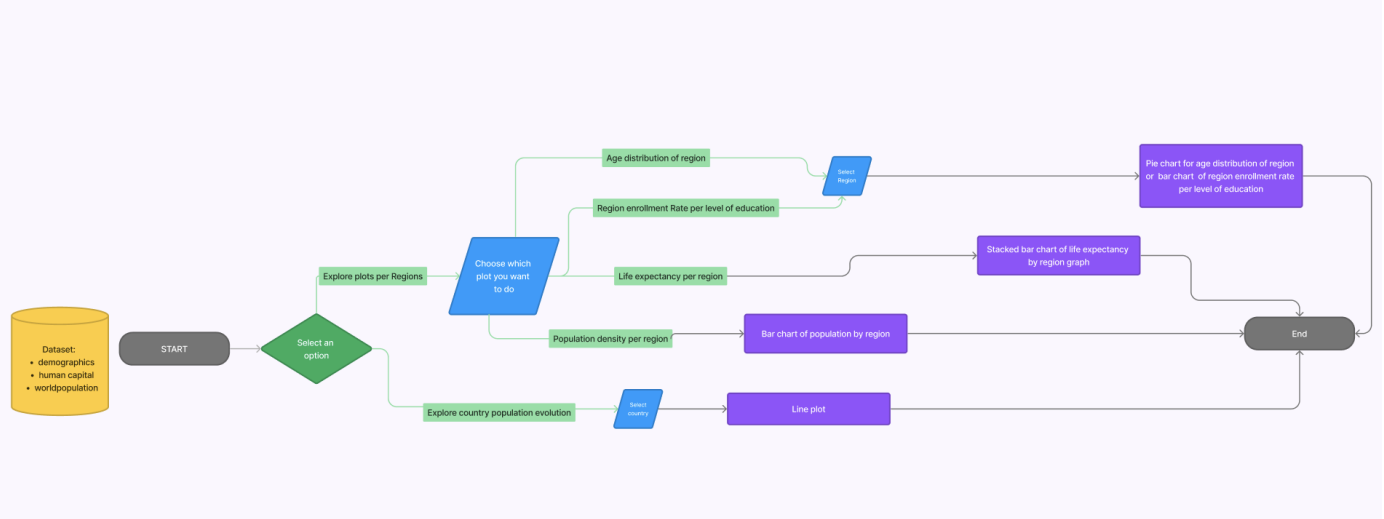
Function	Arguments	Pseudocode	Return
Main()	NA	NA	NA
Load Data/ Reading Function	Demographics Dataset Humancapital dataset	Reads both datasets and saves them both	Returns two dataframes in python containing the loaded data
Start	Window	Displays the start button to allow user to start using the app It waits until it is clicked to end.	
Create_Button	Window, Label, P1, P2	Automated function to create a Button on a specific window, with a given label in a given place defined by two points which are edges of a rectangle.	Button and Label
Undraw Buttons	Buttons and Labels to undraw	Undraws labels and buttons. This function is called after displaying every menu.	
DisplayMenu	window	Shows the main menu to the user. It uses the create button functions to create two buttons: 1) Explore Region Plots 2) Explore country Population Evolution The function will wait for a click to one of the buttons.	Return user selection of which plots to explore

RegionMenu	Window	Show the region menu to the user by creating buttons for each plot option. Creates the buttons using the create button function. Plot options include: Population Distribution Education Levels Life Expectancy Population Density Waits for the user to click a button of the available plot options. Once the user clicks it calls the adequate function to plot the plot selected. If necessary for the chosen plot it will allow the user to select a region by calling the select_region. Once the option is selected it calls the function Undraw_buttons to undraw the display menu.	Returns the plot option that the user selects, will also return the region if applicable for that plot.
Select_Region	Window	Show a menu with regions as buttons. Creates the buttons using the create button function. Waits for the user to click on the regions.	Returns region Selected
inputTitle	Takes a window	It takes a window that has the question of choosing a title and has a button with “Done” so the user clicks it when it's done choosing the title. Waits for the user to click on the button “Done”	Returns a title
population_piechart	Takes region, data, and a title	Plots pie chart of population distribution per age group for the region selected	Displays plot in pycharm
education_graph	Takes region and a title	Plots barchart of enrolment rate for different levels of education for the region selected	Displays plot in pycharm

Life_expectancy	Takes a title and data	Displays a plot of life expectancy stacked by gender for all Regions	Displays plot in pycharm
Total_population	Takes title and data	Displays a bar chart displaying the population density of each region	Displays plot in pycharm
CountryMenu	window	Displays a Country Menu by creating buttons for different countries. Waits for the user to click one of the buttons. Undraws all of the buttons and returns the selected country so we can plot the corresponding line chart by calling country_population	Returns the country for which the user wants to see population evolution.
country_population	Country selected, data and title	Plots a line plot of evolution of the population of the selected country from 1970- 2022	Displays Plot in pycharm
End_or_Repeat	Window	Creates two buttons, one to go back to the menu and one to quit the program. The function waits for the user to click one of the options.	Returns the selected option. (Back to Menu/ Quit)

Mockups and Design

Figma User-Interface Plan[1]

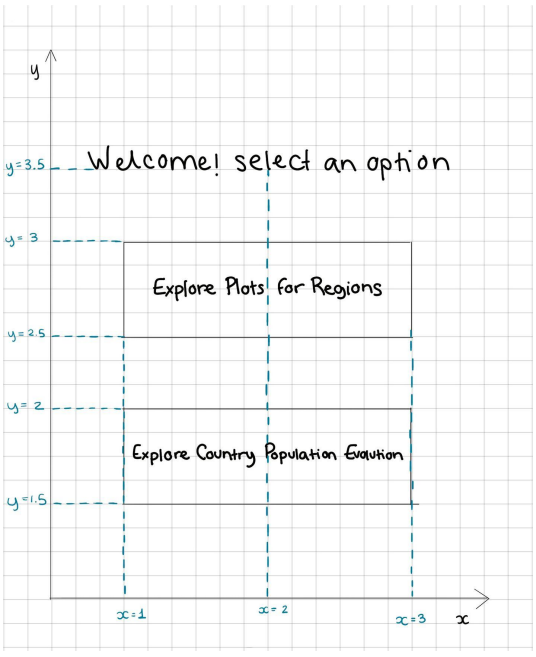


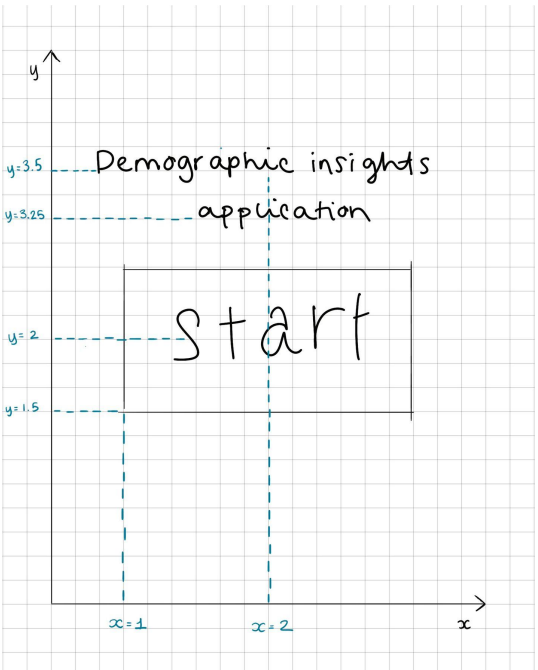
Link To Figma

Main Visualizations

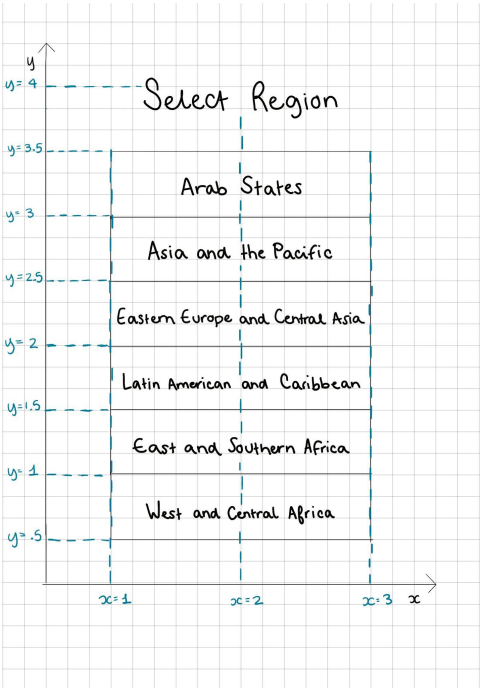
To present our data to the prospective businesses as clear as possible, we considered that some of the most useful and important visualizations would be the following:

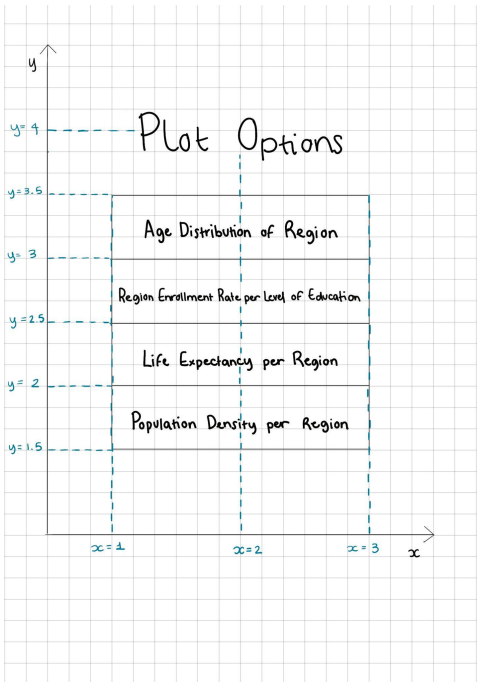
Plan of graphics interface:

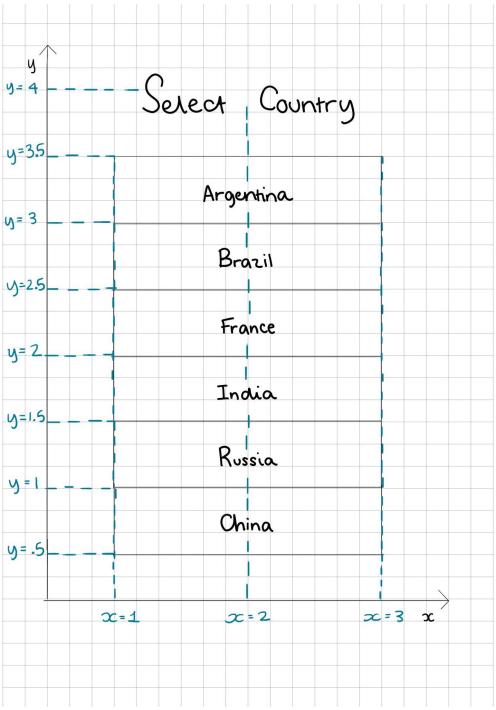


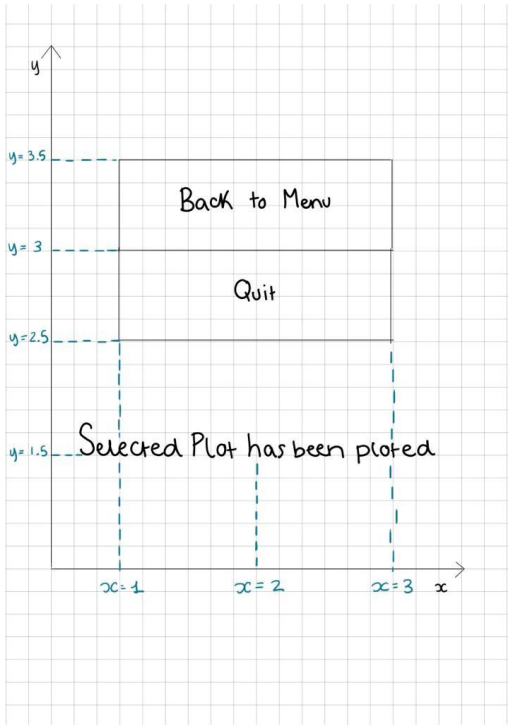


1 https://www.figma.com/file/M2LJpPpJdj3OYjvoJXC7w7/PROGRAMMING?type=whiteboard&node-id=976%3A499&t=PqYyrJEZhXMsHPvU-1


Select Region
Arab States
Asia and the Pacific
Eastern Europe and Central Asia
Latin American and Caribbean
East and Southern Africa
West and Central Africa
y
x
y = 4
y = 3.5
y = 3
y = 2.5
y = 2
y = 1.5
y = 1
y = .5
x = 1
x = 2
x = 3

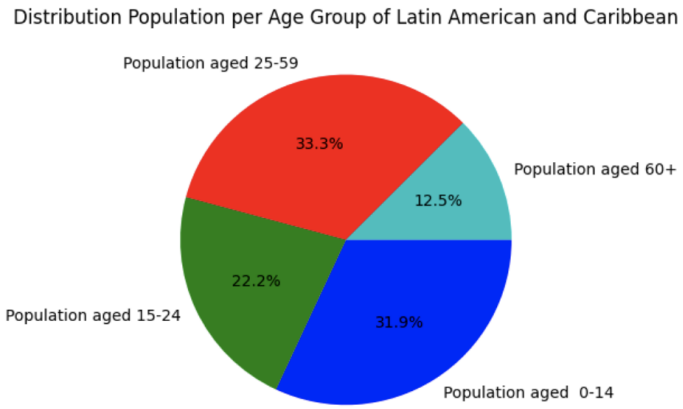

Plot Options
Age Distribution of Region
Region Enrollment Rate per Level of Education
Life Expectancy per Region
Population Density per Region
y
x
y = 4
y = 3.5
y = 3
y = 2.5
y = 2
y = 1.5
x = 1
x = 2
x = 3


Select Country
Argentina
Brazil
France
India
Russia
China
y
x
y = 4
y = 3.5
y = 3
y = 2.5
y = 2
y = 1.5
y = 1
y = .5
x = 1
x = 2
x = 3


Back to Menu
Quit
Selected Plot has been plotted
y
x
y = 3.5
y = 3
y = 2.5
y = 1.5
x = 1
x = 2
x = 3

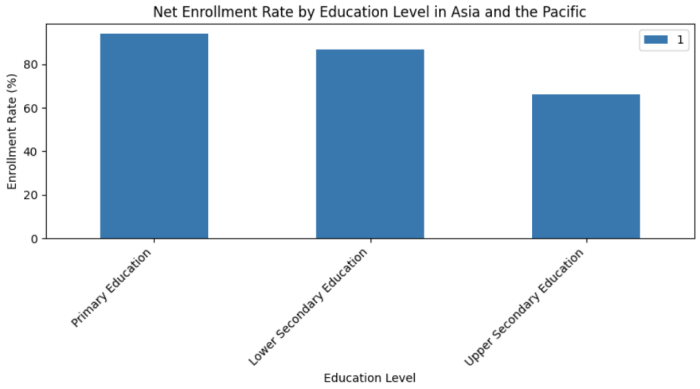
Visualization plots:

Pie chart:



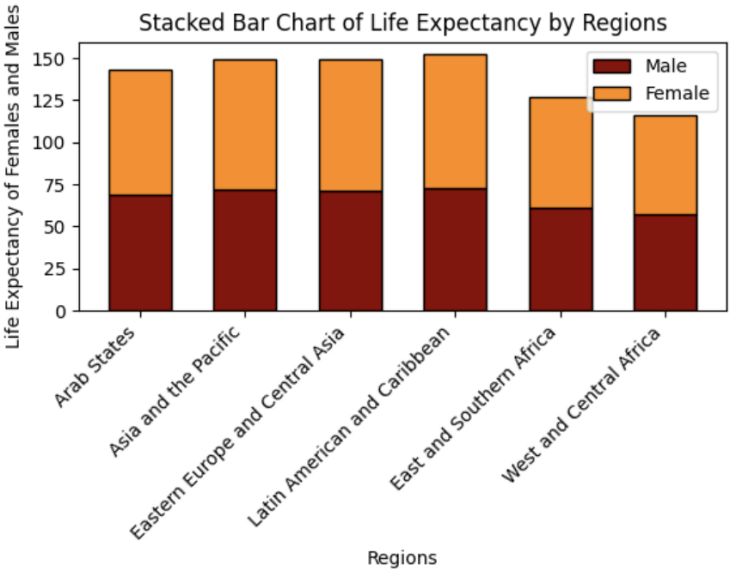
The population age distribution pie chart is a useful tool that helps the understanding of demographics. It has great visual clarity which allows users to identify population patterns, and for our target audience it is extremely helpful for their analysis as they will be able to see where the business is most likely to succeed . Besides its immediate use, it provides information on demographic trends such as aging or youth concentration.

Bar chart:



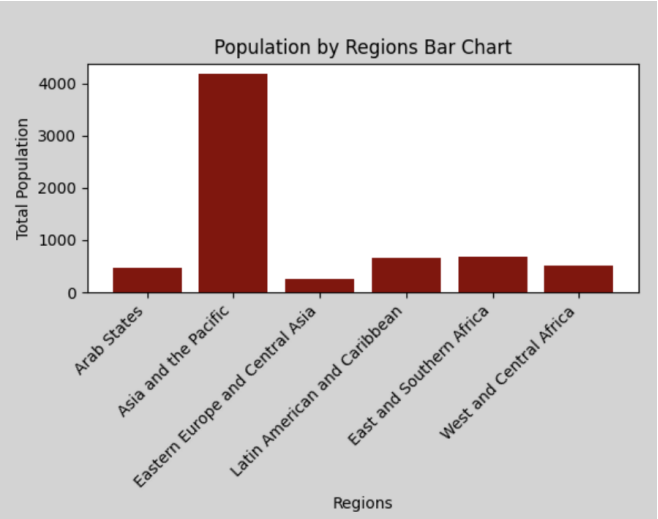
It displays the level of education in each country and allows for comparisons between the different levels of education. This is extremely helpful for companies as they are interested on getting educated labor for their businesses, so countries with a greater percentage of upper education are more attractive .

Stacked bar:



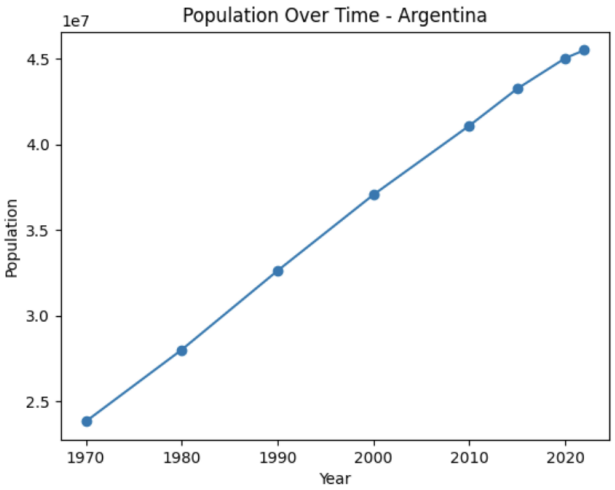
Display the dependency ratio, showing the proportion of people that are male or female while also portraying its life expectancy. Portraying this data is useful to assess the life-expectancy in each country and with relation to gender.

Bar chart:



It allows businesses to identify regions with larger or more suitable consumer and employee bases, contributing to the decision making in market expansion. It can also help for resource allocation.

Line plot:



Population growth over time can be visualized through line charts which show the growth of countries over time and could reveal fluctuations and increasing or decreasing growth rates.

As our demographic data analysis would be used for potential businesses, we considered that two topics to focus more in depth would be a toy company and fertility clinic. A toy company business would benefit from these demographics, as knowing the marriage and fertility rates of a country can show the potential of population growth, as well as dependency ratios that display the proportion of young people in a country that could become the target customers of the business. For our other topic, a fertility clinic could analyze marriage and fertility rates as well to evaluate in which countries would their clinics be of most use to people, and where there are more potential couples who would benefit from their business.

Conclusions and Recommendations

In conclusion, the Python code provided shows a simple graphical interface for exploring demographic information through pie charts and bar graphs. The program effectively visualizes age distribution, education rates, life expectancy and total population in different regions and countries. The design allows for easy expansion with additional features or enhancements in the future.

Furthermore, for improving the program and as recommendations we could include a greater variety of graphics and visualizations so that the user can get more insights on the information and can perform better analysis. It would also help to develop a detailed documentation to guide users into interpreting the visualizations given by the program and understanding the data presented, this will contribute to an accurate interpretation of information and will make the user experience easier and more friendly. Finally, we would also recommend improving the functionality to implement statistical analysis tools.

Bibliography

Source Datasets 1 & 2:
https://www.unfpa.org/modules/custom/unfpa_global_sowp_portal/data-file/SWOP-Data-2023.xlsx

Source dataset 3:
https://www.unfpa.org/data/world-population-dashboard

(We emailed you the actual datasets given that the datasets 1 & 2 where in different sheets and to access the dataset we separated them into two different excel files) - Check the email pilarguerreromorales@gmail.com

Appendix

Code

```
import pandas as pd
import openpyxl
import matplotlib.pyplot as plt
from graphics import *
from matplotlib.ticker import FuncFormatter

def LoadData():
    demographics_table = pd.read_excel('demographics.xlsx')
    humancapital_table = pd.read_excel('humancapital.xlsx')
    countrydata = pd.read_csv('world_population.csv')
    return demographics_table, humancapital_table, countrydata


def startButton(win):
    title = Text(Point(2, 3.5), "InsightGRAPHIX")
    title.setSize(30)
    title.setStyle("bold")
    title.setFace("courier")
    title.draw(win)

    title2 = Text(Point(2, 3.25), "Application")
    title2.setSize(20)
    title2.setFace("courier")
    title2.draw(win)

    start = Rectangle(Point(1,1.5), Point(3,2.5))
    start.setFill("lightgrey")
    start.draw(win)

    startText= Text(Point(2,2), "START")
    startText.setSize(30)
    startText.setStyle("bold")
    startText.setFace("courier")
    startText.draw(win)

    while True:
        pt = win.getMouse()
        if pt:
            if pt.getX() > 1 and pt.getX() < 3:
                if pt.getY() > 1 and pt.getY() < 3:
                    start.undraw()
                    startText.undraw()
```

```
                    title.undraw()
                    title2.undraw()
                    break

def create_button(win, label, p1, p2):
    button = Rectangle(p1, p2)
    button.setFill("lightgrey")
    button.draw(win)

    text_point = Point((p1.getX() + p2.getX()) / 2, (p1.getY() + p2.getY())
/ 2)
    text = Text(text_point, label)
    text.setFace("courier")
    text.setSize(15)
    text.draw(win)

    return button, text

def undrawbuttons(*elements):
    for element in elements:
        element.undraw()

def DisplayMenu(win):
    title = Text(Point(2, 3.5), "SELECT AN OPTION")
    title.setSize(30)
    title.setFace("courier")
    title.setStyle("bold")
    title.draw(win)

    button_region, label_region = create_button(win, "Plots for Regions ",
Point(1, 2.5), Point(3, 3))
    button_country, label_country = create_button(win, "Country
Populations", Point(1, 1.5), Point(3, 2))

    while True:
        click_point = win.getMouse()
        if button_region.getP1().getX() < click_point.getX() <
button_region.getP2().getX() and \
                button_region.getP1().getY() < click_point.getY() <
button_region.getP2().getY():
            undrawbuttons(button_region, label_region,button_country,
label_country,  title)
            return "region"

        elif button_country.getP1().getX() < click_point.getX() <
button_country.getP2().getX() and \
                button_country.getP1().getY() < click_point.getY() <
button_country.getP2().getY():
            undrawbuttons(button_region, label_region,button_country,
label_country,  title)
            return "country"



def RegionMenu(win):
    title = Text(Point(2, 4), "PLOT OPTIONS")
    title.setSize(30)
    title.setFace("courier")
    title.setStyle("bold")
    title.draw(win)
```

```
    button_population, label_population = create_button(win, "Age
Distribution /region", Point(1, 3), Point(3, 3.5))
    button_education, label_education = create_button(win, "Level of
Education /region", Point(1, 2.5), Point(3, 3))
    button_life_expectancy, label_life_expectancy = create_button(win, "Life
Expectancy /region", Point(1, 2), Point(3, 2.5))
    button_population_density, label_population_density = create_button(win,
"Population Density /region", Point(1, 1.5), Point(3, 2))

    while True:
        click_point = win.getMouse()


        if button_population.getP1().getX() < click_point.getX() <
button_population.getP2().getX() and \
                button_population.getP1().getY() < click_point.getY() <
button_population.getP2().getY():
            undrawbuttons(button_population, label_population,
button_education, label_education,
                          button_life_expectancy, label_life_expectancy,
button_population_density,
                          label_population_density, title)
            region = select_region(win)
            return(1, region)

        elif button_education.getP1().getX() < click_point.getX() <
button_education.getP2().getX() and \
                button_education.getP1().getY() < click_point.getY() <
button_education.getP2().getY():
            undrawbuttons(button_population, label_population,
button_education, label_education,
                          button_life_expectancy, label_life_expectancy,
button_population_density,
                          label_population_density, title)
            region = select_region(win)
            return(2, region)

        elif button_life_expectancy.getP1().getX() < click_point.getX() <
button_life_expectancy.getP2().getX() and \
                button_life_expectancy.getP1().getY() < click_point.getY() <
button_life_expectancy.getP2().getY():
            undrawbuttons(button_population, label_population,
button_education, label_education,
                          button_life_expectancy, label_life_expectancy,
button_population_density,
                          label_population_density, title)
            return(3, None)

        elif button_population_density.getP1().getX() < click_point.getX() <
button_population_density.getP2().getX() and \
                button_population_density.getP1().getY() <
click_point.getY() < button_population_density.getP2().getY():
            undrawbuttons(button_population, label_population,
button_education, label_education,
                          button_life_expectancy, label_life_expectancy,
button_population_density,
                          label_population_density, title)
            return(4, None)

def select_region(win):
    title = Text(Point(2, 4), "SELECT REGION")
```

```
   title.setSize(30)
   title.setFace("courier")
   title.setStyle("bold")
   title.draw(win)

   button_Arab, label_Arab = create_button(win, "Arab States", Point(1, 3),
Point(3, 3.5))
   button_Asia, label_Asia = create_button(win, "Asia and the Pacific",
Point(1, 2.5), Point(3, 3))
   button_Europe, label_Europe = create_button(win, "Eastern Europe &
CentralAsia", Point(1, 2), Point(3, 2.5))
   button_LatinAmerica, label_LatinAmerica = create_button(win, "Latin
American & Caribbean", Point(1, 1.5),
                                                           Point(3, 2))
   button_EastAfrica, label_EastAfrica = create_button(win, "East &
Southern Africa", Point(1, 1), Point(3, 1.5))
   button_WestAfrica, label_WestAfrica = create_button(win, "West & Central
Africa", Point(1, 0.5), Point(3, 1))


   while True:
       click_point = win.getMouse()

       if button_Arab.getP1().getX() < click_point.getX() <
button_Arab.getP2().getX() and \
               button_Arab.getP1().getY() < click_point.getY() <
button_Arab.getP2().getY():
           undrawbuttons(button_Arab, label_Arab,button_Asia,
label_Asia,button_Europe, label_Europe, button_LatinAmerica,
label_LatinAmerica,
                         button_EastAfrica,
label_EastAfrica,button_WestAfrica, label_WestAfrica , title)
           return ("Arab States")

       elif button_Asia.getP1().getX() < click_point.getX() <
button_Asia.getP2().getX() and \
                   button_Asia.getP1().getY() < click_point.getY() <
button_Asia.getP2().getY():
           undrawbuttons(button_Arab, label_Arab, button_Asia, label_Asia,
button_Europe, label_Europe,
                             button_LatinAmerica, label_LatinAmerica,
                             button_EastAfrica, label_EastAfrica,
button_WestAfrica, label_WestAfrica, title)
           return ("Asia and the Pacific")

       elif button_Europe.getP1().getX() < click_point.getX() <
button_Europe.getP2().getX() and \
                   button_Europe.getP1().getY() < click_point.getY() <
button_Europe.getP2().getY():
           undrawbuttons(button_Arab, label_Arab, button_Asia, label_Asia,
button_Europe, label_Europe,
                             button_LatinAmerica, label_LatinAmerica,
                             button_EastAfrica, label_EastAfrica,
button_WestAfrica, label_WestAfrica, title)
           return ("Eastern Europe and Central Asia")

       elif button_LatinAmerica.getP1().getX() < click_point.getX() <
button_LatinAmerica.getP2().getX() and \
                   button_LatinAmerica.getP1().getY() < click_point.getY()
< button_LatinAmerica.getP2().getY():
```

```
            undrawbuttons(button_Arab, label_Arab, button_Asia, label_Asia,
button_Europe, label_Europe,
                                button_LatinAmerica, label_LatinAmerica,
                                button_EastAfrica, label_EastAfrica,
button_WestAfrica, label_WestAfrica, title)
            return ("Latin American and Caribbean")

        elif button_EastAfrica.getP1().getX() < click_point.getX() <
button_EastAfrica.getP2().getX() and \
                    button_EastAfrica.getP1().getY() < click_point.getY() <
button_EastAfrica.getP2().getY():
            undrawbuttons(button_Arab, label_Arab, button_Asia, label_Asia,
button_Europe, label_Europe,
                                button_LatinAmerica, label_LatinAmerica,
                                button_EastAfrica, label_EastAfrica,
button_WestAfrica, label_WestAfrica, title)
            return ("East and Southern Africa")

        elif button_WestAfrica.getP1().getX() < click_point.getX() <
button_WestAfrica.getP2().getX() and \
                    button_WestAfrica.getP1().getY() < click_point.getY() <
button_WestAfrica.getP2().getY():
            undrawbuttons(button_Arab, label_Arab, button_Asia, label_Asia,
button_Europe, label_Europe,
                                button_LatinAmerica, label_LatinAmerica,
                                button_EastAfrica, label_EastAfrica,
button_WestAfrica, label_WestAfrica, title)
            return ("West and Central Africa")


def inputTitle(win):
    text = Text(Point(2, 3.5), "CHOOSE A TITLE")
    text.setSize(25)
    text.setFace("courier")
    text.setStyle("bold")
    text.draw(win)

    text2 = Text(Point(2, 1.5), "If no title provided App will set Default
title.")
    text2.setSize(14)
    text2.setFace("courier")
    text2.draw(win)
    button_Done, label_Done = create_button(win, "Done", Point(1, 2.5),
Point(3, 3))

    title = Entry(Point(2, 2), 50)
    title.draw(win)

    while True:
        click_point = win.getMouse()
        if button_Done.getP1().getX() < click_point.getX() <
button_Done.getP2().getX():
            undrawbuttons(button_Done, label_Done, text, text2, title)
            return title.getText()

def population_piechart(df,region, title ):
    selected_region = region
    df_region = df[df['SUMMARY INDICATORS'] == selected_region]

    population_columns = ['Population aged 0-14, percent', 'Population aged
15-24, percent',
```

```
                        'Population aged 25-59, percent', 'Population aged
60+, percent']
    df_population = df_region[population_columns]

    values =  df_population.iloc[0].tolist()
    colors = ['b', 'g', 'm', 'c']
    labels = ['Population aged  0-14', 'Population aged 15-24', 'Population
aged 25-59', 'Population aged 60+']

    plt.pie(values, colors=colors, labels=labels, autopct='%1.1f%%',
            counterclock=False, shadow=False)

    if title == "":
        title = 'Distribution Population per Age Group of ' + region
    plt.title(title)

    plt.show()


def education_graph(df, region, title ):
    selected_region = region
    df_region = df[df['SUMMARY INDICATORS'] == selected_region]

    education_columns = ['Total net enrolment rate, primary education,
percent',
                         'Total net enrolment rate, lower secondary
education, percent',
                         'Toal net enrolment rate, upper secondary
education, percent']
    df_education = df_region[education_columns]
    df_education.columns = ['Primary Education', 'Lower Secondary
Education', 'Upper Secondary Education']

    df_education_transposed = df_education.transpose()


    ax = df_education_transposed.plot(kind='bar', stacked=True, figsize=(10,
6))

    ax.set_xlabel('Education Level')
    ax.set_ylabel('Enrollment Rate (%)')

    if title == "" :
        title = f'Net Enrollment Rate by Education Level in {region}'

    plt.title(title)
    ax.set_title(title)

    plt.xticks(rotation=45, ha='right')
    plt.subplots_adjust(bottom=0.45)

    # Show the plot
    plt.show()

def life_expectancy(data, title = ""):
    barWidth = 0.6

    plt.bar(data['SUMMARY INDICATORS'], data['life expectancy at birth rate
male'], color='darkred', edgecolor='black',
            width=barWidth, label='Male')
```

```
    plt.bar(data['SUMMARY INDICATORS'], data['life expectancy at birth rate
female'],
            bottom=data['life expectancy at birth rate male'],
color='darkorange', edgecolor='black', width=barWidth,
            label='Female')

    plt.xlabel('Regions')
    plt.ylabel('Life Expectancy of Females and Males')

    if title  =="":
        title = f'Stacked chart bar of life expectancy by regions'

    plt.title(title)
    plt.legend()

    plt.xticks(rotation=45, ha='right')
    plt.subplots_adjust(bottom=0.45)

    plt.show()


def total_population(data, title ):

    regions = data['SUMMARY INDICATORS']
    totalpop = data['Total population in millions']

    plt.figure(facecolor='lightgrey')

    plt.bar(regions, totalpop)

    plt.xlabel('Regions')
    plt.ylabel('Total Population')

    if title == "":
        title = 'Population by Regions Bar Chart'
    plt.title(title)

    bars = plt.bar(regions, totalpop, color=['darkred'])

    plt.xticks(rotation=45, ha='right')
    plt.subplots_adjust(bottom=0.45)

    plt.show()

def CountryMenu(win):
    title = Text(Point(2, 4), "SELECT COUNTRY")
    title.setSize(30)
    title.setStyle("bold")
    title.setFace("courier")
    title.draw(win)


    button_Argentina, label_Argentina = create_button(win, "Argentina",
Point(1, 3), Point(3, 3.5))
    button_Brazil, label_Brazil = create_button(win, "Brazil", Point(1,
2.5), Point(3, 3))
    button_France, label_France = create_button(win, "France", Point(1, 2),
Point(3, 2.5))
    button_India, label_India = create_button(win, "India", Point(1, 1.5),
Point(3, 2))
```

```
   button_Russia, label_Russia = create_button(win, "Russia", Point(1, 1),
Point(3, 1.5))
   button_China, label_China = create_button(win, "China", Point(1, 0.5),
Point(3, 1))

   while True:
       click_point = win.getMouse()

       if button_Argentina.getP1().getX() < click_point.getX() <
button_Argentina.getP2().getX() and \
               button_Argentina.getP1().getY() < click_point.getY() <
button_Argentina.getP2().getY():
           undrawbuttons(button_Argentina, label_Argentina, button_Brazil,
label_Brazil, button_France, label_France,
                         button_India, label_India, button_Russia,
label_Russia, button_China, label_China, title)
           return "Argentina"

       elif button_Brazil.getP1().getX() < click_point.getX() <
button_Brazil.getP2().getX() and \
               button_Brazil.getP1().getY() < click_point.getY() <
button_Brazil.getP2().getY():
           undrawbuttons(button_Argentina, label_Argentina, button_Brazil,
label_Brazil, button_France, label_France,
                         button_India, label_India, button_Russia,
label_Russia, button_China, label_China, title)
           return "Brazil"

       elif button_France.getP1().getX() < click_point.getX() <
button_France.getP2().getX() and \
               button_France.getP1().getY() < click_point.getY() <
button_France.getP2().getY():
           undrawbuttons(button_Argentina, label_Argentina, button_Brazil,
label_Brazil, button_France, label_France,
                         button_India, label_India, button_Russia,
label_Russia, button_China, label_China, title)
           return "France"

       elif button_India.getP1().getX() < click_point.getX() <
button_India.getP2().getX() and \
               button_India.getP1().getY() < click_point.getY() <
button_India.getP2().getY():
           undrawbuttons(button_Argentina, label_Argentina, button_Brazil,
label_Brazil, button_France, label_France,
                         button_India, label_India, button_Russia,
label_Russia, button_China, label_China, title)
           return "India"

       elif button_Russia.getP1().getX() < click_point.getX() <
button_Russia.getP2().getX() and \
               button_Russia.getP1().getY() < click_point.getY() <
button_Russia.getP2().getY():
           undrawbuttons(button_Argentina, label_Argentina, button_Brazil,
label_Brazil, button_France, label_France,
                         button_India, label_India, button_Russia,
label_Russia, button_China, label_China, title)
           return "Russia"

       elif button_China.getP1().getX() < click_point.getX() <
button_China.getP2().getX() and \
```

```
            button_China.getP1().getY() < click_point.getY() <
button_China.getP2().getY():
            undrawbuttons(button_Argentina, label_Argentina, button_Brazil,
label_Brazil, button_France, label_France,
                          button_India, label_India, button_Russia,
label_Russia, button_China, label_China, title)
            return "China"

def country_population(df, country, title):
    df_country = df[df['Country/Territory'] == country]

    population_columns = ['1970 Population','1980 Population','1990
Population',
                          '2000 Population','2010 Population','2015
Population','2020 Population',
                          '2022 Population',]
    df_country = df_country[population_columns]

    values = df_country.iloc[0].tolist()
    values2 = [1970,1980,1990,2000,2010,2015,2020,2022]

    plt.plot(values2, values, marker='o')

    plt.xlabel('Year')
    plt.ylabel('Population')

    if title == "":
        title = f'Population Over Time - {country}'
    plt.title(title)

    plt.title(title)

    # Define a custom formatting function to avoid scinetific notation in
plots
    def custom_formatter(value, _):
        return f"{value:.0f}"
    plt.gca().xaxis.set_major_formatter(FuncFormatter(custom_formatter))
    plt.gca().yaxis.set_major_formatter(FuncFormatter(custom_formatter))
    plt.yticks(rotation=45, ha='right')

    plt.show()

def End_or_Repeat(win):
    title = Text(Point(2, 1.5), "SELECTED PLOT HAS BEEN PLOTTED")
    title.setSize(20)
    title.setStyle("bold")
    title.setFace("courier")
    title.draw(win)
    button_start, label_start = create_button(win, "Back to Menu", Point(1,
3), Point(3, 3.5))
    button_end, label_end = create_button(win, "Quit", Point(1, 2.5),
Point(3, 3))

    while True:
        click_point = win.getMouse()
        if button_start.getP1().getX() < click_point.getX() <
button_start.getP2().getX() and \
                button_start.getP1().getY() < click_point.getY() <
button_start.getP2().getY():
            undrawbuttons(button_start, label_start, button_end, label_end,
title)
```

```
            return 1

        elif button_end.getP1().getX() < click_point.getX() <
button_end.getP2().getX() and \
                button_end.getP1().getY() < click_point.getY() <
button_end.getP2().getY():
            undrawbuttons(button_start, label_start, button_end, label_end,
title)
            return 2

def main():
    win = GraphWin("Regions Insights", 500, 500)
    win.setCoords(0.0, 0.0, 4.0, 4.5)
    win.setBackground('white')

    demographics_table, humancapital_table, country_data = LoadData()

    startButton(win)

    while True:
        selection = DisplayMenu(win)

        if selection == "region":
            plot, region = RegionMenu(win)

            if plot == 1:
                title = inputTitle(win)
                population_piechart(demographics_table, region, title)

            if plot == 2:
                title = inputTitle(win)
                education_graph(humancapital_table , region, title)

            if plot == 3:
                title= inputTitle(win)
                life_expectancy(demographics_table, title)

            if plot == 4:
                title = inputTitle(win)
                total_population(demographics_table, title)

        elif selection == "country":
            country = CountryMenu(win)
            title = inputTitle(win)
            country_population(country_data , country, title)



        value = End_or_Repeat(win)

        if value ==1:
            continue
        elif value == 2:
            break


main()
```