

Created by: The AVELA Team

# Welcome to **AVELA's** Research Study!

Your goal is to complete as many tasks as you can. You can use your AI model and the slides provided for support.

## Introduction: Data Activism for Climate Change

In this activity, we will analyze recent [carbon footprint](#) trends to learn why changes in our climate happen. We will be able determine the impact our carbon footprint has on our environment to help us better understand [climate change](#)!

### ✓ Section 1: Imports

1. Open the dataset by clicking on [this link](#)
2. Download the dataset by clicking on the download icon
3. To upload the dataset on Google Colab, first click the folder icon near the top left of this page
4. Click the upload button and select the "c02\_data.csv" file you previously downloaded

Once you've completed these steps, your Files should look like this:

Use `pd.read_csv` to load the "AVELAMXLX\_Study\_Full.csv" file into a DataFrame called `df_messy`

```
df_messy = pd.read_csv("c02_data.csv")
```

# Display the contents of the DataFrame by typing its variable name

df\_messy

## 4a. Understanding the Data

Before manipulating the data, it's important to understand what it represents and how it applies to the situation or topic being explored.

### **Why are the numbers decimals? Aren't there more CO2 emissions?**

The numbers look "small" because these values actually use really big measuring units (like millions) that would be hard to read and work with in our data frame.

*For Example:*

The value 0.041 has the unit *million metric tons*, which is the equivalent to filling *470 billion* balloons with CO2!

NOTE: You won't need to understand this conversion to complete the activity; it's enough to understand that larger values correlate to more CO2 emissions *by the millions*.

### **NaN?**

Not a Number (NaN) values are used in this table to represent missing or incomplete data.

**NOTE:** Negative values in this table pertain to columns we WON'T be analyzing.

## ✓ **Section 2: Data Cleaning**

To make the Data Frame easier to understand, we'll remove irrelevant columns and rename the long column names to something shorter.

**TASK:** Create a new DataFrame called `2023_global_emissions` using the `df_CO2_stats` dataframe and clean it.

```
ISO Code
Country
Year
Total CO2 Emissions
CO2 Emissions per Person
CO2 from Oil
CO2 from Coal
CO2 from Natural Gas
```

@title ↓ Run this cell to filter the year and only see values from the year 2023. renaming the columns for the DataFrame to make it easier for indexing the columns `df_2023_emissions = df_CO2_stats[df_CO2_stats["Year"] == 2023]` `df_2023_emissions`

title ↓ Run this cell to filter out any NaN (Not a Number) values. `df_2023_only = df_CO2_stats[df_CO2_stats["Year"] == 2023].dropna()` `df_2023_only`

## ✓ Section 3: Basic Coding Concepts

**TASK (Variables):** Declare the variables `country_name`, `emissions_year`, and `emissions_per_person` using the corresponding data points from a row in the `df_2023_emissions` DataFrame.

" Exercise 2: Declaring variables " `country_name = "United States"` `emissions_year = 2023`  
`emissions_per_person = 14.299`

''' Exercise 3: Print Statements '''

➤ Print the country\_name variable that we declared above

```
print(country_name) # print the variable country_name
```

Print the emissions\_year variable that we declared earlier

```
print(emissions_year) # print the variable emissions_year
```

Print the emissions\_year variable first then the country\_name variable second,

then the amount of tons per person as the third.

```
print("\n", emissions_year, country_name, "had", emissions_per_person, "million tons of emissions per person.")
```

↳ 1 cell hidden

""" Exercise 4: Calculate balloon equivalents for CO<sub>2</sub> emissions.

Using the 'emission\_per\_person' column in our 'df\_2023\_only' DataFrame, write conditional statements to calculate how many balloons represent the CO<sub>2</sub> emissions per person in 2023.

NOTE: One balloon can hold about 0.02 metric tons of CO<sub>2</sub>. """

## Calculate how many balloons represent the CO<sub>2</sub>

✓ emissions per person in 2023 as the variable  
'people\_to\_balloons'

```
people_to_balloons = emissions_per_person / 0.02
```

If emissions\_per\_person is less than or equal to 5, print  
"Each person fills 250 or fewer balloons with CO<sub>2</sub>!"

```
if(people_to_balloons <= 5): print("Each person filled 250 or fewer balloons with CO2 in United States in 2023!")
```

Else if emissions\_per\_person is greater than 5 and less than or equal to 15, print "Each person fills between 250 and 750 balloons with CO<sub>2</sub>!"

```
elif(people_to_balloons > 5 and people_to_balloons <= 15): print("Each person filled between 250 and 750 balloons with CO2 in United States in 2023!")
```

Else, print "Each person fills more than 800 big balloons with CO<sub>2</sub>!"

```
else: print("Each person filled more than 800 balloons with CO2 in the United States in 2023!")
```

**TASK (Loops):** Write a for loop that prints the lowest value among the first 10 values from the CO<sub>2</sub> from Natural Gas column in df\_2023\_emissions

" Exercise 6: Use the .iloc method to extract the first 10 rows of the "CO<sub>2</sub> from Oil" column from the DataFrame df\_2023\_emissions and store them in a variable named oil\_em\_country, which holds the emission values for 10 countries. "

✓ Set a starting value that's high to help find the smallest number

```
min_value = 1 # start with a really high percentage
```

Get the first 10 values from the "CO<sub>2</sub> from Natural Gas" column (column index 5)

```
oil_emissions = df_2023_emissions.iloc[0:10, 5]
```

Loop through each emission value in the list

```
for emission in oil_emissions:
```

```
# If the current emission is smaller than the current min_value
if emission < min_value:
    # Update min_value to this smaller emission
    min_value = emission
```

Print out the smallest emission value that was found

```
print("The smallest oil emission percentage is:", min_value)
```

**TASK (Functions):** Write a function that converts oil emissions into their equivalent as balloons filled

" Exercise 7: Declaring functions "

Declare a function named: oil\_emission\_conversion

Name the two input variables: country, CO2

```
def oil_emission_conversion(country, CO2):
```

```
# Calculate the equivalent number of balloons by dividing CO2 emissions by 0.02
# Use int() to convert the result to a whole number (no decimals)
balloons = int(CO2 / 0.02) #int() should used here to turn the result into a who

# Print detailed information about CO2 emissions for the given country
# Convert numeric values to strings with str() to concatenate them with text in
print("In " + country + ",")
print("CO2 emissions from people were about " + str(CO2) + " million metric tons
print("That's similar to " + str(balloons) + " million balloons filled with CO2!
print("")
```

" Exercise 8: Calling Functions "

Pass two input variables into the oil\_emission\_conversion  
function: country\_name and emissions\_per\_person

```
oil_emission_conversion(country_name, emissions_per_person)
```

## Section 4: Creating Visualizations

In this section, you will create a bar graph, pie chart, and heatmap using the data frame we previously created.

*NOTE: Go through these at your own pace. It's okay if you don't finish all 3!*

### Creating a Bar Graph

CONTEXT:

You will create a bar graph that shows CO<sub>2</sub> emissions from oil for the first 7 countries in the dataset.

This activity uses the `df_2023_emissions` DataFrame. You'll focus on the column `CO2 from Oil`, and apply a different color to each bar to help visually compare how much each country contributed through oil-related emissions. This chart helps us explore the role of oil in global CO<sub>2</sub> emissions and how it varies by country.

```
df_few = df_2023_only.iloc[0:7, 3]
```

```
colors = ['green', 'blue', 'pink', 'orange', 'purple', 'yellow', 'teal']
```

### Plot

```
ax = df_few.plot.bar(x="Country", y="Total CO2 Emissions", color=colors)
```

```
ax.set_xlabel("Country") ax.set_ylabel("CO2 Emissions") ax.set_title("CO2 Emissions for the First 7 Countries")
```

```
ax.figure.show()
```



## ✓ Creating a Pie Chart

### CONTEXT:

You will create a pie chart that represents 5 different countries and their CO2 emissions from oil in the year 2023.

This will be based on data from our new `df_5_countries` data frame, and you will visualize this column:

CO2 from Oil

**NOTE:** Make sure to copy+paste your code to the next cell as you progress through this section. Each cell builds upon each other!

> Run this cell to create a new data frame for our 5 countries: United States, India, Germany, Brazil, and Australia

[Show code](#)



[Show hidden output](#)

**TASK:** Create a pie chart using the `df_5_countries` DataFrame to show CO2 emission from oil

""" Exercise 4) Create a pie chart for the df\_5\_countries DataFrame showing CO2 emissions from oil """

✓ Use the df\_5\_countries DataFrame to create a pie chart of "CO2 from Oil"

The 'y' parameter specifies the column to visualize

```
oil_emissions_pie = df_5_countries.plot.pie(y = "CO2 from Oil")
```

display the pie chart

```
oil_emissions_pie
```

**TASK:** Add the percentage of CO2 emissions from oil for each country to your pie chart

""" Exercise 5) Create a pie chart for the df\_5\_countries DataFrame showing CO2 emissions from oil Use the autopct parameter to display the percentage each slice represents """

✓ Use the df\_5\_countries DataFrame to create a pie chart of CO2 emissions from oil

autopct='%1.0f%%' formats the percentage labels with no decimals

```
oil_emissions_pie = df_5_countries.plot.pie(y = "CO2 from Oil", autopct = '%1.0f%%')
```

visualize the pie chart

```
oil_emissions_pie
```

**TASK:** Remove the legend from your pie chart.

" Exercise 6) Create a pie chart for the df\_5\_countries DataFrame showing CO2 emissions from oil Use the autopct parameter to display the percentage each slice represents Remove the legend from the chart "

✓ Create a pie chart of CO2 emissions from oil using df\_5\_countries

autopct='%1.0f%%' shows percentage labels without decimals

```
oil_emissions_pie = df_5_countries.plot.pie(y = "CO2 from Oil", autopct = '%1.0f%%')
```

visualize the pie chart

```
oil_emissions_pie.get_legend().remove() # remove the legend oil_emissions_pie
```

**TASK:** Make finishing touches to your pie chart

" Exercise 7) Create a pie chart for the df\_5\_countries DataFrame for the total for your chosen type of CO2 emission using the autopct parameter to show the percentages of each sector of the chart, removing the legend, and adding a title that describes the type of CO2 emission you chose, ending with "Comparison from 5 Different Countries in 2023" "

use the df\_wash DataFrame that we created in the first code cell

```
oil_emissions_pie = df_5_countries.plot.pie(y = "CO2 from Oil", autopct = '%1.0f%%', labels = df_5_countries.loc[:, "Country"], ylabel="", title = "CO2 Emissions Comparison from 5 Different Countries in 2023")
```

make these question marks

remove the legend

```
oil_emissions_pie.get_legend().remove()
```

visualize the pie chart

```
oil_emissions_pie
```

## Creating a Heatmap

CONTEXT:

In this activity, you'll create a heatmap on a world map that shows how much total CO<sub>2</sub> each country emitted in the year 2023.

You'll use the column `Total CO2 Emissions` to color each country based on how much it contributed to global emissions.

" The choropleth function is called on the Plotly Express library The first input parameter into the choropleth function is 'df\_2023\_only' Set locations = "ISO Code", locationmode = "ISO-3", color = "Total CO2 Emissions" The color\_continuous\_scale can also = "Viridis" "

```
fig2 = px.choropleth(df_2023_only, locations="ISO Code", locationmode="ISO-3", color="Total CO2 Emissions", color_continuous_scale="Turbo", title="Total CO2 Emissions by Country (2023)" )
```

## Center the title and show the map

```
fig2.update_layout(title_x=0.5) fig2.show()
```

## End of Activity

**Stop and think:** What are some things our charts don't show about how these emissions affect people or the environment? How might the data be different if we looked at other years or more countries?"