Table of Contents

PROJECT REPORT: CO2 EMISSION REPORT GENERATOR

Prepared by: Aiman Qaid

Student ID: GH1023975

1. Introduction:

The CO2 Emission Report Generator is a software project developed to provide users with a comprehensive tool for analyzing and addressing environmental impact.

In the following steps you can see how this program works:

- Users must enter the number of emissions desired and run the program.
- Program will prompt a new input field to insert the \`type\` & \`amount\` of that emission.
- A file will be created with \`.pdf\` extension in the same location of source-code.

This report outlines the objectives, methodology, and outcomes of the project.

2. Objectives:

- ✓ Develop a user-friendly tool for analyzing CO2 emissions.
- ✓ Show visualizations to demonstrate the most CO2 emission cause.
- ✓ Provide insights and recommendations for reducing environmental impact.

3. Methodology:

- Research:
    - Conducted extensive research on CO2 emissions, environmental impact assessment methodologies, and software development frameworks.
- Design:
    - Designed the user interface, database schema, and report generation algorithms.
- Development:
    - Implemented the software using python and visual studio framework.
- Testing:

- Conducted rigorous testing to ensure functionality, usability, and reliability.

4. Features:

The CO2 Emission Report Generator presents a suite of sophisticated functionalities designed to enhance user experience and facilitate comprehensive environmental analysis:

- User-Friendly Interface: Intuitive design for easy navigation and usage.
- Various Reports: Allows users to modify reports according to their inputs.
- Data Visualization: Utilizes graphs, charts, and maps to present CO2 emission data effectively.

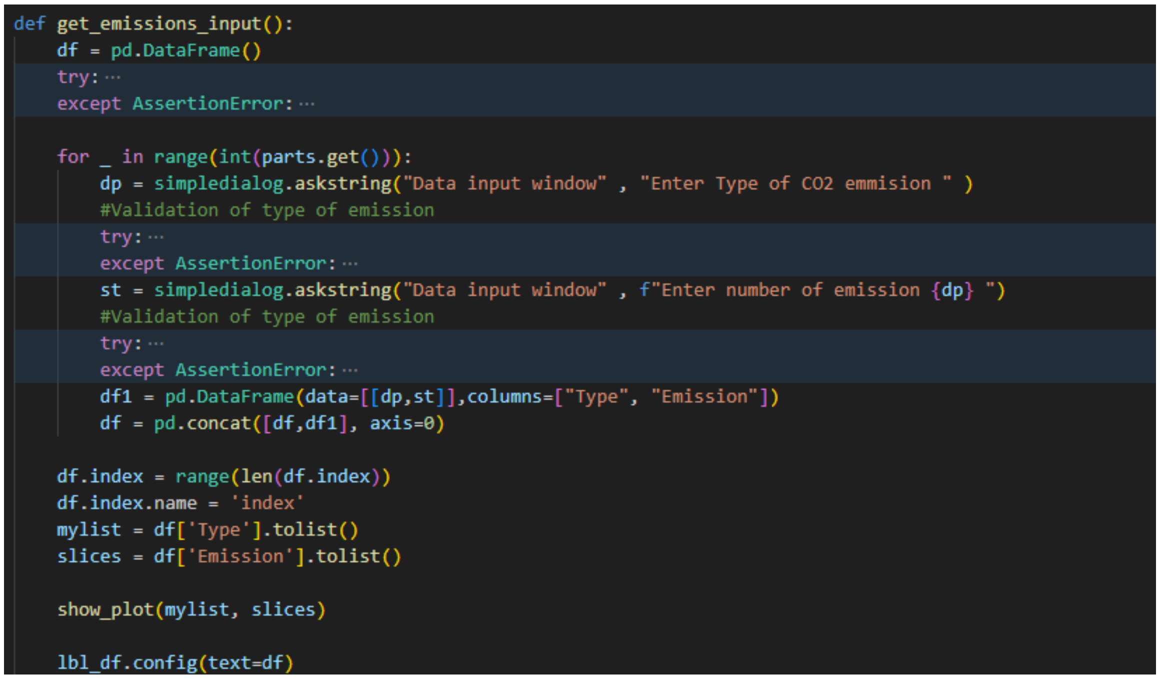
5. Outcomes:

Functional Software: Successfully developed a functional CO2 Emission Report Generator meeting all project requirements.

Impactful Insights: Provided users with valuable insights into their environmental impact and actionable recommendations for improvement.

6. Images:

The Emission Report Generator, is designed to streamline emissions input from various companies/clients using “tinkiter” library, visualize data in a pie chart using “Matplotlib” library, and generate PDF reports that includes suggestions on how we can reduce the CO2 emission,



```python
def get_emissions_input():
    df = pd.DataFrame()
    try:...
    except AssertionError:...

    for _ in range(int(parts.get())):
        dp = simpledialog.askstring("Data input window" , "Enter Type of CO2 emmision " )
        #Validation of type of emission
        try:...
        except AssertionError:...
        st = simpledialog.askstring("Data input window" , f"Enter number of emission {dp} ")
        #Validation of type of emission
        try:...
        except AssertionError:...
        df1 = pd.DataFrame(data=[[dp,st]],columns=["Type", "Emission"])
        df = pd.concat([df,df1], axis=0)

    df.index = range(len(df.index))
    df.index.name = 'index'
    mylist = df['Type'].tolist()
    slices = df['Emission'].tolist()

    show_plot(mylist, slices)

    lbl_df.config(text=df)
```

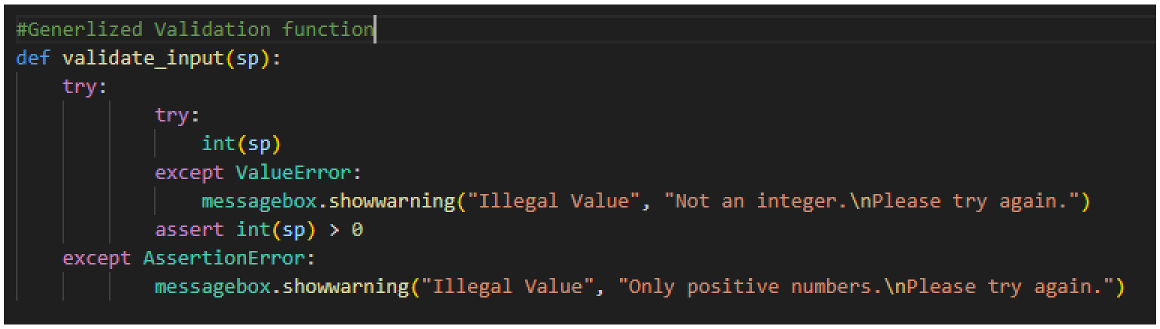
Figure 1 This function takes the input of CO2 emissions.

Using this function the users/clients can insert data easily with a GUI (Graphical user interface) using a library called "tkinter". This library prompts the user with an input field so they can easily insert the data they need; it also allows them to enter as many emissions as they need; using "pandas" data frame function users are able to insert a multi dimensions data frame (more like a schedule) to enter the data as shown below:

User selected 3 emission inputs	
Cars	40%
Trains	20%
Airplanes	70%

Table 1 Sample Data of CO2 Emissions

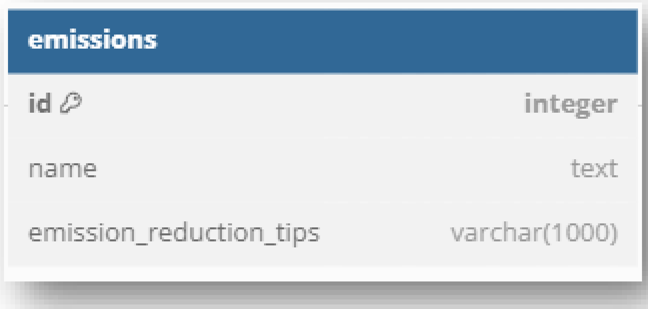
Before moving through, it better to protect your input fields, to avoid any vulnerabilities in our code, making a block of code and using it in other places is not a good practice so take a look at the "Generalized Validation Function" in the image below:



```
#Generlized Validation function
def validate_input(sp):
    try:
            try:
                int(sp)
            except ValueError:
                messagebox.showwarning("Illegal Value", "Not an integer.\nPlease try again.")
            assert int(sp) > 0
    except AssertionError:
            messagebox.showwarning("Illegal Value", "Only positive numbers.\nPlease try again.")
```

Figure 2 Generalized Validation Function

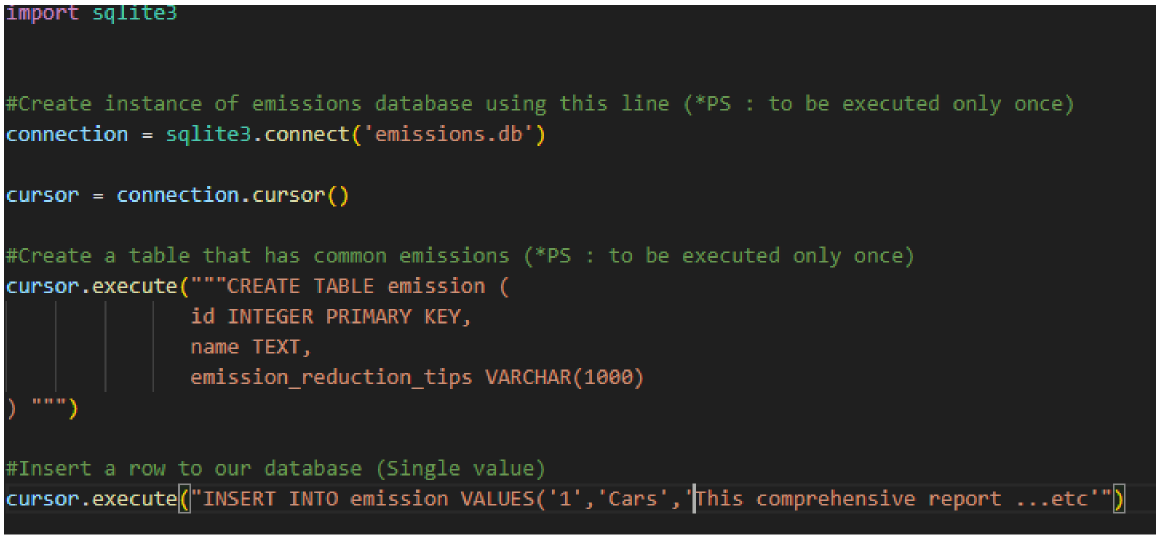
Next, we will go through our (SQLite) database and the scheme of this database. In Python SQLite database is built in so no need to install any library to use it. With only 3 variables we can create our database as show below:



emissions	
id	integer
name	text
emission_reduction_tips	varchar(1000)

Figure 3 SQLite Database Scheme

Now, we will go through the code of how we can implement these kinds of databases using the snip of code show below:

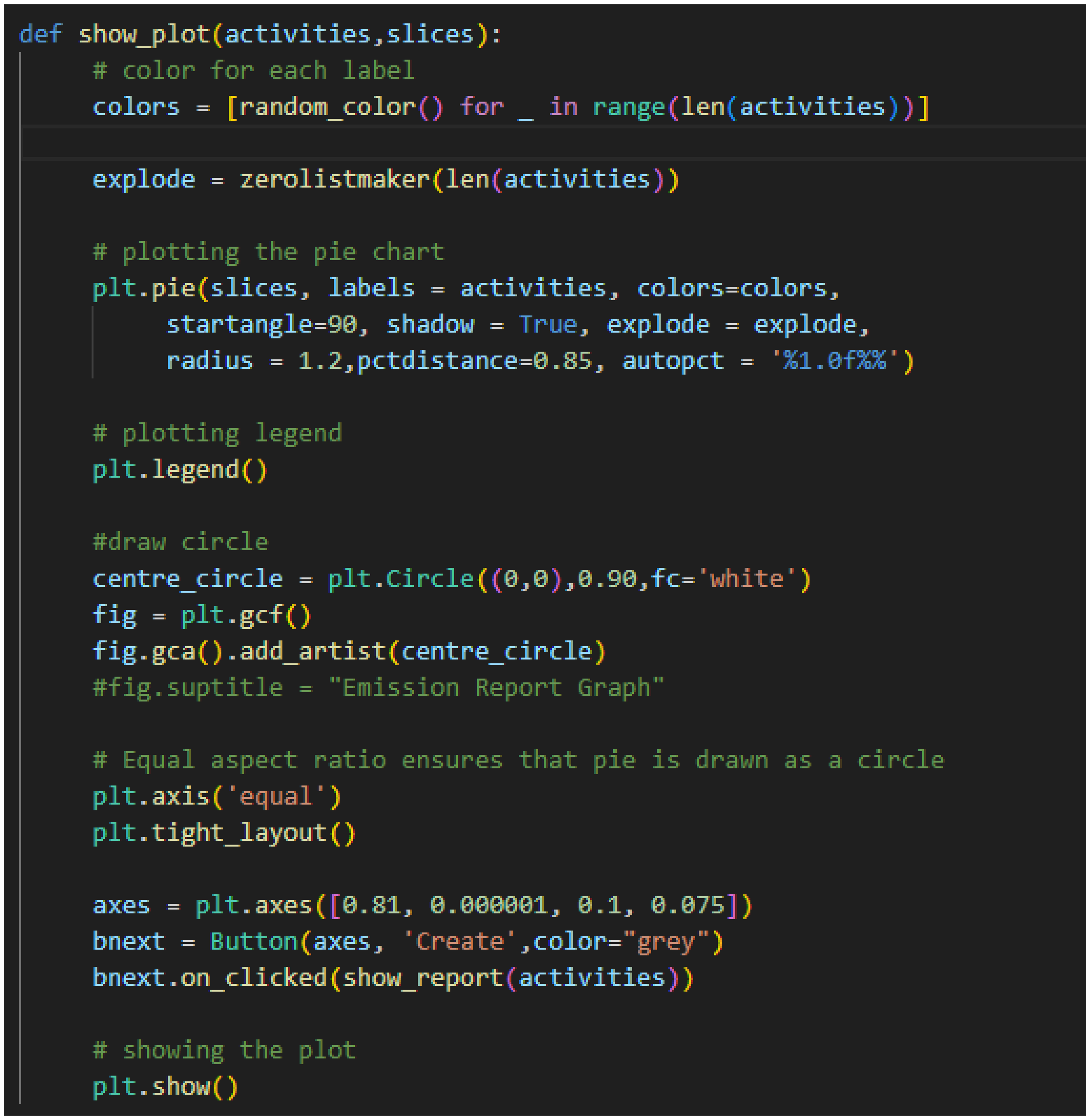


```
import sqlite3


#Create instance of emissions database using this line (*PS : to be executed only once)
connection = sqlite3.connect('emissions.db')

cursor = connection.cursor()

#Create a table that has common emissions (*PS : to be executed only once)
cursor.execute("""CREATE TABLE emission (
               id INTEGER PRIMARY KEY,
               name TEXT,
               emission_reduction_tips VARCHAR(1000)
) """)

#Insert a row to our database (Single value)
cursor.execute("INSERT INTO emission VALUES('1','Cars','This comprehensive report ...etc'")
```

Figure 4 Database Implementation Code

Reading the comments in the code snippet above demonstrates how the code works.

After that, we will go to the part where the data will be visualized in simple “Pie Chart.” Depending on what users inserted, check out the image below to have a better view of what will the code generate the chart:



```python
def show_plot(activities,slices):
    # color for each label
    colors = [random_color() for _ in range(len(activities))]

    explode = zerolistmaker(len(activities))

    # plotting the pie chart
    plt.pie(slices, labels = activities, colors=colors,
        startangle=90, shadow = True, explode = explode,
        radius = 1.2,pctdistance=0.85, autopct = '%1.0f%%')

    # plotting legend
    plt.legend()

    #draw circle
    centre_circle = plt.Circle((0,0),0.90,fc='white')
    fig = plt.gcf()
    fig.gca().add_artist(centre_circle)
    #fig.suptitle = "Emission Report Graph"

    # Equal aspect ratio ensures that pie is drawn as a circle
    plt.axis('equal')
    plt.tight_layout()

    axes = plt.axes([0.81, 0.000001, 0.1, 0.075])
    bnext = Button(axes, 'Create',color="grey")
    bnext.on_clicked(show_report(activities))

    # showing the plot
    plt.show()
```

Figure 5 Show Chart Pie Code

Again, reading the comments in the code snippet is more than enough to understand what each line of code does, in more details, we will take the user inputs emission “type” & “numbers” as “activities” & “slices” respectively, have a look at the pie chart below:

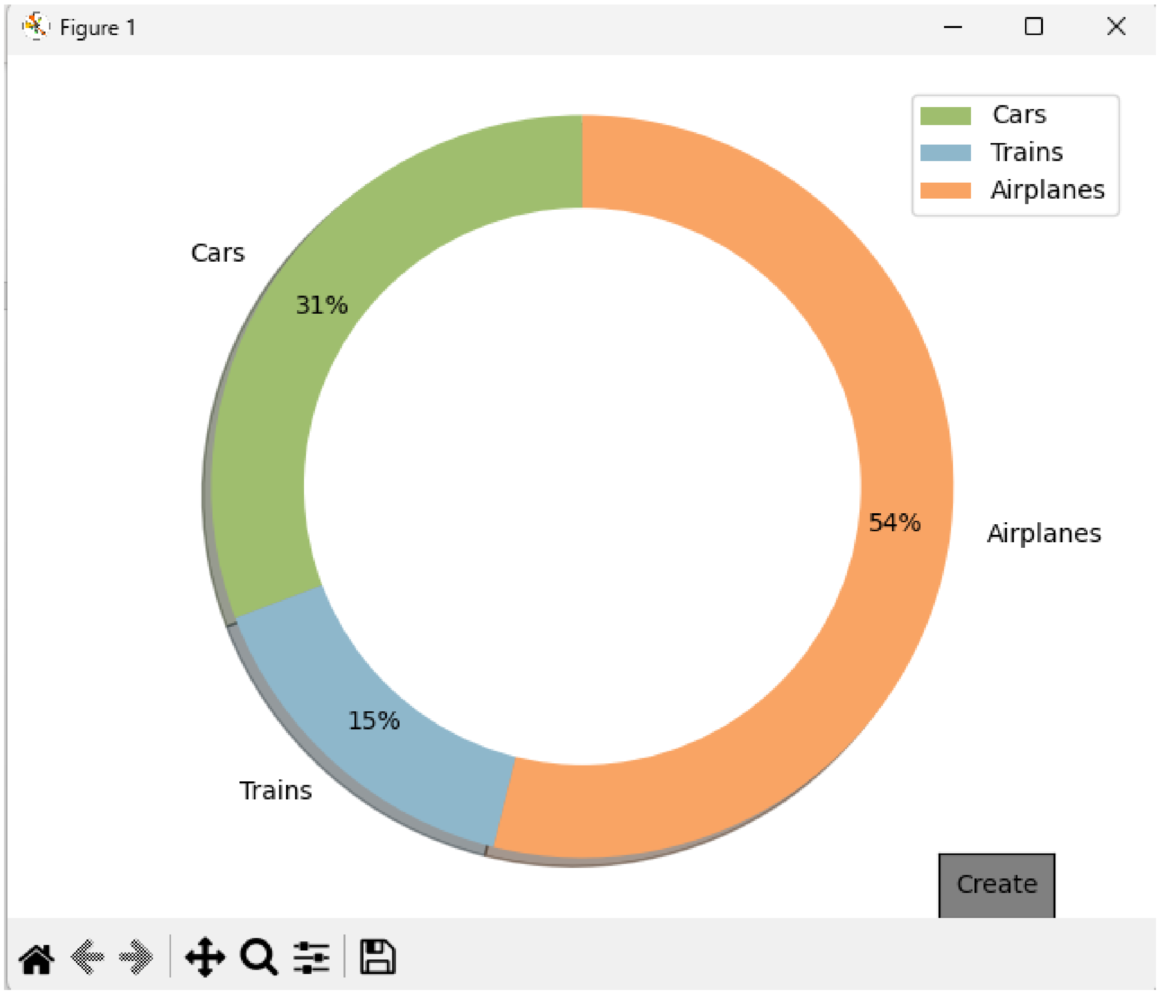
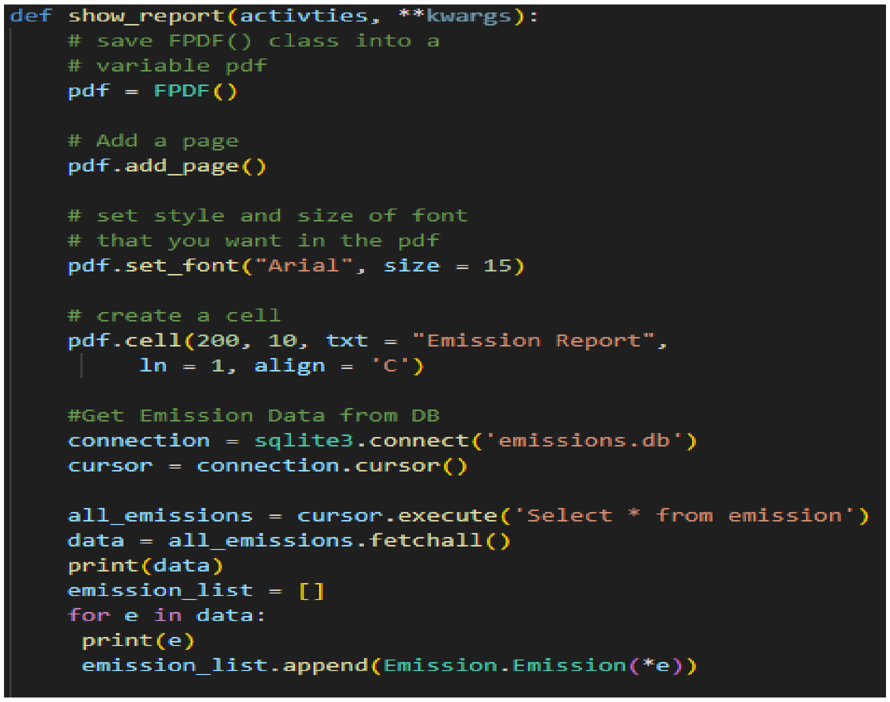


Figure 6 Pie Chart

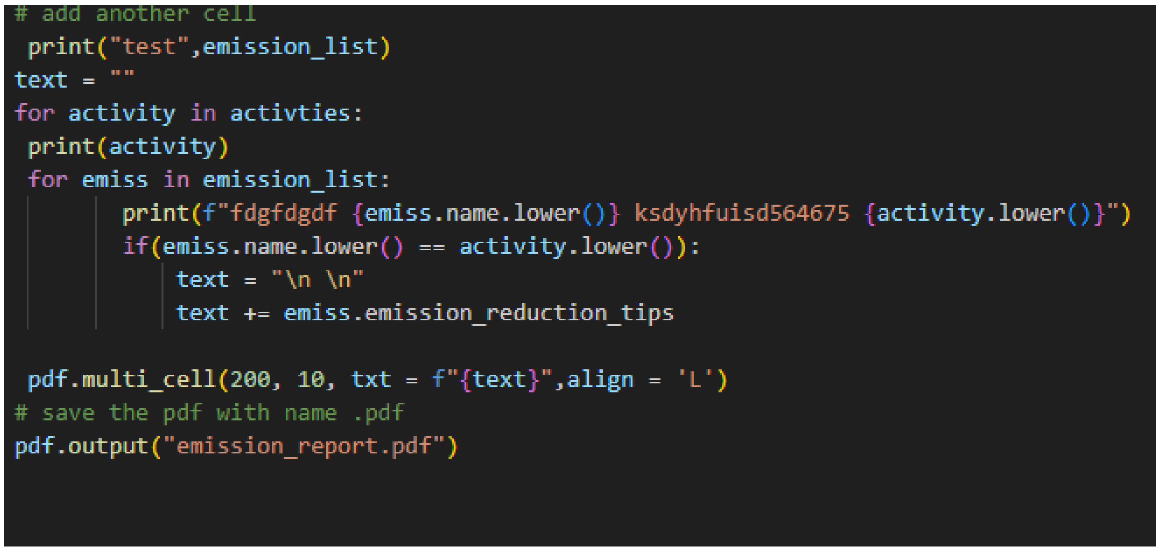
Note that, we create a button called “Create” which will be our last part of code.

Finally, we will generate a .pdf document to the user depending on their inputs from the previous steps on our code, before we dig into the code note that by this point we need to have “FPDF” library ready, once done look at the code below and we can explain it further once we go through it:



```python
def show_report(activties, **kwargs):
    # save FPDF() class into a
    # variable pdf
    pdf = FPDF()

    # Add a page
    pdf.add_page()

    # set style and size of font
    # that you want in the pdf
    pdf.set_font("Arial", size = 15)

    # create a cell
    pdf.cell(200, 10, txt = "Emission Report",
         ln = 1, align = 'C')

    #Get Emission Data from DB
    connection = sqlite3.connect('emissions.db')
    cursor = connection.cursor()

    all_emissions = cursor.execute('Select * from emission')
    data = all_emissions.fetchall()
    print(data)
    emission_list = []
    for e in data:
     print(e)
     emission_list.append(Emission.Emission(*e))
```

Figure 7.1 Generating PDF Code



```python
# add another cell
 print("test",emission_list)
text = ""
for activity in activties:
 print(activity)
 for emiss in emission_list:
         print(f"fdgfdgdf {emiss.name.lower()} ksdyhfuisd564675 {activity.lower()}")
         if(emiss.name.lower() == activity.lower()):
             text = "\n \n"
             text += emiss.emission_reduction_tips

 pdf.multi_cell(200, 10, txt = f"{text}",align = 'L')
# save the pdf with name .pdf
pdf.output("emission_report.pdf")
```

Figure 8.2 Generating PDF Code

As we can see in Figure 6.1 the function is for creating a PDF document, and this function will trigger when the user clicks on “Create” button mentioned on Pie Chart section.

Its fairly customizable, as we can set font type, size, colors,,,etc., we can also set alignment. Perhaps you noticed our database in the middle of the 6.1 snippet of code, here we call the database to get the suggestions on how we can reduce CO2 emissions depending on user inputs, which are in our case (Cars, Trains & Airplanes).

7. Conclusion:

The CO2 Emission Report Generator project has successfully achieved its objectives of providing users with a powerful tool for analyzing and addressing environmental impact. With its user-friendly interface, customizable reports, and actionable insights, it empowers individuals and organizations to make informed decisions and contribute to sustainability efforts.

8. References:

1) GfG (2024) *Graph plotting in Python: Set 1*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/graph-plotting-in-python-set-1/ (Accessed: 03 March 2024).

2) hthomashthomas Python GUI to accept user input in pop up, Stack Overflow. Available at: https://stackoverflow.com/questions/51774639/python-gui-to-accept-user-input-in-pop-up (Accessed: 05 March 2024).

3) Amipara, K. (2019) *Better visualization of PIE charts by Matplotlib*, *Medium*. Available at: https://medium.com/@kvnamipara/a-better-visualisation-of-pie-charts-by-matplotlib-935b7667d77f (Accessed: 20 March 2024).

4) GfG (2022) *Convert text and text file to PDF using Python*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/convert-text-and-text-file-to-pdf-using-python/ (Accessed: 05 March 2024).

5) *SQLite database with python: How to create tables, insert data, and run queries* (2022) *YouTube*. Available at: https://www.youtube.com/watch?v=ZQAnkjfvZAw&ab_channel=ThePyCoach (Accessed: 17 March 2024).

6) Jack EdmondsJack *SQLite add primary key*, *Stack Overflow*. Available at: https://stackoverflow.com/questions/946011/sqlite-add-primary-key (Accessed: 17 March 2024).

7) *Concatenate strings in Python (+ operator, join, etc..)* (no date) *nkmk note*. Available at: https://note.nkmk.me/en/python-string-concat/ (Accessed: 17 March 2024).

8) Cestes *et al. Easier way to read sqlite3 row into object?*, *Stack Overflow*. Available at: https://stackoverflow.com/questions/52823404/easier-way-to-read-sqlite3-row-into-object (Accessed: 17 March 2024).

9) *CO2 Emission Reporting* (2013) *International Chamber of Shipping*. Available at: https://www.ics-shipping.org/current-issue/co2-emission-reporting/ (Accessed: 17 March 2024).

Github Link

https://github.com/wakka-2/EmissionReportCreator