Name:Khushi Singh
Class:D15C Roll no:71

Experiment No. 3

Aim:To perform Exploratory Data Analysis and visualization using python
Theory:
Exploratory Data Analysis (EDA) is an essential step in the data analysis process that involves summarizing and visualizing the dataset to understand its underlying structure, detect patterns, identify anomalies, and generate insights. It helps in making informed decisions about data preprocessing, feature engineering, and model selection in later stages of analysis.

EDA consists of descriptive statistics, data visualization, and correlation analysis, which provide a comprehensive understanding of the dataset.

Since our dataset consists of different Pokémon with attributes such as type, generation, legendary status, and battle statistics (HP, Attack, Defense, Speed, Special Attack, Special Defense, Total), EDA will help us uncover trends related to Pokémon types, strengths, weaknesses, and distribution across generations.

Descriptive Analysis – Central Tendency
Central tendency refers to the measure that represents the center or typical value of a dataset. The three common measures are:

- Mean – Average value.
- Median – Middle value when sorted.
- Mode – Most frequent value.

Execution:
For our dataset, we calculate the mean, median, and mode for numerical variables such as:

- HP (Health Points)
- Attack
- Defense
- Speed
- Total Stats

Inference:

- If the mean Attack is higher than the median, it suggests a few very powerful Pokémon dominate the dataset.
- The mode of Type may highlight the most common Pokémon type.

Descriptive Analysis – Dispersion

Dispersion measures how spread out the data is. The key measures are:

- Range – Difference between max and min.
- Variance & Standard Deviation – Spread around the mean.
- Interquartile Range (IQR) – 50% spread.

Execution:

We compute dispersion metrics for HP, Attack, and Speed.

Inference:

- A high standard deviation in Speed indicates wide variation (very slow vs very fast Pokémon).
- Outliers in Attack may represent Legendary Pokémon with unusually high stats.

Correlation Analysis

Correlation measures the relationship between two numerical variables.

- Pearson Correlation: Linear relationships (range: -1 to +1).

Execution:

We calculate correlation between:

- Attack and Defense
- HP and Total
- Speed and Attack

Inference:

- Strong correlation between Total and HP/Attack shows these features directly influence the total stat value.
- Weak correlation between Defense and Speed indicates defensive Pokémon are not necessarily fast.

Data Visualization

1. **Histogram (Distribution of Attack)**

- Shows how Pokémon attacks are distributed.
  Inference: If right-skewed, most Pokémon have moderate attack while only a few are extremely powerful.

2. **Box Plot (HP by Legendary Status)**

- Compares health of Legendary vs Non-Legendary Pokémon.
  Inference: Legendary Pokémon will likely show higher median HP with outliers.

3. **Scatter Plot (Attack vs Defense)**

- Shows trade-off between attack and defense.
  Inference: Some Pokémon with high attack may have low defense, highlighting glass-cannon Pokémon.

4. **Bar Chart (Count of Pokémon by Type)**

- Compare the number of Pokémon for each type.
  Inference: Some types (like Water) are more common than others (like Ice).

5. **Heatmap (Correlation Matrix)**

- Shows relationships among all numerical stats.
  Inference: Total is highly correlated with Attack, Defense, and HP, proving total stat is a sum-based measure.

Code and Output:

```
import pandas as pd
import numpy as np
import re
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style="whitegrid")
pd.set_option('display.max_columns', None)
```

```
[2] from google.colab import files
    uploaded = files.upload()  # upload pokemonDB_dataset.csv

    df = pd.read_csv("pokemonDB_dataset.csv")
    print(df.shape)
    df.head()
```

```
Choose files pokemonDB_dataset.csv
• pokemonDB_dataset.csv(text/csv) - 388865 bytes, last modified: 24/08/2025 - 100% done
Saving pokemonDB_dataset.csv to pokemonDB_dataset.csv
(1215, 32)
```

	Pokemon	Type	Species	Height	Weight	Abilities	EV Yield	Catch Rate	Base Friendship	Base Exp	Growth Rate	Egg Groups	Gender	Egg Cycles	HP Base	HP Min	HP Max
0	Abomasnow	Grass, Ice	Frost Tree Pokémon	2.2 m (7′03″)	135.5 kg (298.7 lbs)	1. Snow Warning, Soundproof (hidden ability)	1 Attack, 1 Sp. Atk	60 (7.8% with PokéBall, full HP)	50 (normal)	173	Slow	Grass, Monster	50% male, 50% female	20 (4,884–5,140 steps)	90	290	384
1	Mega Abomasnow	Grass, Ice	Frost Tree Pokémon	2.7 m (8′10″)	185.0 kg (407.9 lbs)	1. Snow Warning	1 Attack, 1 Sp. Atk	60 (7.8% with PokéBall, full HP)	50 (normal)	208	Slow	Grass, Monster	50% male, 50% female	20 (4,884–5,140 steps)	90	290	384
2	Abra	Psychic	Psi Pokémon	0.9 m (2′11″)	19.5 kg (43.0 lbs)	1. Synchronize, 2. Inner Focus, Magic Guard (h...	1 Sp. Atk	200 (26.1% with PokéBall, full HP)	50 (normal)	62	Medium Slow	Human-Like	75% male, 25% female	20 (4,884–5,140 steps)	25	160	254
3	Absol	Dark	Disaster Pokémon	1.2 m (3′11″)	47.0 kg (103.6 lbs)	1. Pressure, 2. Super Luck, Justified (hidden ...	2 Attack	30 (3.9% with PokéBall, full HP)	35 (lower than normal)	163	Medium Slow	Field	50% male, 50% female	25 (6,169–6,425 steps)	65	240	334
4	Mega Absol	Dark	Disaster Pokémon	1.2 m (3′11″)	49.0 kg (108.0 lbs)	1. Magic Bounce	2 Attack	30 (3.9% with PokéBall, full HP)	35 (lower than normal)	198	Medium Slow	Field	50% male, 50% female	25 (6,169–6,425 steps)	65	240	334

```python
# --- Split Type into Type 1 and Type 2
df['Type 1'] = df['Type'].str.split(',').str[0].str.strip()
df['Type 2'] = df['Type'].str.split(',').str[1].str.strip()
df['Type 2'] = df['Type 2'].fillna('None')

# --- Helpers to parse numbers from text
def extract_leading_number(s):
    if pd.isna(s):
        return np.nan
    m = re.match(r'\s*(\d+(?:\.\d+)?)', str(s))
    return float(m.group(1)) if m else np.nan

def parse_height_m(s):
    if pd.isna(s): return np.nan
    m = re.match(r'\s*(\d+(?:\.\d+)?)\s*m', str(s))
    return float(m.group(1)) if m else np.nan

def parse_weight_kg(s):
    if pd.isna(s): return np.nan
    m = re.match(r'\s*(\d+(?:\.\d+)?)\s*kg', str(s))
    return float(m.group(1)) if m else np.nan

# --- Numeric extractions
df['Base Exp Num']        = df['Base Exp'].apply(extract_leading_number)
df['Catch Rate Num']      = df['Catch Rate'].apply(extract_leading_number)
df['Base Friendship Num'] = df['Base Friendship'].apply(extract_leading_number)
df['Height_m']            = df['Height'].apply(parse_height_m)
df['Weight_kg']           = df['Weight'].apply(parse_weight_kg)

# --- Total of base stats present in your file
base_stats = [
    'HP Base','Attack Base','Defense Base',
    'Special Attack Base','Special Defense Base','Speed Base'
]
df['Total Base'] = df[base_stats].sum(axis=1)

# Quick sanity check
df[['Pokemon','Type','Type 1','Type 2','Base Exp','Base Exp Num','Catch Rate','Catch Rate Num','Height','Height_m','Weight','Weight_kg','Total Base']].head()
```

	Pokemon	Type	Type 1	Type 2	Base Exp	Base Exp Num	Catch Rate	Catch Rate Num	Height	Height_m	Weight	Weight_kg	Total Base
0	Abomasnow	Grass, Ice	Grass	Ice	173	173.0	60 (7.8% with PokéBall, full HP)	60.0	2.2 m (7'03")	2.2	135.5 kg (298.7 lbs)	135.5	494
1	Mega Abomasnow	Grass, Ice	Grass	Ice	208	208.0	60 (7.8% with PokéBall, full HP)	60.0	2.7 m (8'10")	2.7	185.0 kg (407.9 lbs)	185.0	594
2	Abra	Psychic	Psychic	None	62	62.0	200 (26.1% with PokéBall, full HP)	200.0	0.9 m (2'11")	0.9	19.5 kg (43.0 lbs)	19.5	310
3	Absol	Dark	Dark	None	163	163.0	30 (3.9% with PokéBall, full HP)	30.0	1.2 m (3'11")	1.2	47.0 kg (103.6 lbs)	47.0	465
4	Mega Absol	Dark	Dark	None	198	198.0	30 (3.9% with PokéBall, full HP)	30.0	1.2 m (3'11")	1.2	49.0 kg (108.0 lbs)	49.0	565

```python
print("Shape:", df.shape, "\n")
print("DTypes:\n", df.dtypes, "\n")

print("Missing values per column:\n", df.isnull().sum(), "\n")
print("Duplicate rows:", df.duplicated().sum())
```

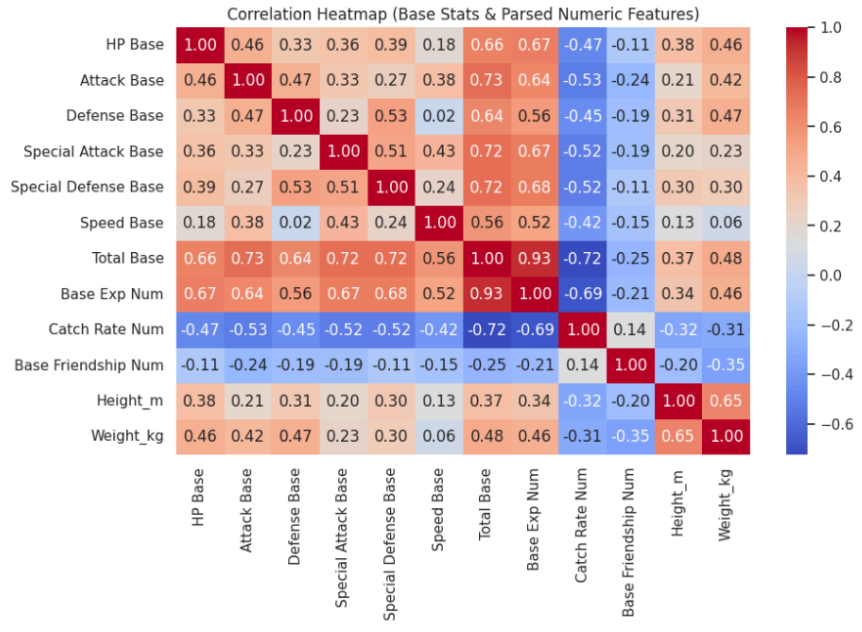
```
Shape: (1215, 40)

DTypes:
 Pokemon                 object
Type                     object
Species                  object
Height                   object
Weight                   object
Abilities                object
EV Yield                 object
Catch Rate               object
Base Friendship          object
Base Exp                 object
Growth Rate              object
Egg Groups               object
Gender                   object
Egg Cycles               object
HP Base                   int64
HP Min                    int64
HP Max                    int64
Attack Base               int64
Attack Min                int64
Attack Max                int64
Defense Base              int64
Defense Min               int64
Defense Max               int64
Special Attack Base       int64
Special Attack Min        int64
Special Attack Max        int64
Special Defense Base      int64
Special Defense Min       int64
Special Defense Max       int64
Speed Base                int64
Speed Min                 int64
Speed Max                 int64
```

```
Missing values per column:
 Pokemon                   0
Type                       0
Species                    0
Height                     0
Weight                     0
Abilities                  0
EV Yield                   0
Catch Rate                 0
Base Friendship            0
Base Exp                   0
Growth Rate                0
Egg Groups                 0
Gender                     0
Egg Cycles                 0
HP Base                    0
HP Min                     0
HP Max                     0
Attack Base                0
Attack Min                 0
Attack Max                 0
Defense Base               0
Defense Min                0
Defense Max                0
Special Attack Base        0
Special Attack Min         0
Special Attack Max         0
Special Defense Base       0
Special Defense Min        0
Special Defense Max        0
Speed Base                 0
Speed Min                  0
Speed Max                  0
Type 1                     0
Type 2                     0
Base Exp Num              23
Catch Rate Num             1
Base Friendship Num       23
Height_m                   0
Weight_kg                  1
Total Base                 0
dtype: int64
```

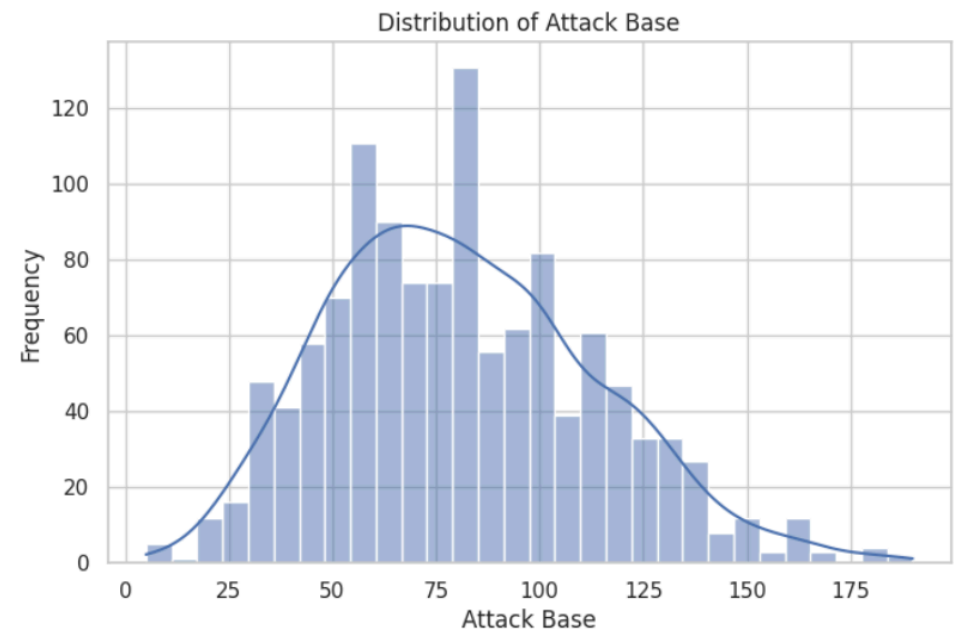
```
numeric_cols = base_stats + ['Total Base','Base Exp Num','Catch Rate Num','Base Friendship Num','Height_m','Weight_kg']

desc = df[numeric_cols].describe().T
desc[['mean','50%','std','min','max']].rename(columns={'50%':'median'})
```

	mean	median	std	min	max
HP Base	71.244444	70.0	26.927819	1.0	255.0
Attack Base	81.152263	80.0	32.037134	5.0	190.0
Defense Base	75.007407	70.0	30.740999	5.0	250.0
Special Attack Base	73.224691	65.0	32.757152	10.0	194.0
Special Defense Base	72.441152	70.0	27.578188	20.0	250.0
Speed Base	70.034568	68.0	30.161298	5.0	200.0
Total Base	443.104527	465.0	121.193406	175.0	1125.0
Base Exp Num	159.199664	162.0	84.159281	36.0	635.0
Catch Rate Num	90.615321	60.0	75.330475	3.0	255.0
Base Friendship Num	47.151846	50.0	18.511020	0.0	140.0
Height_m	1.378354	1.0	3.150568	0.1	100.0
Weight_kg	73.883526	30.0	133.733048	0.1	999.9

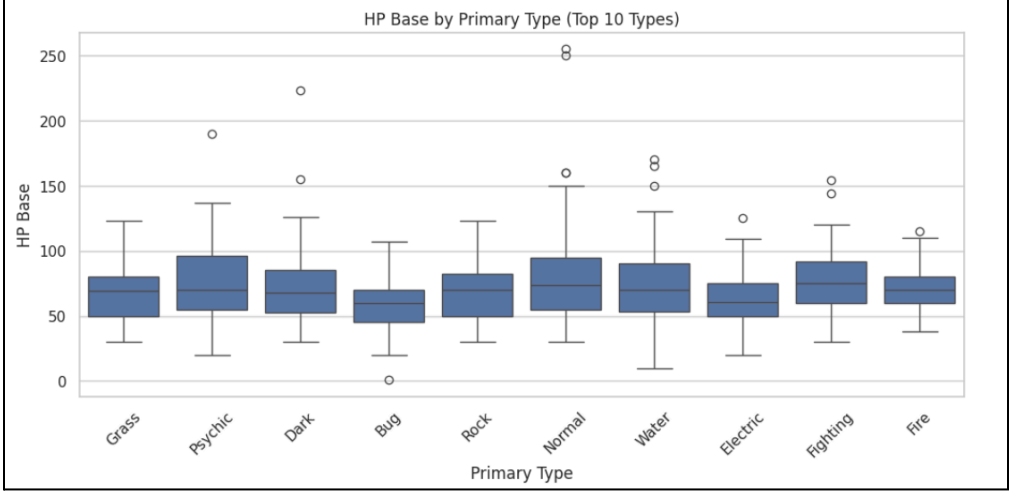
```
corr = df[numeric_cols].corr()
plt.figure(figsize=(10,6))
sns.heatmap(corr, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap (Base Stats & Parsed Numeric Features)")
plt.show()
```



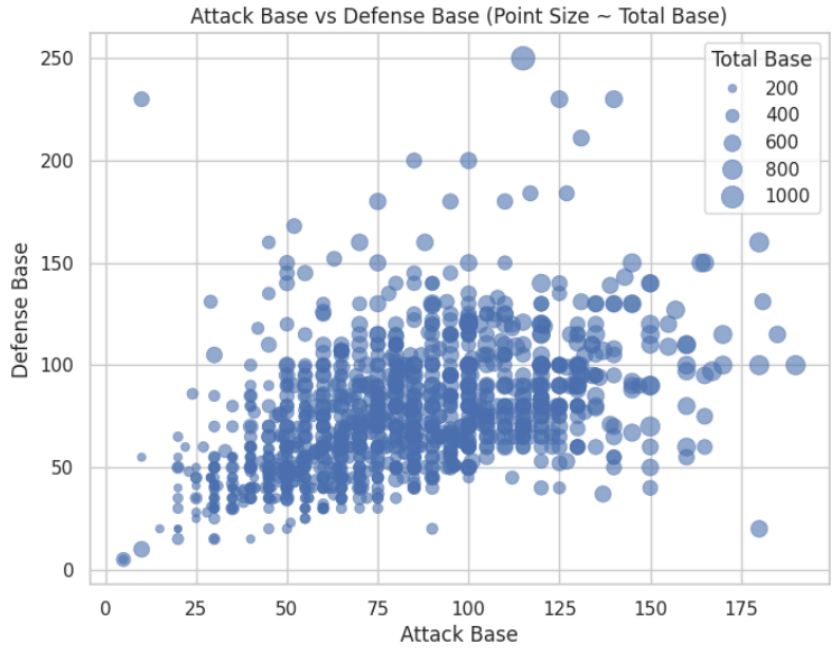
```python
plt.figure(figsize=(8,5))
sns.histplot(df['Attack Base'], bins=30, kde=True)
plt.title("Distribution of Attack Base")
plt.xlabel("Attack Base")
plt.ylabel("Frequency")
# plt.savefig("hist_attack_base.png", dpi=300, bbox_inches='tight')
plt.show()
```



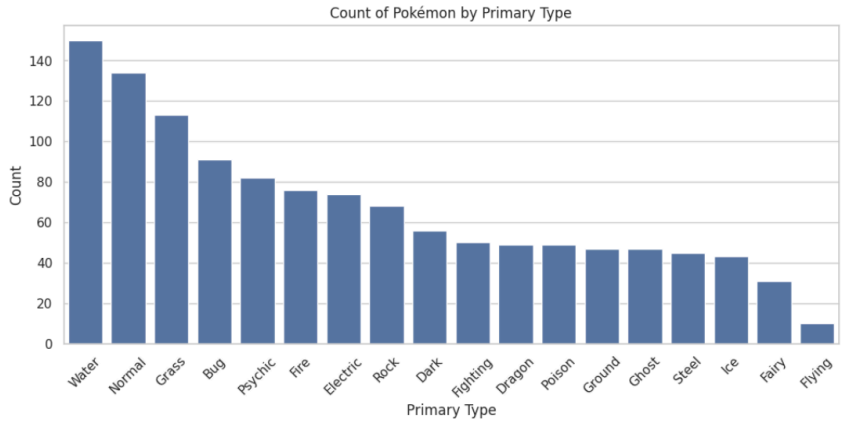

```python
top_types = df['Type 1'].value_counts().head(10).index
subset = df[df['Type 1'].isin(top_types)]

plt.figure(figsize=(12,5))
sns.boxplot(data=subset, x='Type 1', y='HP Base')
plt.title("HP Base by Primary Type (Top 10 Types)")
plt.xlabel("Primary Type")
plt.ylabel("HP Base")
plt.xticks(rotation=45)
# plt.savefig("box_hp_by_type.png", dpi=300, bbox_inches='tight')
plt.show()
```



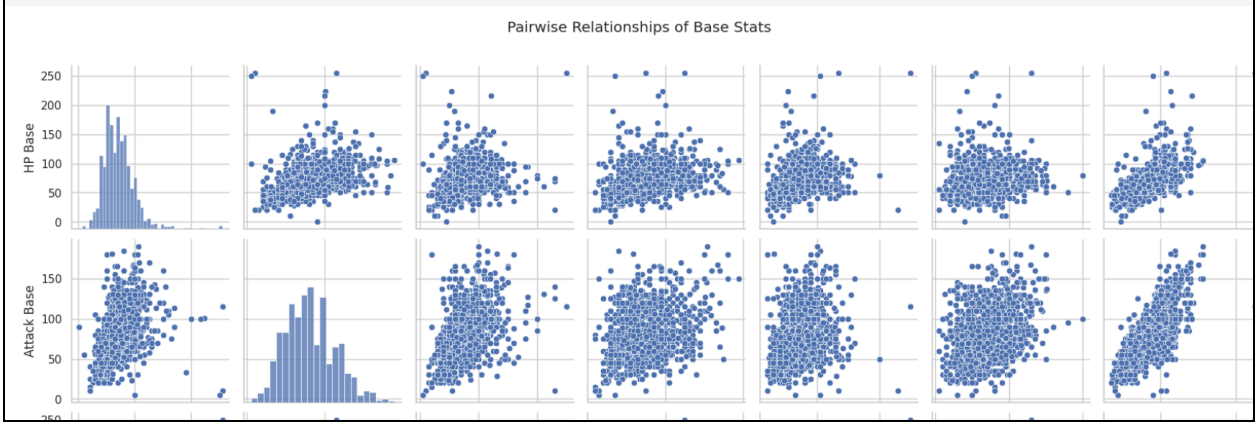
```
plt.figure(figsize=(8,6))
sns.scatterplot(
    data=df, x='Attack Base', y='Defense Base',
    size='Total Base', sizes=(20, 200), alpha=0.6, edgecolor=None
)
plt.title("Attack Base vs Defense Base (Point Size ~ Total Base)")
plt.xlabel("Attack Base")
plt.ylabel("Defense Base")
# plt.savefig("scatter_atk_def.png", dpi=300, bbox_inches='tight')
plt.show()
```




```
type_counts = df['Type 1'].value_counts()

plt.figure(figsize=(12,5))
sns.barplot(x=type_counts.index, y=type_counts.values)
plt.title("Count of Pokémon by Primary Type")
plt.xlabel("Primary Type")
plt.ylabel("Count")
plt.xticks(rotation=45)
# plt.savefig("bar_type_counts.png", dpi=300, bbox_inches='tight')
plt.show()
```



```
sns.pairplot(df[base_stats + ['Total Base']])
plt.suptitle("Pairwise Relationships of Base Stats", y=1.02)
plt.show()
```



Conclusion

EDA provides critical insights into the Pokémon dataset, helping us understand key patterns:

- Some Pokémon types (like Water, Normal, and Grass) are much more frequent than others.
- Legendary Pokémon consistently show higher stats, especially in HP, Attack, and Total.
- Attack values are widely dispersed, showing a few extremely strong Pokémon compared to the average.
- Speed has weak correlation with Defense, indicating that being fast does not guarantee tankiness.
- Outliers exist across attributes, representing unique and powerful Pokémon.

Through descriptive statistics, correlation analysis, and visualizations, we gain a clearer picture of how Pokémon differ in their abilities and distributions across types and generations.