1. The goals for your project, including what APIs/websites you planned to work with and what data you planned to gather

Our team worked with the following APIs:

1. https://restcountries.com/#endpoints-code - RESTCountries API
    a. This API gave us country name, country ISO code, country population, and country language
2. https://api-ninjas.com/api/country - ApiNinja API
    a. This API gave us country's GDP, country refugees number
3. https://www.travel-advisory.info/data-api - Travel Advisory API
    a. This API gave us the travel advisory score for a given country

With these APIs, we wanted to gather country data, specifically GDP, population, risk advisory score, number of refugees, and income data. Using country names and ISO codes as the primary data key, we wanted to determine how factors impact a country's income and risk advisory levels.

2. The goals that were achieved, including what APIs/websites you worked with and what data you did gather

Using the previously stated APIs, we gathered GDP, country populations, refugee numbers, income level, and risk advisory level by country. Before the data analysis, our original goal was to discover how a country's wealth interacts with numerous factors such as population, language, safety, and societal makeup. Throughout data analysis, we achieved the following goals:

- Using the RESTCountries API and ApiNinja API to organize countries by population and income levels, we could draw a connection between a country's population and income level, discovering that the wealthiest country maintained average population levels relative to the global average. In contrast, the low-middle-income countries were above average, and the lowest-income countries were below.
- Using the Travel Advisory API and RESTCountries API to organize countries by income levels, we could draw a connection between a country's income level and average travel advisory risk score, discovering that the wealthiest (High Income) country maintained an average risk score lower than that of Upper-Middle income, Lower-Middle Income, and Low-Income countries.
- Using the Travel Advisory API and RESTCountries API to group countries by primary language spoken. We could then draw a connection between the average risk score for a language, discovering that countries that spoke Korean, Thai, and Macedonian had the lowest advisory risk score.
- Using the Travel Advisory API and ApiNinja API to explore the relationship between travel advisory risk scores and the average numbers of refugees. We noticed that countries with a higher travel advisory risk score were correlated with a higher number of

refugees. Regions with lower travel risk scores typically had a lower number of refugees present. We could then draw a conclusion that areas with elevated risk scores typically host larger populations of refugees.

- Using the Travel Advisory API and ApiNinja API to explore the relationship between travel advisory risk scores and the average GDP in these regions. Contrary to what might be expected, there isn't a clear correlation between higher risk scores and lower GDP or vice versa. The data points appear scattered without much of a distinct pattern. This lack of a discernible correlation suggests that risk levels, as measured by these risk scores, do not always affect the economic output shown by GDP.

3. The problems that you faced

We faced many challenges while working on our report.

- With the average population by income graph, we needed help understanding how to visually represent data so the viewer could clearly understand the goal of our analysis. We then realized we could use color coding to translate why these populations by income level are relevant to the global average population of the country. By overcoming this, we realized how helpful color coding, legends, and other graph details can be to illustrate the full story of an analysis on a graph.

- Establishing the database proved to be a highly challenging endeavor due to the complexity involved in navigating the intricacies of accessing the API and correctly importing the relevant data into the SQLite database. The process was marked by confusion, particularly in deciphering the appropriate methods to interact with the API effectively and ensuring the accurate integration of the retrieved data into the structured format of the SQLite database. The challenges primarily stemmed from the need to understand and coordinate the various steps involved in this technical process, emphasizing the importance of a comprehensive and well-organized approach to successfully set up and manage the database.

- During the beginning stages of the project, we needed to identify and integrate three distinct APIs. We had a difficult time finding our APIs, and had a few ideas prior to our current one. A lot of APIs we found had restricted access, either requiring payment or a passcode. We came up with a plan involving 3 different travel APIs, but eventually realized the plan and data involved was much too complex. This stage of the project consisted of a lot of research and redirection, but eventually led us to our current APIs. These limitations consumed a lot of time and we were unable to continue with our intended analysis until we found 3 free and compatible APIs.

4. The calculations from the data in the database (i.e. a screenshot)

Average_Country_Population_by_Income_Level.txt

```
Average Country Population By Income Level:
================================================================================

Avg population of High Income: 40.13965668 million
Avg population of Upper-Middle Income: 39.84223872 million
Avg population of Lower-Middle Income: 59.808219439999995 million
Avg population of Low Income: 16.04797664 million
```

```
Average Risk Score Per Income Level:
================================================================================

Avg risk score for High Income: 2.944
Avg risk score for Upper-Middle Income: 3.1439999999999997
Avg risk score for Lower-Middle Income: 3.2879999999999994
Avg risk score for Low Income: 3.7640000000000002
```

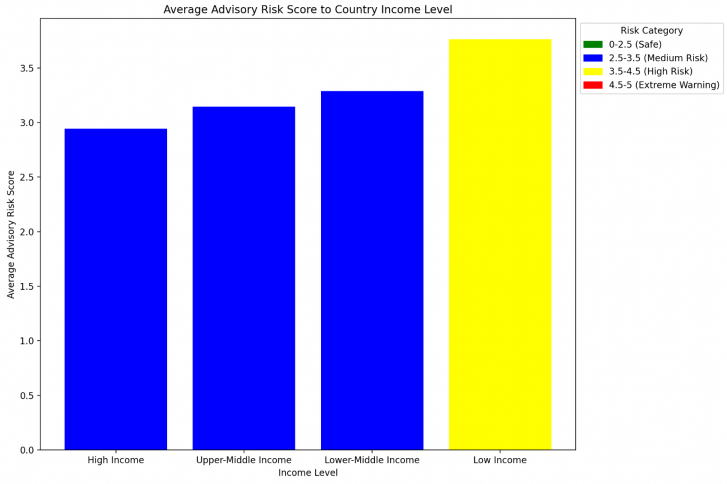
```
Average Risk Score Per Language:
================================================================================

Avg risk score for Afrikaans: 3.2
Avg risk score for Arabic: 3.5
Avg risk score for Armenian: 3.3
Avg risk score for Aymara: 3.3
Avg risk score for Azerbaijani: 3.25
Avg risk score for Bengali: 3.0999999999999996
Avg risk score for Bosnian: 3.5
Avg risk score for Bulgarian: 2.8
Avg risk score for Burmese: 4.25
Avg risk score for Czech: 3.1
Avg risk score for Danish: 3.6
Avg risk score for Dari: 4.0
Avg risk score for Dutch: 3.3
Avg risk score for English: 2.7
Avg risk score for Estonian: 2.6
Avg risk score for Finnish: 3.0
Avg risk score for French: 2.65
Avg risk score for German: 3.0
Avg risk score for Greek: 3.0
Avg risk score for Guaraní: 3.2
Avg risk score for Hungarian: 2.8
Avg risk score for Icelandic: 2.8
```

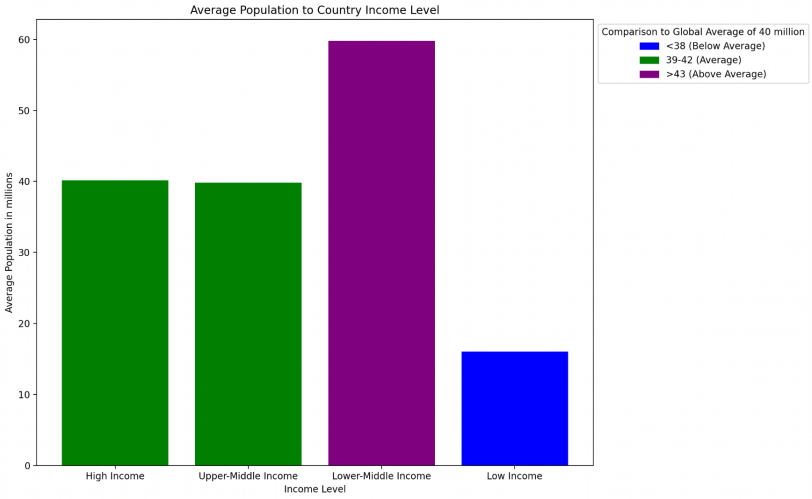
```
Risk Score, Average GDP
2.3, 653200.4
2.5, 382674.0
2.6, 39542.666666666664
2.7, 5332820.5
2.8, 830209.2727272727
3, 681075.95
3.1, 169252.66666666666
3.2, 249994.0909090909
3.3, 385507.1
3.4, 2307643.25
3.5, 231887.0
3.6, 1223401.0
3.7, 11387.0
3.8, 352083.5
3.9, 370588.0
4, 705141.0
4.1, 16200.0
4.2, 3285.0
4.4, 1660514.0
4.6, 72745.0
4.7, 130832.0
4.8, 11054.0
5, 11826.0
```

```
Risk Score, Average Number of Refugees
2.3, 36.15999999999999
2.5, 14.4
2.6, 102.53333333333335
2.7, 413.54999999999995
2.8, 68.58181818181819
3, 147.16999999999996
3.1, 252.30000000000004
3.2, 39.11818181818182
3.3, 1250.2800000000002
3.4, 276.0
3.5, 735.2833333333333
3.6, 302.1
3.7, 667.4
3.8, 1864.75
3.9, 55.1
4, 120.1
4.1, 196.3
4.2, 121.2
4.4, 135.5
4.6, 883.5
4.7, 1540.4
4.8, 207.1
5, 2469.2400000000002
```

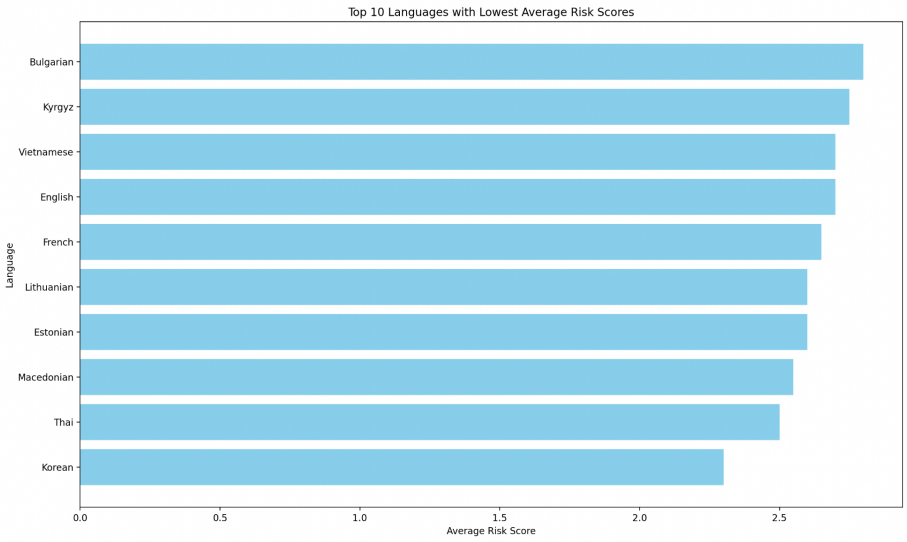
5. The visualization that you created (i.e. screenshot or image file)



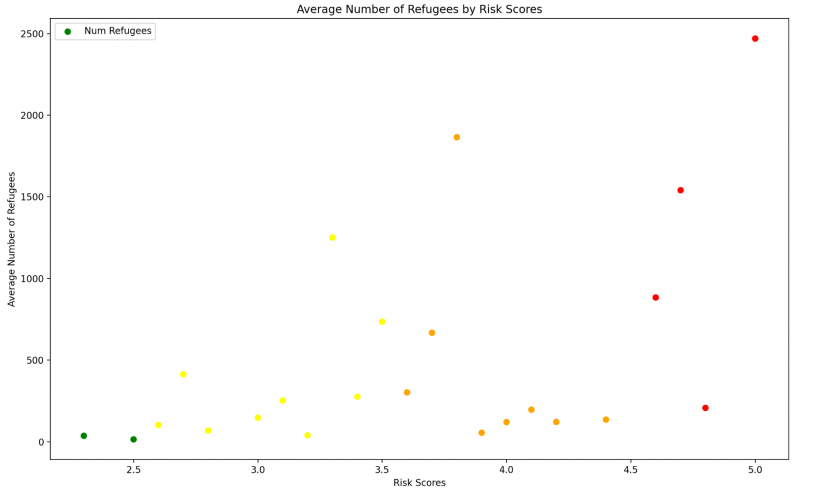
The bar graph above shows the relationship between average risk advisory scores and income level. For each income group (high, upper-middle, lower-middle, and low), we calculated the average risk score. The legend shows the different risk categories, ranging from safe to extreme warning. We found that even though low income countries aren't experiencing extreme risk warnings, high income doesn't mean the country is completely safe either. The graph shows a linear relationship between income and risk level, but all income levels fell into either the medium risk or high risk category.



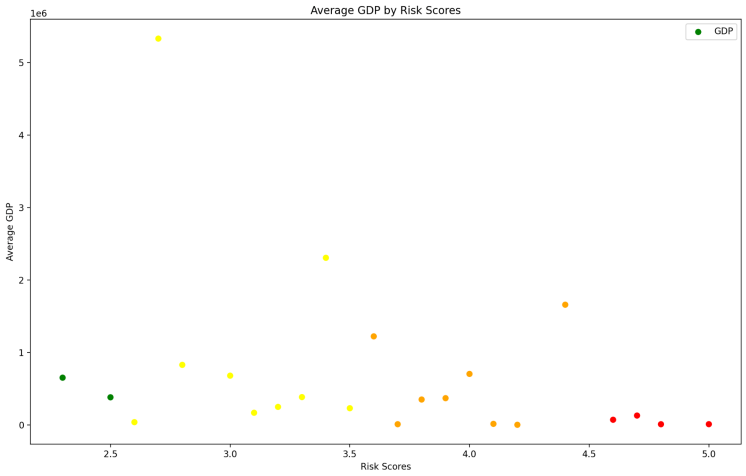
The graph above shows the average population of each income group relative to the global average population of the country. Higher, upper middle-income groups are closer to the global average population than lower income. Low-income countries have fewer people, lower/middle income have way above average populations, so therefore the most profitable population is at the average level.



The graph above shows the top 10 languages with the lowest average risk scores. We grouped countries together that spoke similar languages, and then calculated the average risk score for each group. We are able to see that countries that speak Korean have the lowest travel risk scores, and countries that speak Bulgarian have the highest travel risk score.



The graph above looks at the average number of refugees by risk scores. The colors of the data points represent different risk levels. Green is safe, yellow is medium risk, orange is high risk, and red is extreme warning. From the graph, we are able to see that countries with the highest risk scores (extreme warning) have the most amount of refugees. On the other hand, countries with the lowest risk scores (safe) have the least amount of refugees.



We also looked at the average GDP by risk scores. The data points represent the same risk levels as the graph above. However, it is not as easy to find a correlation between GDP and risk score. There doesn't seem to be a distinct pattern/relationship between high average GDP and low risk and low average GDP and high risk. However, we found that the highest average GDP tends to be in places with medium-high risk levels, compared to safe or extreme warning.

6. Instructions for running your code

1. Clear all of the previous data by deleting final_data
2. Press the run code button for the driver and wait for it to run entirely - until you receive a message saying “Should be done section.” This will give you the first 25 rows of data.
3. Repeat this step 3 more times, so that there are 100 total rows.
4. The database will populate in SQLite.
5. Once you have the datatable complete, you can begin to run the visualizations.
6. Start with averagePopByIncome and run the code.
7. A bar graph will pop up with the average population to country income level. Close out of this graph when finished.
8. Next, run the code for riskLevelIncomeClassAndLanguage.
9. Another graph will pop up with the top 10 languages with lowest average risk scores.
10. Once you close out of that graph, a second bar graph will show the average advisory risk score to country income level. Close out of this graph when finished.
11. To see the final visualizations, run the code for riskLevelRefugeesAndGDP.
12. A scatter plot will appear showing the average number of refugees by risk scores. Close out of this graph when finished.
13. A second graph will appear showing the average GDP by risk scores. Close out of this graph when finished.
14. To repeat the entire code, click on final_data and delete it. Run the driver.

7. Documentation for each function that you wrote. This includes describing the input and output for each function.

def set_up_database(db_name):
Input: db_name (string): The name of the database to be set up.
Output:
cur (sqlite3.cursor): Cursor object for the database.
conn (sqlite3.Connection): Connection object for the database.
Description:
This function sets up a SQLite database with the provided name (db_name). It returns a cursor (cur) and connection (conn) objects, allowing interaction with the database.

def create_tables(cur):
Input: cur (sqlite3.cursor): Cursor object for the database.
Output: None
Description:

Creates two tables in the database: LanguageTable and country_data. LanguageTable stores language information, and country_data stores country-specific data.

def populate_database(cur, conn):
Input:
cur (sqlite3.cursor): Cursor object for the database.
conn (sqlite3.Connection): Connection object for the database.
Output: None
Description:
Populates the LanguageTable and country_data tables with data obtained from external APIs, including country names, GDP, population, language, risk scores, and income levels.

def populate_income_level(cur, conn):
Input:
cur (sqlite3.cursor): Cursor object for the database.
conn (sqlite3.Connection): Connection object for the database.
Output: None
Description:
Populates the income_levels table with predefined income levels.
def sanity_check():
Input: None
Output: None
Description:
Performs a check to make sure that only 25 data points are getting printed out at a time, otherwise, prints a “ERROR!!” message.

def main():
Input: None
Output: None
Description:
The main function orchestrates the execution of other functions to set up the database, create tables, populate data, perform a sanity check, and visualize the results on SQLite.

def write_txt(data, filename):
Input: data (str): The text data to be written to the TXT file.
filename (str): The name of the TXT file to be created or overwritten.
Output: Creates a new TXT file named 'filename' and writes the provided 'data' as the text.
Description:
This function creates a new TXT file or overwrites an existing one with the specified 'filename', and writes the provided 'data' as the text content of the file.

def calc_avg_risk_score_per_income_level(cur):
Input:
cur (sqlite3.cursor): Cursor object for the database.
Output:
avg_scores (dict): A dictionary containing average risk scores for each income level.
Description:
Calculates the average risk score for each income level by querying the country_data table.

def calc_avg_risk_score_per_language(cur):
Input: cur (sqlite3.cursor): Cursor object for the database.
Output: avg_scores (dict): A dictionary containing average risk scores for each language.
Description:
Calculates the average risk score for each language by querying the country_data and LanguageTable tables. The data is GROUP BY language_name, then from that group AVG risk_score is calculated.

def get_color(avg_risk_score):
Input: avg_risk_score (float): Average risk score.
Output: color (string): A color code based on the risk score.
Description:
This function categorizes risk scores into color codes based on predefined ranges. It's designed to assign a specific color to each risk score range (green = 0-2.5, blue = 2.5-3.5, orange = 3.5-4.5, red = 4.5-5)

def bar_graph_risk_score(cur, conn):
Input:
cur (sqlite3.cursor): Cursor object for the database.
conn (sqlite3.Connection): Connection object for the database.
Output: None
Description:
Creates a bar graph displaying the average risk scores for each income level.

def plot_top_10_languages_with_lowest_risk_scores(cur):
Input:
cur (sqlite3.cursor): Cursor object for the database.
Output: None
Description:
Plots a bar graph showing the top 10 languages with the lowest risk scores based on the average risk score.

def calc_avg_population_per_income_level(cur):

Input:
cur (sqlite3.cursor): Cursor object for the database.
Output:
list (float): Average populations of each income level
Description:
This calculation goes through the data table, sums the population numbers of each income level, and counts the number of countries in each level. From there, the calculation divided the sum of the populations of each income level by the number of countries in each income level to return the average population of each income level.

def get_pop_color(avg_populations):
Input: List(float): The average population of each income level
Output: color(string): Color based on the population number
Description:
This calculation takes a number, finds the range the input fits into through if statements, and returns a string of the specified color.

def bar_graph_pop_by_income_lvl(cur, conn):
Input:
cur (sqlite3.cursor): Cursor object for the database.
conn (sqlite3.Connection): Connection object for the database.
Output:
None
Description:
Creates a bar graph of the average population by each income level

def get_color_for_risk_score(score):
Input:
score (float): risk score for a particular data point
Output:
color (string): A color code based on the risk score.
Description:
This function categorizes risk scores into color codes based on predefined ranges. It's designed to assign a specific color to each risk score range (green = 0-2.5, yellow = 2.5-3.5, orange = 3.5-4.5, red = 4.5-5)

def risk_level_avg_refugees(cur):
Input: cur (sqlite3.cursor): Cursor object for the database.
Output:
Scatter plot: displays a scatter plot using Matplotlib that shows the relationship between risk scores and the average number of refugees

Description:
Generates a scatter plot where the x-axis represents risk scores and the y-axis represents the average number of refugees. Each data point is color coded according to the risk levels.

def risk_level_avg_gdp(cur):
Input: cur (sqlite3.cursor): Cursor object for the database.
Output:
Scatter plot: displays a scatter plot using Matplotlib that shows the relationship between risk scores and the average GDP
Description:
Generates a scatter plot where the x-axis represents risk scores and the y-axis represents the average GDP. Each data point is color coded according to the risk levels.

def main():
Input: None
Output: None
Visualization: displays scatter plots and bar graphs based on the functions called within in
Description:
The main function orchestrates the execution of other functions to set up the database, create tables, populate data, perform a sanity check, and visualize the results.

8. You must also clearly document all resources you used. The documentation should be of the following form

Date	Issue Description	Location of Resource	Result(Did it solve the issue?)
12/01/23	Gather data on different country names and ISO codes	https://restcountries.com/#endpoints-code	Yes, to make our tables and to enter into travel advisory API
12/01/23	Gather data on average GDP and number of refugees for different countries	https://api-ninjas.com/api/country	Yes, used for our visualizations
12/01/23	Gather data on travel risk advisory levels	https://www.travel-advisory.info/data-api	Yes, used for our visualizations

12/11/23	Struggled to write results to a TXT file	https://www.w3schools.com/python/python_file_write.asp	Figured out how to open a new file and write text into it. We then implemented this into our calculation functions to write the results as TXT files
12/02/23	Struggled to implement APIs into code.	https://www.dataquest.io/blog/python-api-tutorial/	Referred to this website to understand API documentation
12/01/23 -12/11/23	Struggled to debug code for setting database & setting up color legend	https://chat.openai.com/	Yes, it helped simplify the problem, identify the errors, and gave direction on what to fix.
12/09/23	Struggled with barplot documentation	https://matplotlib.org/stable/plot_types/basic/bar.html#sphx-glr-plot-types-basic-bar-py	Yes, the documentation helped me to understand the syntax of setting up a bar plot
12/09/23	Struggled with scatter plot documentation	https://matplotlib.org/stable/plot_types/basic/scatter_plot.html#sphx-glr-plot-types-basic-scatter-plot-py	Yes, the documentation helped me to understand the syntax of setting up a scatter plot