Project Code Used for the Power BI Visuals

# Per Gen Graph:

import matplotlib.pyplot as plt

import seaborn as sns

# Counting Pokémon per Generation

poke\_count\_per\_gen = dataset['Generation'].value\_counts().sort\_index().reset\_index()

poke\_count\_per\_gen.columns = ['Generation', 'count']

# Plotting out all the new pokemon in each generation

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

sns.barplot(x='Generation', y='count', data=poke\_count\_per\_gen, palette='muted')

plt.xlabel('Generation')

plt.ylabel('Number of Pokémon')

plt.title('Number of Pokémon per Generation')

plt.show()

# Pokemon Per Type Graph:

import pandas as pd

import matplotlib.pyplot as plt

# Set generation\_input to "All" by default

generation\_input = 'All'

# Filter the DataFrame based on the user's input

if generation\_input.lower() == 'all':

    filtered\_data = dataset

else:

    try:

        generation\_number = int(generation\_input)

        filtered\_data = dataset[dataset['Generation'] == generation\_number]

    except ValueError:

        print("Invalid input. Please enter a generation number (1-9) or 'All'.")

        exit()

# Calculate the count of Pokémon for each type

type\_counts = pd.concat([filtered\_data['Type 1'], filtered\_data['Type 2']]).value\_counts()

# Define a color map for Pokémon types

type\_colors = {

    'Normal': 'lightgray',

    'Fire': 'red',

    'Water': 'blue',

    'Electric': 'gold',

    'Grass': 'green',

    'Ice': 'lightblue',

    'Fighting': 'saddlebrown',

    'Poison': 'purple',

    'Ground': 'darkorange',

    'Flying': 'skyblue',

    'Psychic': 'pink',

    'Bug': 'limegreen',

    'Rock': 'tan',

    'Ghost': 'indigo',

    'Dragon': 'darkblue',

    'Dark': 'black',

    'Steel': 'gray',

    'Fairy': 'hotpink',

}

# Plot the bar graph with corresponding colors

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

type\_counts.plot(kind='bar', color=[type\_colors[type] for type in type\_counts.index])

plt.title(f'Number of Pokémon for Each Type in Generation {generation\_input.upper()}')

plt.xlabel('Type')

plt.ylabel('Count')

plt.xticks(rotation=45)

plt.grid(axis='y', linestyle='--', alpha=0.7)

# Add text labels on top of each bar

for i, count in enumerate(type\_counts):

    plt.text(i, count + 0.2, str(count), ha='center')

plt.tight\_layout()

# Show the plot

plt.show()

Average of Each Stat Graph:

import pandas as pd

import matplotlib.pyplot as plt

# Define the mapping of stats to column names

stat\_columns = {

    'Total': 'Total',

    'HP': 'HP',

    'Attack': 'Attack',

    'Defense': 'Defense',

    'Sp. Atk': 'Sp. Atk',

    'Sp. Def': 'Sp. Def',

    'Speed': 'Speed'

}

# Set the default stat to analyze as 'All'

stat\_input = 'All'

# Check if the input stat is valid

if stat\_input not in stat\_columns and stat\_input != 'All':

    print("Invalid input. Please enter a valid Pokémon stat or 'All'.")

    exit()

# Calculate the average of each stat for all Pokémon

if stat\_input == 'All':

    stat\_averages = {stat: dataset[column].mean() for stat, column in stat\_columns.items()}

    plt.bar(stat\_averages.keys(), stat\_averages.values(), color='skyblue')

    plt.title('Average of Each Pokémon Stat')

    plt.xlabel('Stat')

    plt.ylabel('Average Value')

    plt.xticks(rotation=45)

    plt.grid(axis='y', linestyle='--', alpha=0.7)

    # Annotate each bar with its exact value

    for stat, value in stat\_averages.items():

        plt.text(stat, value, str(round(value, 2)), ha='center', va='bottom', fontsize=8)

    plt.tight\_layout()

    plt.show()

    exit()

# Calculate the average of the selected stat for each Pokémon

dataset['Average'] = dataset[stat\_columns[stat\_input]]

# Plot the bar graph for average total of each Pokémon stat

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

plt.bar(dataset.index, dataset['Average'], color='skyblue')

plt.title(f'Average {stat\_input} of Each Pokémon')

plt.xlabel('Pokémon')

plt.ylabel(f'Average {stat\_input}')

plt.xticks(rotation=90)

plt.grid(axis='y', linestyle='--', alpha=0.7)

# Annotate each bar with its exact value

for index, value in enumerate(dataset['Average']):

    plt.text(index, value, str(round(value, 2)), ha='center', va='bottom', fontsize=8)

plt.tight\_layout()

# Show the plot

plt.show()

Average of a Specific Stat by Generation Graph (The code is the same for each graph, just switching the stat used):

import pandas as pd

import matplotlib.pyplot as plt

# Define the mapping of stats to column names

stat\_columns = {

    'Total': 'Total',

    'HP': 'HP',

    'Attack': 'Attack',

    'Defense': 'Defense',

    'Sp. Atk': 'Sp. Atk',

    'Sp. Def': 'Sp. Def',

    'Speed': 'Speed'

}

# Set the default stat to analyze as 'Total'

stat\_input = 'Total'

# Check if the input stat is valid

if stat\_input not in stat\_columns and stat\_input != 'All':

    print("Invalid input. Please enter a valid Pokémon stat or 'All'.")

    exit()

# Group the data by generation and calculate the average total stats for each generation

generation\_averages = dataset.groupby('Generation')[stat\_columns[stat\_input]].mean()

# Plot the bar graph for average total stats of each generation

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

generation\_averages.plot(kind='bar', color='skyblue')

plt.title(f'Average {stat\_input} of Each Pokémon Generation')

plt.xlabel('Generation')

plt.ylabel(f'Average {stat\_input}')

plt.xticks(rotation=0)

plt.grid(axis='y', linestyle='--', alpha=0.7)

# Annotate each bar with its exact value

for index, value in enumerate(generation\_averages.values):

    plt.text(index, value, str(round(value, 2)), ha='center', va='bottom', fontsize=8)

plt.tight\_layout()

# Show the plot

plt.show()

Average of a Specific Stat by Type Graph (The code is the same for each graph, just switching the stat used):

import pandas as pd

import matplotlib.pyplot as plt

# Calculate the average "Total" stat for each type

type1\_averages = dataset.groupby('Type 1')['Total'].mean()

type2\_averages = dataset.groupby('Type 2')['Total'].mean()

# Combine the averages for Type 1 and Type 2, ensuring no duplicates

all\_types = set(type1\_averages.index).union(type2\_averages.index)

combined\_averages = {typ: (type1\_averages.get(typ, 0) + type2\_averages.get(typ, 0)) / 2 for typ in all\_types}

# Sort the combined averages dictionary by values (average total stats)

sorted\_averages = dict(sorted(combined\_averages.items(), key=lambda item: item[1], reverse=True))

# Define a color map for Pokémon types

type\_colors = {

    'Normal': 'lightgray',

    'Fire': 'red',

    'Water': 'blue',

    'Electric': 'gold',

    'Grass': 'green',

    'Ice': 'lightblue',

    'Fighting': 'saddlebrown',

    'Poison': 'purple',

    'Ground': 'darkorange',

    'Flying': 'skyblue',

    'Psychic': 'pink',

    'Bug': 'limegreen',

    'Rock': 'tan',

    'Ghost': 'indigo',

    'Dragon': 'darkblue',

    'Dark': 'black',

    'Steel': 'gray',

    'Fairy': 'hotpink',

}

# Plot the bar graph with corresponding colors

plt.figure(figsize=(12, 6))

colors = [type\_colors.get(typ, 'gray') for typ in sorted\_averages.keys()]

plt.bar(sorted\_averages.keys(), sorted\_averages.values(), color=colors)

plt.title('Average Total Stat for Each Pokémon Type')

plt.xlabel('Type')

plt.ylabel('Average Total Stat')

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

plt.grid(axis='y', linestyle='--', alpha=0.7)

# Annotate each bar with its exact value

for index, value in enumerate(sorted\_averages.values()):

    plt.text(index, value, str(round(value, 2)), ha='center', va='bottom', fontsize=8)

plt.tight\_layout()

# Show the plot

plt.show()