DMBI Practical 3

Name: Ishaan Khan
Class: D15C
Roll No: 57
Batch: C

Aim: To perform Exploratory Data Analysis and visualization using python.

Theory:

- Exploratory Data Analysis (EDA) is an essential step in data science that focuses on investigating datasets to uncover patterns, detect anomalies, test assumptions, and generate insights using statistical and visual methods. It helps identify missing values, outliers, distributions, and relationships among variables, guiding feature engineering and hypothesis testing.
- In this experiment, EDA is applied to a Pokémon dataset, which includes features like primary type (Type 1), HP, Attack, Defense, Special Attack, Special Defense, Speed, and Legendary status. The process involves data loading, cleaning, summary statistics, and visualization to extract insights such as type distribution, statistical patterns, and correlations between different battle stats.
- Data visualization translates raw data into intuitive formats such as charts and graphs, enabling easier detection of trends, outliers, and relationships.

Python libraries used:

- **Pandas:** Data manipulation (DataFrames, CSV reading, summary statistics).
- **NumPy:** Numerical operations and NaN handling.
- **Matplotlib.pyplot:** Basic plotting (figures, labels, visualization).
- **Seaborn:** Statistical visualizations (histograms, bar plots, heatmaps).

Syntax & Graphs applied in this experiment:

- **Histogram:** Shows frequency distribution of a continuous variable like the 'Total' stat.
    - Syntax: sns.histplot(data=df, x='Total', bins=20, hue='Legendary', kde=True)

- **Bar Plot:** Compares categorical and numerical variables. A countplot (a type of bar plot) is used to show the number of Pokémon per primary type.
    - Syntax: sns.countplot(data=df, x='Type 1', order = df['Type 1'].value_counts().index, palette='pastel')

- **Box Plot:** Highlights spread, quartiles, and outliers; used for comparing the 'Attack' stat across different primary types.
    - Syntax: sns.boxplot(x='Type 1', y='Attack', data=df, palette='Set3')

- **Heatmap:** Displays a correlation matrix with color coding, showing relationships between numerical stats like Attack, Defense, and HP.
    - Syntax: sns.heatmap(df_corr.corr(), annot=True, cmap='coolwarm', fmt=".2f")

- **Scatter Plot:** Explores relationships between two continuous variables, such as Attack vs. Defense, with Legendary status as hue.
    - Syntax: sns.scatterplot(x='Attack', y='Defense', hue='Legendary', data=df, alpha=0.7)

- **Pie Chart:** Represents proportions, such as the distribution of Pokémon by their primary type.
    - Syntax: plt.pie(type_counts, labels=type_counts.index, autopct='%1.1f%%', startangle=140, colors=colors)

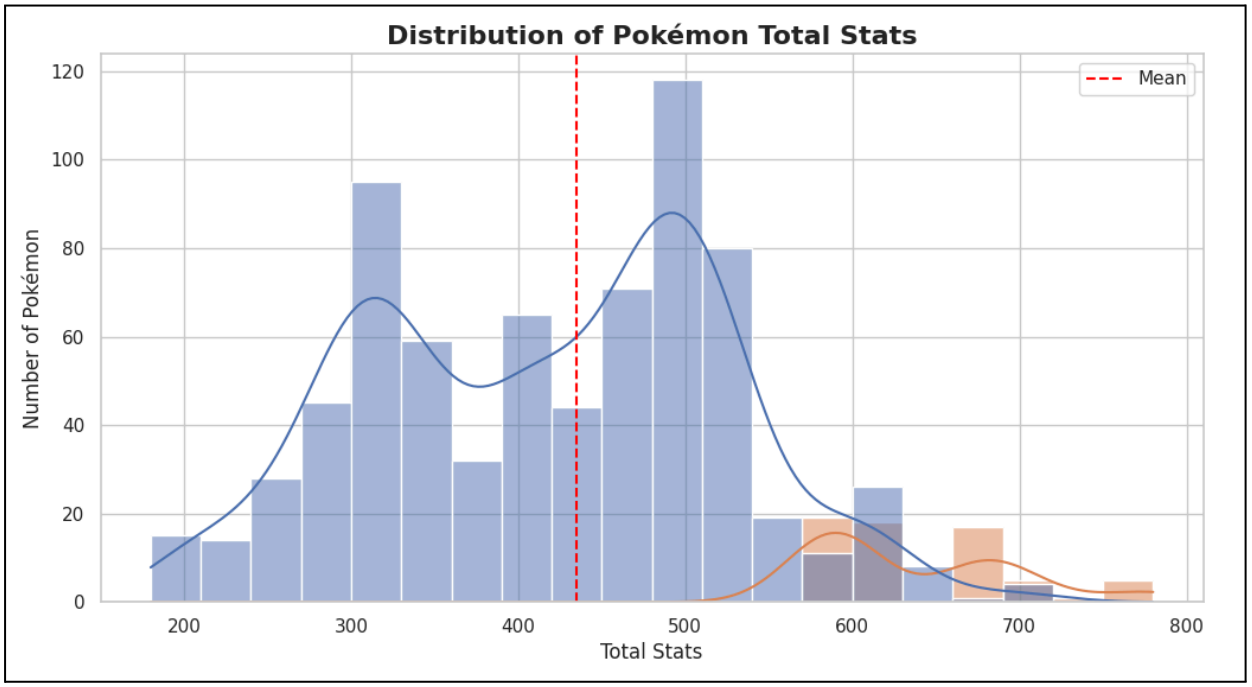
Conclusion:

In this experiment, Exploratory Data Analysis (EDA) was performed on a Pokémon dataset using Python. The dataset was cleaned and preprocessed to handle inconsistent naming for features like Mega and Primal evolutions. Various visualization techniques including histograms, pie charts, box plots, and heatmaps were applied to analyze type distribution, stat variations, and relationships among different battle statistics such as Attack, Defense, and Total stats.

The analysis provided clear insights into the structure of the dataset and revealed patterns, trends, and correlations that would not have been evident from raw data alone. Challenges such as inconsistent text in Pokémon names were addressed using appropriate data cleaning methods. Overall, the experiment demonstrated the effectiveness of EDA in transforming a raw dataset into meaningful information and reinforced the importance of visualization in data analysis.

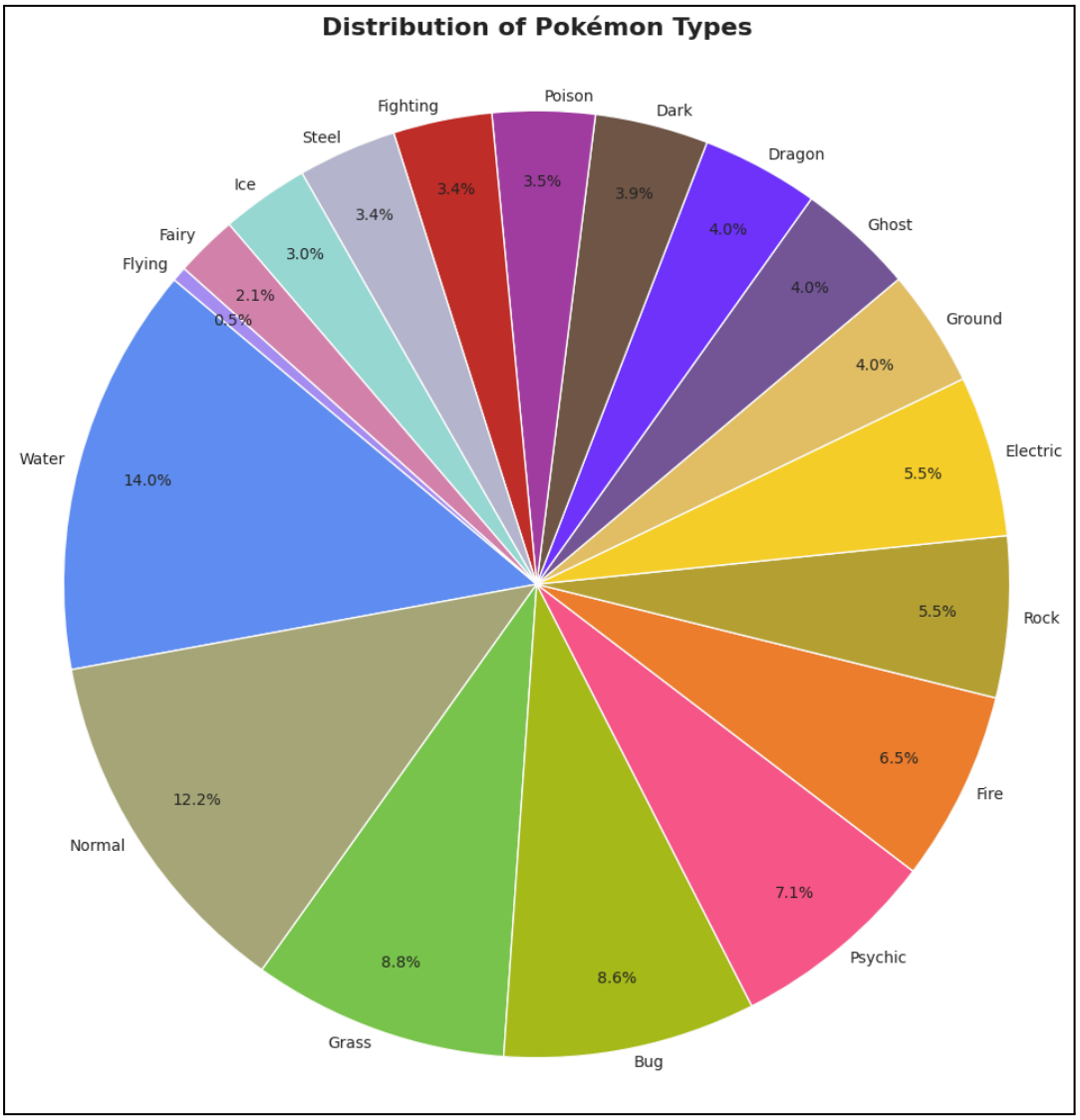
Code & Output

```
#HISTOGRAM

# Show plots inline
%matplotlib inline

# Set the style of seaborn
sns.set(style="whitegrid")

# Define the file path for the Pokémon dataset and read the CSV
file_path = 'Pokemon.csv'
df = pd.read_csv(file_path)

# --- Histogram Code ---
plt.figure(figsize=(12, 6))
sns.histplot(data=df, x='Total', bins=20, hue='Legendary', kde=True)
plt.title('Distribution of Pokémon Total Stats', fontsize=16, fontweight='bold')
plt.xlabel('Total Stats')
plt.ylabel('Number of Pokémon')
plt.axvline(df['Total'].mean(), color='red', linestyle='dashed', label='Mean')
plt.legend()
plt.show()
```

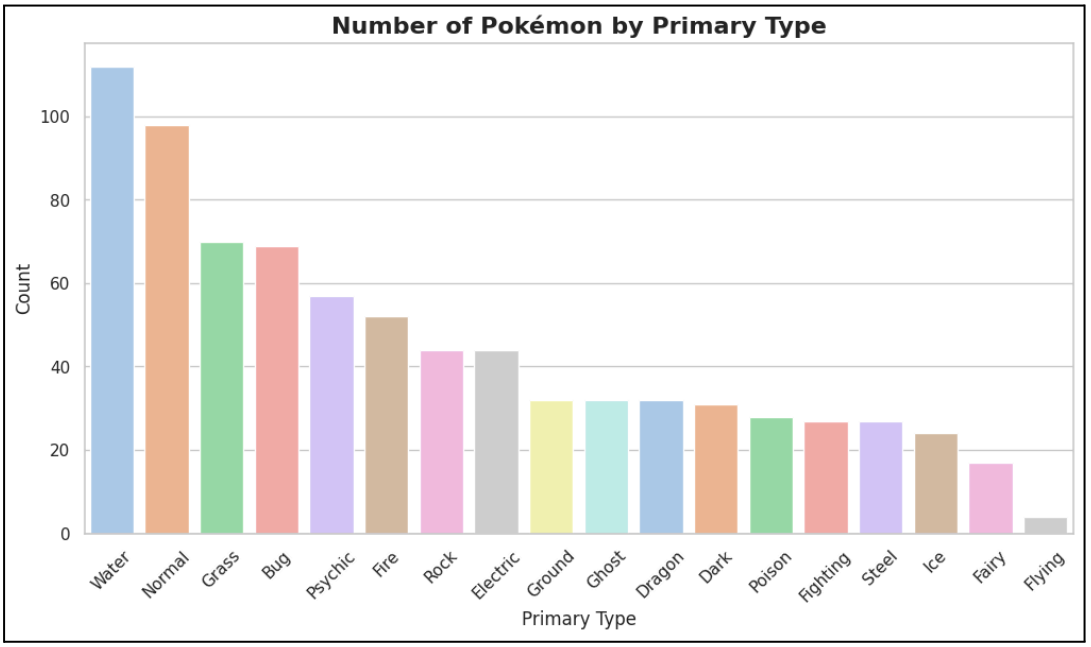


0s

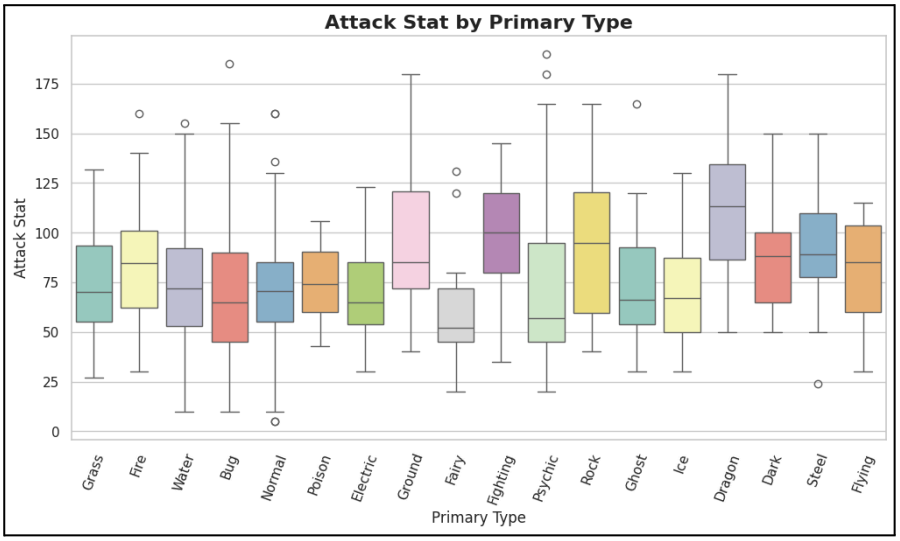
```
type_colors = {
    'Water': '#6390F0', 'Fire': '#EE8130', 'Grass': '#7AC74C', 'Electric': '#F7D02C',
    'Psychic': '#F95587', 'Ice': '#96D9D6', 'Dragon': '#6F35FC', 'Dark': '#705746',
    'Fighting': '#C22E28', 'Poison': '#A33EA1', 'Ground': '#E2BF65', 'Flying': '#A98FF3',
    'Bug': '#A6B91A', 'Rock': '#B6A136', 'Ghost': '#735797', 'Steel': '#B7B7CE',
    'Normal': '#A8A878', 'Fairy': '#D685AD'
}

# Get the counts and corresponding colors
type_counts = df['Type 1'].value_counts()
colors = [type_colors.get(ptype, '#CCCCCC') for ptype in type_counts.index]

plt.figure(figsize=(12, 12))
plt.pie(type_counts, labels=type_counts.index, autopct='%1.1f%%',
        startangle=140, colors=colors, textprops={'fontsize': 10},
        labeldistance=1.03, pctdistance=0.85)
plt.title('Distribution of Pokémon Types', fontsize=16, fontweight='bold', pad=20)
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```



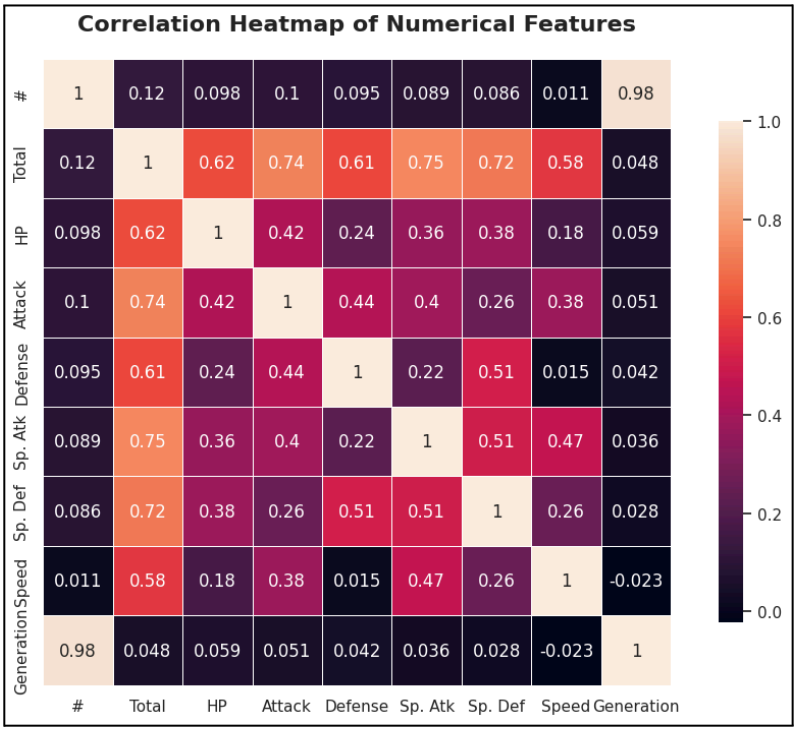
```python
# Bar Plot Code
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x='Type 1', order = df['Type 1'].value_counts().index,
              palette='pastel')
plt.title('Number of Pokémon by Primary Type', fontsize=16, fontweight='bold')
plt.xlabel('Primary Type')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

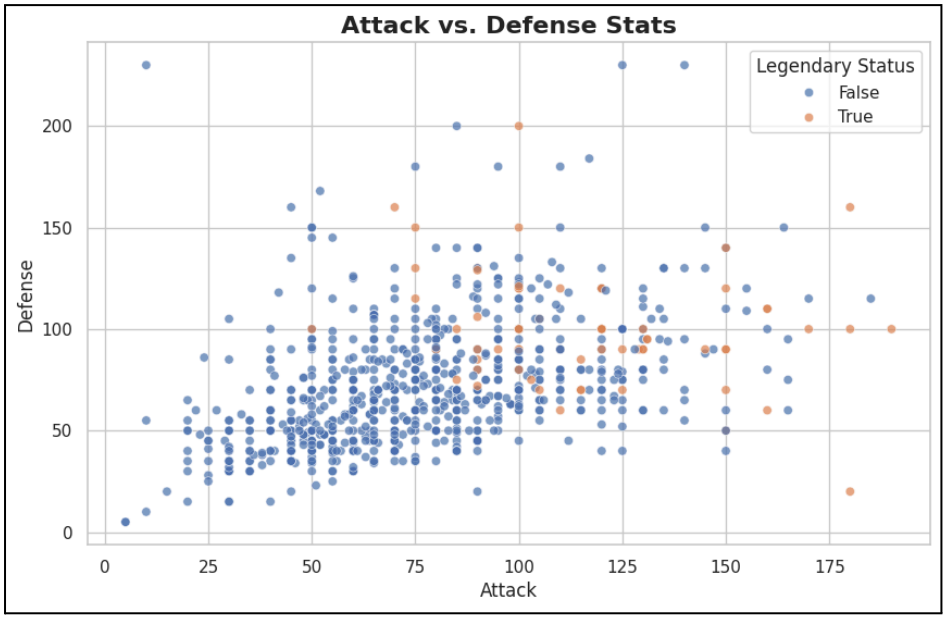



```python
# Attack stat by Primary Type
plt.figure(figsize=(12, 6))
sns.boxplot(x='Type 1', y='Attack',
            data=df, palette='Set3')
plt.xticks(rotation=70)
plt.xlabel('Primary Type')
plt.ylabel('Attack Stat')
plt.title('Attack Stat by Primary Type',
          fontsize=16, fontweight='bold')
plt.show()
```



```python
#HEATMAP
# Correlation heatmap of numerical features
plt.figure(figsize=(12, 8))
numerical_cols = df.select_dtypes(include=[np.number]).columns
sns.heatmap(df[numerical_cols].corr(), annot=True, square=True,
            cbar_kws={"shrink": .8}, linewidths=0.5)
plt.title('Correlation Heatmap of Numerical Features',
          fontsize=16, fontweight='bold', pad=20)
plt.show()
```






```python
# Scatter Plot Code
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Attack', y='Defense',
                hue='Legendary', data=df,
                alpha=0.7)
plt.title('Attack vs. Defense Stats',
          fontsize=16,
          fontweight='bold')
plt.xlabel('Attack')
plt.ylabel('Defense')
plt.legend(title='Legendary Status')
plt.show()
```