pokemon_analysis

November 15, 2023

1 Pokemon Data Analysis

1.1 Introduction

The undertaken project focused on developing a Pokémon database system, commonly referred to as a Pokédex, by employing Python and PostgreSQL. The primary goal was to compile and store comprehensive information about diverse Pokémon species, sourced from https://pokemondb.net/, and present this data in a structured database. This initiative served as a personal project, aiming to highlight my competencies in SQL, Python, and Data Analysis.

Python was selected as the primary programming language for project implementation. The utilization of libraries such as Beautiful Soup and Requests facilitated web scraping of Pokémon data from the designated website, enabling the extraction of details like Pokémon names, types, abilities, and stats.

The integration of Pandas, a potent data manipulation library in Python, played a pivotal role in refining and processing the scraped data. It provided the means to convert raw web data into a structured format suitable for insertion into the database.

For the creation and management of the Pokémon database, PostgreSQL, a robust relational database management system, was employed. SQL was instrumental in defining the schema, establishing tables, and inserting data, ensuring that the information was well-organized and easily retrievable through queries.

To summarize, this project served as a demonstration of my proficiency in Python, expertise in data manipulation using Pandas, and competence in SQL-based database management. It represented a valuable opportunity to enhance skills in web scraping, data cleaning, and database design, underscoring the developer's capabilities in these critical domains.

1.1.1 Tasks of the project:

1. Which type has more pokemon?
2. Which region has introduced more pokemon?
3. Top 10 strongest and weakest non legenday pokemon?(Excludes mega evolve)
4. Top 10 weakest and strongest pokemon from each type? (Excludes mega evolve and legendary pokemons)
5. Top 5 strongest legendary pokemon? (Excludes mega evolve)

1.2 Processing the data

[ ]:
```
# Import Required libraries

import pandas as pd
import psycopg2
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns
```

Connect to the database

[ ]:
```
# Define the connection parameters
conn = psycopg2.connect(
    host="satao.db.elephantsql.com",
    database="kcvwfryw",
    user="kcvwfryw",
    password="WqfcEZ66jLgOVgacQ6IJYTtELlfkSaS9")

cursor = conn.cursor()
```

Inspecting the database

[ ]:
```
query = """select * from pokemon;"""
```

[ ]:
```
cursor.execute(query)

results = []

for i, data in enumerate(cursor):
    results.append(data)
```

[ ]:
```
data_table = pd.DataFrame(results, columns= ['Poke ID','Name','Type 1', 'Type␣
↪2','Hp','Attack','Defense',
                                              'Special Attack','Special␣
↪Defense','Speed', 'Legendary','Region ID','Game ID', 'Mega Evlove'])
data_table.head(20)
```

[ ]:

	Poke ID	Name	Type 1	Type 2	Hp	Attack	\
0	771	Thundurus Therian Forme	Electric	Flying	79	105	
1	248	Yanma	Bug	Flying	65	65	
2	164	Scyther	Bug	Flying	70	110	
3	1025	Obstagoon	Dark	Normal	93	90	
4	849	Pumpkaboo Average Size	Ghost	Grass	49	66	
5	850	Pumpkaboo Small Size	Ghost	Grass	44	66	
6	851	Pumpkaboo Large Size	Ghost	Grass	54	66	
7	852	Pumpkaboo Super Size	Ghost	Grass	59	66	
8	78	Growlithe Hisuian Growlithe	Fire	Rock	60	75	

9	80	Arcanine Hisuian Arcanine	Fire	Rock	95	115	
10	256	Slowking Galarian Slowking	Poison	Psychic	95	65	
11	264	Gligar	Ground	Flying	65	75	
12	254	Murkrow	Dark	Flying	60	85	
13	265	Steelix	Steel	Ground	75	85	
14	83	Poliwrath	Water	Fighting	90	95	
15	255	Slowking	Water	Psychic	95	75	
16	267	Snubbull	Fairy	None	60	80	
17	268	Granbull	Fairy	None	90	120	
18	853	Gourgeist Average Size	Ghost	Grass	65	90	
19	854	Gourgeist Small Size	Ghost	Grass	55	85	

	Defense	Special Attack	Special Defense	Speed	Legendary	Region ID	\
0	70	145	80	101	False	5	
1	45	75	45	95	False	2	
2	80	55	80	105	False	1	
3	101	60	81	95	False	8	
4	70	44	55	51	False	6	
5	70	44	55	56	False	6	
6	70	44	55	46	False	6	
7	70	44	55	41	False	6	
8	45	65	50	55	False	1	
9	80	95	80	90	False	1	
10	80	110	110	30	False	2	
11	105	35	65	85	False	2	
12	42	85	42	91	False	2	
13	200	55	65	30	False	2	
14	95	70	90	70	False	1	
15	80	100	110	30	False	2	
16	50	40	40	30	False	2	
17	75	60	60	45	False	2	
18	122	58	75	84	False	6	
19	122	58	75	99	False	6	

	Game ID	Mega Evlove
0	11	False
1	3	False
2	1	False
3	18	False
4	13	False
5	13	False
6	13	False
7	13	False
8	1	False
9	1	False
10	3	False
11	3	False

```
12        3        False
13        3        False
14        1        False
15        3        False
16        3        False
17        3        False
18       13        False
19       13        False
```

[ ]:
```
df_pulled = pd.DataFrame(results)
df_pulled.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1195 entries, 0 to 1194
Data columns (total 14 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       1195 non-null   int64
 1   1       1195 non-null   object
 2   2       1195 non-null   object
 3   3       653 non-null    object
 4   4       1195 non-null   int64
 5   5       1195 non-null   int64
 6   6       1195 non-null   int64
 7   7       1195 non-null   int64
 8   8       1195 non-null   int64
 9   9       1195 non-null   int64
 10  10      1195 non-null   bool
 11  11      1195 non-null   int64
 12  12      1195 non-null   int64
 13  13      1195 non-null   bool
dtypes: bool(2), int64(9), object(3)
memory usage: 114.5+ KB
```

[ ]:
```
df_pulled.shape
```

[ ]:
```
(1195, 14)
```

[ ]:
```
df_pulled.describe()
```

[ ]:
```
0     8
4     8
5     8
6     8
7     8
8     8
9     8
```

```
11    8
12    8
dtype: int64
```

1.3 Exploratory Data Analysis

1.Which type has more pokemon?

[ ]:
```
# Define the query
query = """select type1, count(type1)  as count from pokemon group by type1␣
 ↪order by  count desc;"""
cursor.execute(query)

results = []

for i, data in enumerate(cursor):
    results.append(data)
```

[ ]:
```
df = pd.DataFrame(results, columns= ['Type','Number of Pokemon'])
df
```

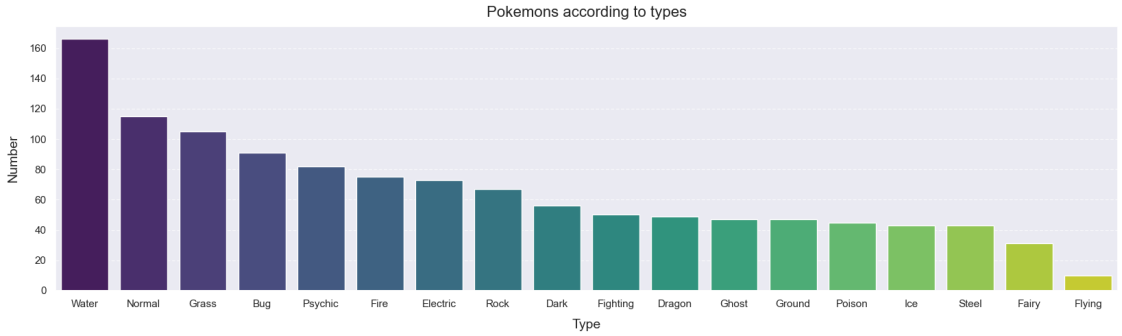
[ ]:

	Type	Number of Pokemon
0	Water	166
1	Normal	115
2	Grass	105
3	Bug	91
4	Psychic	82
5	Fire	75
6	Electric	73
7	Rock	67
8	Dark	56
9	Fighting	50
10	Dragon	49
11	Ghost	47
12	Ground	47
13	Poison	45
14	Ice	43
15	Steel	43
16	Fairy	31
17	Flying	10

Create a barchart

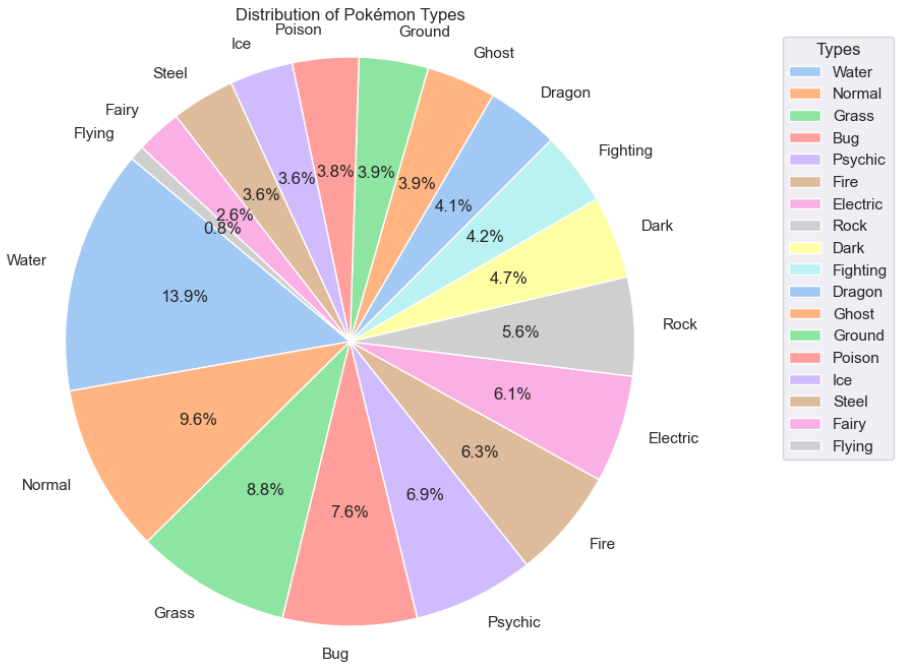
[ ]:
```
# Assuming df is your DataFrame
sns.set(rc={'figure.figsize': (20, 5)})
ax = sns.barplot(x=df.iloc[:, 0], y=df.iloc[:, 1], palette='viridis')

# Adding x-label and y-label
```

```
plt.xlabel('Type', fontsize=14, labelpad=10)
plt.ylabel('Number', fontsize=14, labelpad=10)

# Adding title
plt.title('Pokemons according to types', fontsize=16, pad=10)

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

# Show plot
plt.show()
```



Make a Pie-chart

[ ]:
```
plt.figure(figsize=(14, 8))

labels = df.iloc[:, 0]
sizes = df.iloc[:, 1]
colors = sns.color_palette('pastel')[0:len(labels)]

plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=colors, startangle=140)
plt.title('Distribution of Pokémon Types')
plt.legend(labels, title='Types', loc='upper right', bbox_to_anchor=(1, 1))
plt.axis('equal')
plt.show()
```



The provided charts clearly illustrate a significant prevalence of Water-type Pokémon, making up the largest category at 13.9% of all species. In contrast, Flying-type Pokémon are notably scarce, comprising just 0.8% of the total. This data highlights the diversity within the Pokémon universe, with aquatic creatures being highly represented, while avian creatures are relatively rare. The distribution of types adds depth and variety to the Pokémon world, offering a wide range of strengths, weaknesses, and strategic opportunities for trainers.

2. Top 10 Water-type pokemon

[ ]:
```
# Define the query
query = """select name, SUM(hp + attack + defense + sp_attack + sp_def + speed)␣
 ↪as CP from pokemon where
type1 = 'Water'and mega_evolve = false
and legendary = false group by name ORDER BY CP desc limit 10;"""
cursor.execute(query)

results = []

for i, data in enumerate(cursor):
    results.append(data)
```

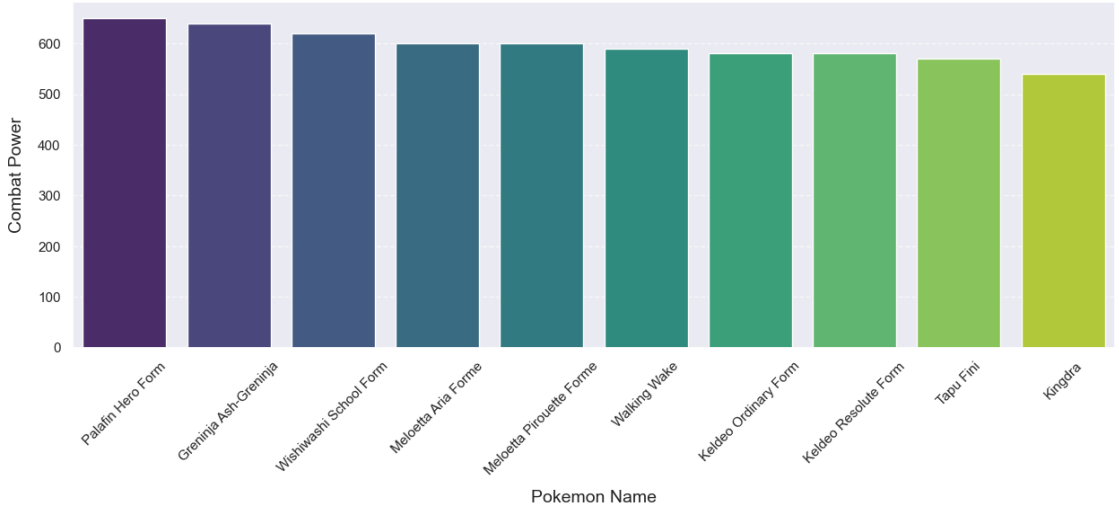
[ ]:
```
df = pd.DataFrame(results, columns= ['Name','Combat Power'])
df
```

[ ]:

	Name	Combat Power
0	Palafin Hero Form	650

```
1     Greninja Ash-Greninja          640
2    Wishiwashi School Form          620
3       Meloetta Aria Forme          600
4  Meloetta Pirouette Forme          600
5              Walking Wake          590
6      Keldeo Ordinary Form          580
7      Keldeo Resolute Form          580
8                 Tapu Fini          570
9                   Kingdra          540
```

[ ]:
```
# Assuming df is your DataFrame
sns.set(rc={'figure.figsize': (15, 5)})
ax = sns.barplot(x=df.iloc[:, 0], y=df.iloc[:, 1], palette='viridis')
plt.xlabel('Pokemon Name', fontsize=14, labelpad=10)
plt.ylabel('Combat Power', fontsize=14, labelpad=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.xticks(rotation=45)
plt.show()
```



3. Top 10 weakest Pokemon

[ ]:
```
# Define the query
query = """select name as Name, type1 as Type, SUM(hp + attack + defense +␣
 ↪sp_attack + sp_def + speed) as CP from pokemon where
mega_evolve = false
and legendary = false group by name, type1 ORDER BY CP asc limit 10;"""
cursor.execute(query)

results = []
```

```
for i, data in enumerate(cursor):
    results.append(data)
```

[ ]:
```
df = pd.DataFrame(results, columns= ['Name','Type','Combat Power'])
df
```

[ ]:

	Name	Type	Combat Power
0	Wishiwashi Solo Form	Water	175
1	Blipbug	Bug	180
2	Sunkern	Grass	180
3	Snom	Ice	185
4	Azurill	Normal	190
5	Kricketot	Bug	194
6	Wurmple	Bug	195
7	Weedle	Bug	195
8	Caterpie	Bug	195
9	Ralts	Psychic	198

4. Pokemon Introducted by Regions

[ ]:
```
# Define the query
query = """select region_name, count(region_name)  from pokemon left join␣
 ↪region on region_id = region.id group by region_name; """
cursor.execute(query)

results = []

for i, data in enumerate(cursor):
    results.append(data)
```

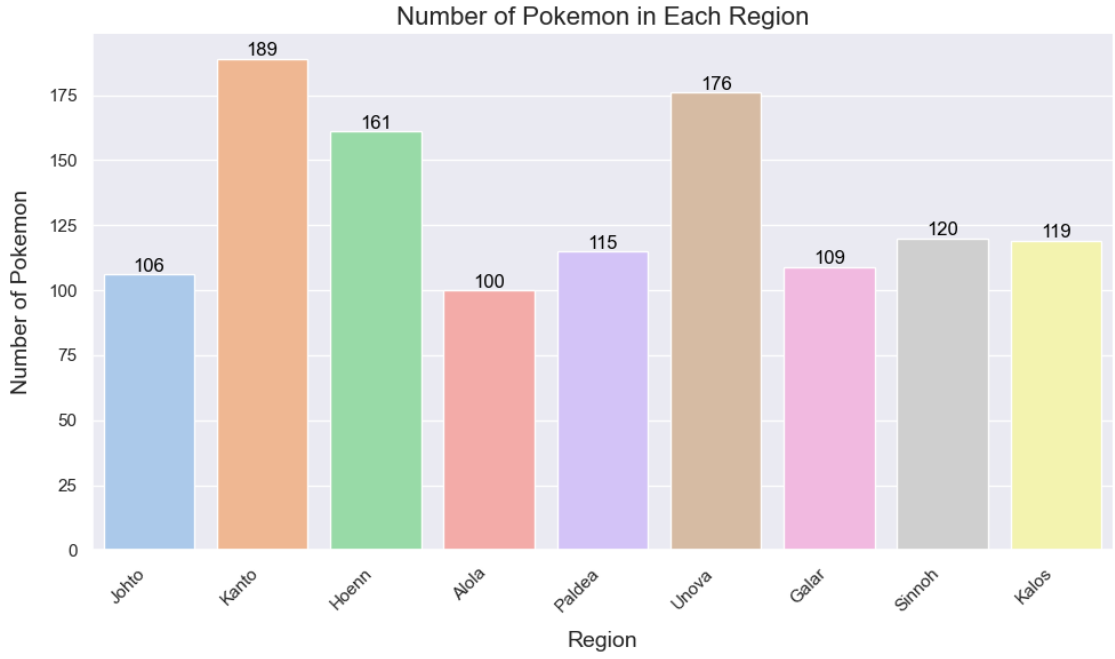
[ ]:
```
df = pd.DataFrame(results, columns= ['Region','Number of Pokemon'])
df
```

[ ]:

	Region	Number of Pokemon
0	Johto	106
1	Kanto	189
2	Hoenn	161
3	Alola	100
4	Paldea	115
5	Unova	176
6	Galar	109
7	Sinnoh	120
8	Kalos	119

[ ]:
```
sns.set_palette("pastel")

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

```
ax = sns.barplot(data=df, x='Region', y='Number of Pokemon')

plt.xlabel('Region', fontsize=14, labelpad=10)
plt.ylabel('Number of Pokemon', fontsize=14, labelpad=10)
plt.title('Number of Pokemon in Each Region', fontsize=16)

ax.set_xticklabels(ax.get_xticklabels(), rotation=45,␣
 ↪horizontalalignment='right')

for p in ax.patches:
    ax.annotate(f'{p.get_height():.0f}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()),
                ha='center', va='center', fontsize=12, color='black',␣
 ↪xytext=(0, 5),
                textcoords='offset points')

ax.get_yaxis().set_major_formatter(plt.FuncFormatter(lambda x, loc: "{:,}".
 ↪format(int(x))))

sns.despine()

plt.tight_layout()
plt.show()
```



The chart provides data on the number of Pokemon in various regions within the Pokemon universe. This data is essential for understanding the distribution of Pokemon species across different in-game

regions, and it holds significance for both game enthusiasts and researchers studying the Pokemon franchise.

The chart encompasses nine different regions, with Kanto having the highest number of Pokemon species at 189, making it one of the most iconic and well-established regions in the series. Following closely are Unova with 176 species and Hoenn with 161, highlighting the diversity of Pokemon across different generations of games. On the other end of the spectrum, we find Alola with 100 species and Galar with 109 species, representing regions from the more recent Pokemon games. The chart illustrates how the number of Pokemon can vary significantly from one region to another, emphasizing the importance of each region's unique characteristics and ecosystems in shaping the diversity of Pokemon species.

In summary, this chart provides a concise overview of the number of Pokemon in various regions, shedding light on the franchise's evolving world-building and game development over the years. It showcases the rich variety of Pokemon across different regions, which has been a key factor in the enduring popularity and appeal of the Pokemon series among fans and players worldwide.

5. Top 5 strongest legendary Pokemon

[ ]:
```
# Define the query
query = """select name, SUM(hp + attack + defense + sp_attack + sp_def + speed)␣
 ↪as CP,
type1 from pokemon where mega_evolve = false and legendary = true group by␣
 ↪name, type1 order by CP desc limit 5; """
cursor.execute(query)

results = []

for i, data in enumerate(cursor):
    results.append(data)

conn.close()
cursor.close()
```

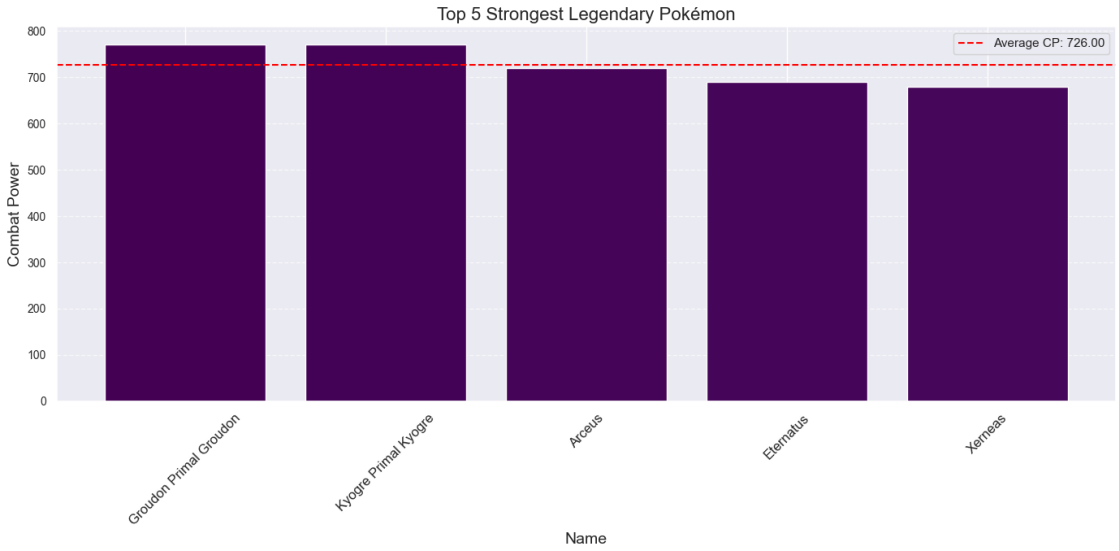
[ ]:
```
df = pd.DataFrame(results, columns= ['Name','Combat Power','Type'])
df
```

[ ]:

	Name	Combat Power	Type
0	Groudon Primal Groudon	770	Ground
1	Kyogre Primal Kyogre	770	Water
2	Arceus	720	Normal
3	Eternatus	690	Poison
4	Xerneas	680	Fairy

[ ]:
```
top5_df = df.sort_values(by='Combat Power', ascending=False).head(5)

colors = plt.cm.viridis(range(len(top5_df)))

plt.figure(figsize=(14, 7))
```

```
# Create the bar chart with customized aesthetics
plt.bar(top5_df['Name'], top5_df['Combat Power'], color=colors)
plt.xlabel('Name', fontsize=14)
plt.ylabel('Combat Power', fontsize=14)
plt.title('Top 5 Strongest Legendary Pokémon', fontsize=16)
plt.xticks(rotation=45, fontsize=12)  # Rotate x-axis labels and adjust font size
plt.yticks(fontsize=10)  # Adjust y-axis labels font size
plt.grid(axis='y', linestyle='--', alpha=0.7)
average_cp = top5_df['Combat Power'].mean()
plt.axhline(y=average_cp, color='red', linestyle='--', label=f'Average CP: {average_cp:.2f}')
plt.legend()
plt.tight_layout()
plt.show()
```



The chart displays a list of Legendary Pokémon along with their names, Combat Power (CP), and respective types. The CP is a calculated value derived from the sum of the Pokémon's individual stats, such as HP, attack, defense, special attack, special defense, and speed. Among the Pokémon listed, Groudon Primal Groudon and Kyogre Primal Kyogre share the highest CP of 770, both representing the Ground and Water types, respectively. Arceus follows closely with a CP of 720, classified as a Normal type. Eternatus and Xerneas have CP values of 690 and 680, respectively, and are associated with the Poison and Fairy types. These Pokémon are notable for their legendary status and formidable CP, making them highly sought after and powerful additions to any Pokémon collection.

1.4 Conclusion

This project has not only provided me with a robust database for Pokémon enthusiasts but has also demonstrated my proficiency in web scraping, data cleaning, database design, and data analysis. It has enriched my understanding of Pokémon statistics, types, and characteristics, offering me a valuable resource for my personal interest in the Pokémon world. Additionally, it showcases my skills in Python, Pandas, and SQL, which are highly transferable to other data-centric projects. This project stands as a testament to the power of programming and data management in bringing order and insight to complex datasets, and it has been a fulfilling journey of learning and exploration. make markdown