Data Wizards : Comprehensive analysis of Pokemon data

Mounika Jakkula, Manasa Akula, Gayathri Nangineni, PoojaSri Ramineni

Northwest Missouri State University, Maryville MO 64468, USA
S560445@nwmissouri.edu,
S560998@nwmissouri.edu,s559973@nwmissouri.edu,s559300@nwmissouri.edu

1 44517-02

2 Team name: Data Wizards

3 Team Members

- Mounika Jakkula - S560445
- Manasa Akula - S560998
- Gayathri Nangineni - S559973
- PoojaSri Ramineni - S559300

4 Project Title

Comprehensive analysis of Pokemon data

5 Project Idea

The project idea is to undertake an in-depth analysis of Pokemon data to acquire insight into numerous elements of Pokemon traits and gaming dynamics. This analysis may include studying evolutionary patterns, examining basic stats, investigating type variety, assessing ability impact, doing sentiment analysis on descriptions, investigating physical qualities, performing ranking analysis, analyzing generation trends, and so on. The purpose is to discover patterns, trends, and linkages in the Pokemon universe in order to better comprehend its evolution over time and the impact it has on gaming dynamics.

6 Technology Summary

Data Cleaning: Python (Pandas, NumPy)
Kafka: Integration with kafka for continuous ingestion and analysis of real time data which ensures scalability, fault tolerance, reliability.
PySpark: For handling large-scale data analysis. PySpark provides the ability to work with big data efficiently using the Spark framework.
matplotlib: To visualize the results in a pictorial form.

7 Architecture Diagram

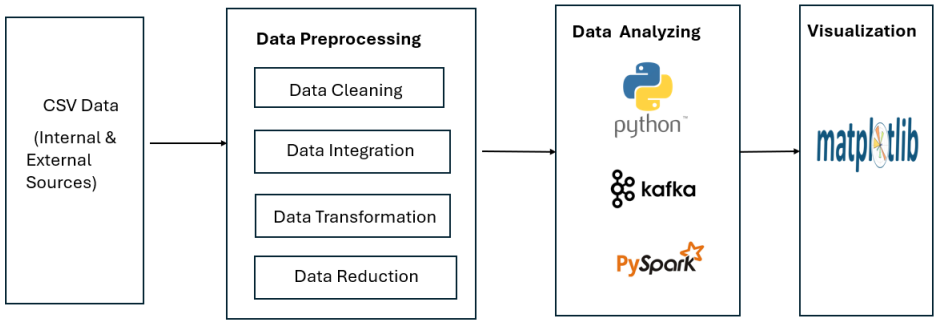


Fig. 1. Architecture Diagram

8 Architecture Summary

Architecture designed for managing various data sources, including CSV files.
Stages: data cleaning, integration, transformation, and reduction.
Data Cleaning: Identify and rectify inconsistencies, errors, and missing values.
Data Integration: Dataset is consumed by using kafka
Data Transformation: Standardize formats, apply calculations, and create new features.
Data Reduction: Minimize dataset size without sacrificing analytical value.
Framework implemented in PySpark and Python for scalability and flexibility with large datasets.
PySpark's distributed computing capabilities ensure efficient processing.
Python allows for versatile manipulation tasks Visualizing processed data through Matplotlib for intuitive exploration and presentation.
Interconnected workflow ensures smooth data flow from ingestion to visualization.
Benefits: Improved data integrity, reduced processing time, enhanced decision-making capabilities.

9 Project Goals

The project goals encompass a range of analytical tasks aimed at understanding different facets of the Pokemon universe using data analysis techniques.

Goal 1: Conducting a statistical analysis of the base stats (hp, atk, def, spatk, spdef, speed, total) to identify the average, median, and variation of stats among different Pokemon generations or types, providing insights into the overall power balance in the Pokemon world.
Goal 2: Examining the physical attributes of Pokemon, such as height and weight, to understand the diversity and distribution of sizes among different species, and investigate whether there are any correlations between physical attributes and other characteristics like base stats or evolutionary stages.
Goal 3: Analyzing Pokemon rankings to understand the factors influencing a Pokemon's rank, such as its base stats, evolutionary stage, or abilities, and identify any patterns or trends in how Pokemon are ranked across different generations or types.
Goal 4: Analyzing trends in Pokemon characteristics (such as base stats, types, and abilities) across different generations to understand how the Pokemon universe has evolved over time and identify any significant changes or developments in gameplay mechanics.
Goal 5: Segmenting Pokemon based on weight classes (e.g., lightweight, middleweight, heavyweight) to analyze how weight influences factors like base stats, abilities, and evolutionary patterns, and identify any correlations between weight class and competitive performance.
Goal 6: Investigating the descendants of Pokemon species by tracing the evolutionary paths of Pokemon mentioned in the evolves from column, identifying any evolutionary trends or patterns that emerge across generations and types of Pokemon.
Goal 7: Exploring the characteristics and attributes of legendary Pokemon, including their base stats, abilities, and rarity, and analyze how they differ from non-legendary Pokemon in terms of gameplay mechanics and competitive viability.
Group 8: Analyzing Pokemon rankings to understand the factors influencing a Pokemon's rank, such as its base stats, evolutionary stage, or abilities, and identify any patterns or trends in how Pokemon are ranked across different generations or types.

10 Project Description

This project aims to comprehensively analyze various facets of the Pokemon universe through data analysis techniques and tools such as Python, Pandas, NumPy, Kafka, PySpark and Matplotlib. Focusing on main objectives.
Firstly, we will conduct statistical analyses of base stats to discern patterns among different generations or types, shedding light on the power balance in the Pokemon world.
Secondly, we'll delve into the physical attributes of Pokemon to understand size diversity and investigate correlations with other characteristics like base stats or evolutionary stages.
Thirdly, we'll analyze Pokemon rankings to identify factors influencing rank and

discern patterns across generations or types.
Additionally, we'll examine trends in Pokemon characteristics across generations, aiming to understand the evolution of the Pokemon universe's gameplay mechanics.
Furthermore, we'll segment Pokemon based on weight classes to analyze correlations with performance.
Moreover, we'll trace evolutionary paths to identify trends among descendants. finally, explore the attributes of legendary Pokemon and their impact on gameplay and competitive viability. Through these endeavors, we seek to deepen our understanding of the Pokemon universe's dynamics and evolution.

11 Implementation steps

11.1 Processing of data for the project

- **Importing Modules and Loading Data:**
  - We begin by importing the necessary Python modules, including Pandas and NumPy. Then, we load the dataset from the CSV file 'pokemons2.csv' into a Pandas DataFrame using the pd.read-csv() function. We also display the first few rows of the DataFrame using the df.head() method to inspect the data structure.
- **Datatypes of the Variables**
  - Next, we check the data types of each variable in the DataFrame using the df.info() method. This provides information about the number of non-null values and the data type of each column.
- **Missing Data:**
  - In this step, we identify and quantify missing data in the DataFrame using the df.isnull().sum().sort-values(ascending=False) method. This allows us to see which columns have missing values and how many missing values are present in each column.
- **Replacing with Mean Values:**
  - We address missing data by replacing them with the mean values of the respective columns. We identify columns with missing values and replace them with the most frequent value using a loop. After replacing missing values in the 'height' and 'type2' columns, we display the cleaned DataFrame.
- **Dropping of Columns:**
  - Here, we drop the 'def' and 'weight' columns from the DataFrame using the df.drop(columns=['def','weight'], inplace=True) method. This is done to remove unnecessary columns from the dataset.
- **Columns after Dropping Some of Them:**
  - We verify the columns present in the DataFrame after dropping some of them using the df.columns attribute. This provides a list of column names currently present in the DataFrame.
- **Checking Null Values and Column Information After Dropping Some Columns:**

  - After dropping some columns, we again check for null values in the DataFrame using the df.isna().sum() method. This helps ensure that no null values are present after dropping the specified columns.
- **Adjust DataFrame Structure:**
  - Verify the changes by displaying the first few rows of the updated DataFrame using the `head()` method.
- **Output DataFrame Shape:**
  - Display the shape of the DataFrame, which shows the number of rows and columns, using the `shape` attribute. This provides a summary of the DataFrame's dimensions after cleaning.
- **Saving Cleaned Dataset:**
  - Finally, we save the cleaned DataFrame to a new CSV file named 'output.csv' using the df.to csv('output.csv', index=False) method. This allows us to store the cleaned data for further analysis or future use.

11.2 Goal-1 implementation steps:

- **Installing Required Packages:**
  - The first step involves installing the necessary Python package, kafka-python, using the pip install command. This package enables communication with Kafka for data streaming capabilities.
- **Initializing Kafka Producer:**
  - In this step, a Kafka producer is initialized with the specified bootstrap servers. The producer is responsible for sending data from a local CSV file to a Kafka topic named 'pokemondata'. Each row of the CSV file is encoded and sent to the Kafka topic.
- **Initializing Spark Session:**
  - A Spark session is created using the SparkSession.builder method. This session is named "PokemonAnalysis" and is essential for executing Spark code and managing resources effectively within the Spark framework.
- **Defining Schema and Loading Data:**
  - The schema for the Pokemon dataset is defined, specifying the data types for each column. This schema ensures proper interpretation of the dataset during loading. The spark.read.csv method is then used to load the data from the CSV file into a Spark DataFrame named df.
- **Perform Grouped Statistics:**
  - In this step, we're performing grouped statistics on the DataFrame to analyze the base stats (hp, atk, def, spatk, spdef, speed, total) of different Pokemon generations. We're calculating the mean, standard deviation, and median for each stat across different generations.
- **Display Grouped Statistics:**
  - Finally, we're displaying the grouped statistics DataFrame, showing the average, standard deviation, and median values for each base stat across different Pokemon generations.

    Each step in the code corresponds to a specific action in achieving the goal of conducting a statistical analysis of Pokemon base stats. By following these steps, we can gather insights into the power balance among different Pokemon generations or types.

11.3 Goal-2 implementation steps:

- **Installing Required Packages:**
  - The first step involves installing the necessary Python package, kafka-python, using the pip install command. This package enables communication with Kafka for data streaming capabilities.
- **Initializing Kafka Producer:**
  - In this step, a Kafka producer is initialized with the specified bootstrap servers. The producer is responsible for sending data from a local CSV file to a Kafka topic named 'pokemondata'. Each row of the CSV file is encoded and sent to the Kafka topic.
- **Initializing Spark Session:**
  - A Spark session is created using the SparkSession.builder method. This session is named "PokemonAnalysis" and is essential for executing Spark code and managing resources effectively within the Spark framework.
- **Defining Schema and Loading Data:**
  - The schema for the Pokemon dataset is defined, specifying the data types for each column. This schema ensures proper interpretation of the dataset during loading. The spark.read.csv method is then used to load the data from the CSV file into a Spark DataFrame named df.
- **Handling Missing Values:**
  - In this step, any rows with missing values in the "height" and "weight" columns are dropped from the DataFrame using the dropna method. This ensures the integrity of the analysis by removing incomplete or unreliable data.
- **Calculating Correlations:**
  - Correlation coefficients between height, weight, and other characteristics such as base stats are calculated using PySpark's statistical functions. This analysis aims to identify any relationships or dependencies between physical attributes and other features of Pokemon species.
- **Displaying Results:**
  - The calculated correlation coefficients are printed to the console, providing insights into the relationships between physical attributes like height and weight, and other characteristics such as base stats. This step aids in understanding the diversity and distribution of sizes among different Pokemon species and investigates potential correlations with other attributes.

11.4 Goal-3 implementation steps:

- **Installing Required Packages:**
  - The first step involves installing the necessary Python package, kafka-python, using the pip install command. This package enables communication with Kafka for data streaming capabilities.
- **Initializing Kafka Producer:**

- In this step, a Kafka producer is initialized with the specified bootstrap servers. The producer is responsible for sending data from a local CSV file to a Kafka topic named 'pokemondata'. Each row of the CSV file is encoded and sent to the Kafka topic.

– **Initializing Spark Session:**
  - A Spark session is created using the SparkSession.builder method. This session is named "PokemonAnalysis" and is essential for executing Spark code and managing resources effectively within the Spark framework.
– **Defining Schema and Loading Data:**
  - The schema for the Pokemon dataset is defined, specifying the data types for each column. This schema ensures proper interpretation of the dataset during loading. The spark.read.csv method is then used to load the data from the CSV file into a Spark DataFrame named df.
– **Performing Ranking Analysis:**
  - In this step, the DataFrame df is grouped by the "rank" column, and various statistical metrics such as average base stats (hp, atk, def, spatk, spdef, speed, total) are calculated for each rank category using PySpark's mean function. The resulting DataFrame, rankstats, provides insights into the average characteristics of Pokemon within each rank category.
– **Ordering by Rank and Displaying Results:**
  - The rankstats DataFrame is ordered by the "rank" column in descending order to facilitate analysis and identify patterns or trends across different ranks. Finally, the ranking analysis results are displayed, showing the average base stats for each rank category, providing insights into factors influencing a Pokemon's rank.

11.5 Goal-4 implementation steps:

– **Installing Required Packages:**
  - The first step involves installing the necessary Python package, kafka-python, using the pip install command. This package enables communication with Kafka for data streaming capabilities.
– **Initializing Kafka Producer:**
  - In this step, a Kafka producer is initialized with the specified bootstrap servers. The producer is responsible for sending data from a local CSV file to a Kafka topic named 'pokemondata'. Each row of the CSV file is encoded and sent to the Kafka topic.
– **Initializing Spark Session:**
  - A Spark session is created using the SparkSession.builder method. This session is named "PokemonAnalysis" and is essential for executing Spark code and managing resources effectively within the Spark framework.
– **Defining Schema and Loading Data:**
  - The schema for the Pokemon dataset is defined, specifying the data types for each column. This schema ensures proper interpretation of the dataset during loading. The spark.read.csv method is then used to load the data from the CSV file into a Spark DataFrame named df.

- **Data Manipulation - Explode and Split:**
  - In this step, the DataFrame df is manipulated to extract information from columns containing lists or arrays. The explode and split functions are used to split the "type1" and "abilities" columns into multiple rows, creating new rows for each type and ability associated with each Pokemon.
- **Grouping and Calculating Statistics:**
  - The DataFrame df is grouped by the "generation" column, and various statistical metrics such as average base stats (hp, atk, def, spatk, spdef, speed), count of types, and count of abilities are calculated for each generation using PySpark's aggregation functions (mean and count).
- **Displaying Generation Trend Analysis:**
  - Finally, the generation-wise statistics and trends are displayed in the generationstats DataFrame. The analysis provides insights into how Pokemon characteristics have evolved over different generations, including changes in base stats, types, and abilities, which helps in understanding the development of gameplay mechanics throughout the Pokemon universe.

11.6 Goal-5 implementation steps:

- **Installing Required Packages:**
  - The first step involves installing the necessary Python package, kafka-python, using the pip install command. This package enables communication with Kafka for data streaming capabilities.
- **Initializing Kafka Producer:**
  - In this step, a Kafka producer is initialized with the specified bootstrap servers. The producer is responsible for sending data from a local CSV file to a Kafka topic named 'pokemondata'. Each row of the CSV file is encoded and sent to the Kafka topic.
- **Initializing Spark Session:**
  - A Spark session is created using the SparkSession.builder method. This session is named "PokemonAnalysis" and is essential for executing Spark code and managing resources effectively within the Spark framework.
- **Defining Schema and Loading Data:**
  - The schema for the Pokemon dataset is defined, specifying the data types for each column. This schema ensures proper interpretation of the dataset during loading. The spark.read.csv method is then used to load the data from the CSV file into a Spark DataFrame named df.
- **Data Manipulation - Creating Weight Classes:**
  - In this step, a new column named "weightclass" is added to the DataFrame df. This column categorizes Pokemon into different weight classes based on their weight. The when and otherwise functions are used to define conditions for categorization.
- **Grouping and Analyzing Statistics:**

- The DataFrame df is grouped by the "weightclass" column, and various statistics such as average base stats (hp, atk, def, spatk, spdef, speed), count of abilities, and count of evolutionary patterns are calculated for each weight class. The groupBy and agg functions are employed for this analysis.

– **Displaying Weight Class Segmentation Analysis:**
  - Finally, the analysis results are displayed in the weightclassanalysis DataFrame. This analysis provides insights into how weight influences factors like base stats, abilities, and evolutionary patterns among Pokemon. The segmentation into different weight classes allows for a better understanding of correlations between weight class and competitive performance.

11.7 Goal-6 implementation steps:

– **Installing Required Packages:**
  - The first step involves installing the necessary Python package, kafka-python, using the pip install command. This package enables communication with Kafka for data streaming capabilities.

– **Initializing Kafka Producer:**
  - In this step, a Kafka producer is initialized with the specified bootstrap servers. The producer is responsible for sending data from a local CSV file to a Kafka topic named 'pokemondata'. Each row of the CSV file is encoded and sent to the Kafka topic.

– **Initializing Spark Session:**
  - A Spark session is created using the SparkSession.builder method. This session is named "PokemonAnalysis" and is essential for executing Spark code and managing resources effectively within the Spark framework.

– **Defining Schema and Loading Data:**
  - The schema for the Pokemon dataset is defined, specifying the data types for each column. This schema ensures proper interpretation of the dataset during loading. The spark.read.csv method is then used to load the data from the CSV file into a Spark DataFrame named df.

– **Data Manipulation - Grouping by Evolves From:**
  - In this step, the DataFrame df is grouped by the "evolvesfrom" column, which contains information about the Pokemon species from which each Pokemon evolves. The groupBy function is used for this grouping.

– **Aggregating Descendants:**
  - For each unique value in the "evolvesfrom" column, the collectlist function is applied to gather the names of all descendants (Pokemon species that evolve from the specified ancestor). The result is stored in a new column named "descendants". This analysis helps identify the evolutionary paths of different Pokemon species.

– **Displaying Descendants Analysis:**
  - Finally, the results of the descendants analysis are displayed in the DataFrame descendantsanalysis. This analysis provides insights into the evolutionary trends and patterns across generations and types of Pokemon. Each row represents an ancestor Pokemon along with its list of descendants.

11.8 Goal-7 implementation steps:

- **Installing Required Packages:**
  - The first step involves installing the necessary Python package, kafka-python, using the pip install command. This package enables communication with Kafka for data streaming capabilities.
- **Initializing Kafka Producer:**
  - In this step, a Kafka producer is initialized with the specified bootstrap servers. The producer is responsible for sending data from a local CSV file to a Kafka topic named 'pokemondata'. Each row of the CSV file is encoded and sent to the Kafka topic.
- **Initializing Spark Session:**
  - A Spark session is created using the SparkSession.builder method. This session is named "PokemonAnalysis" and is essential for executing Spark code and managing resources effectively within the Spark framework.
- **Defining Schema and Loading Data:**
  - The schema for the Pokemon dataset is defined, specifying the data types for each column. This schema ensures proper interpretation of the dataset during loading. The spark.read.csv method is then used to load the data from the CSV file into a Spark DataFrame named df.
- **Filtering Legendary and Non-Legendary Pokemon:**
  - Two separate DataFrames are created: one containing only legendary Pokemon (legendarydf) and the other containing non-legendary Pokemon (nonlegendarydf). This segregation is based on the "rank" column, where Pokemon with the rank "Legendary" are filtered into the legendary DataFrame.
- **Summary Statistics for Base Stats:**
  - Summary statistics are computed for the base stats (hp, atk, def, spatk, spdef, speed) of both legendary and non-legendary Pokémon. The summary function is applied to each DataFrame to calculate count, mean, standard deviation, min, max, and quartile values for each base stat.
- **Analyzing Abilities:**
  - The unique abilities of legendary and non-legendary Pokémon are extracted separately. The distinct function is used to obtain unique values from the "abilities" column for both legendary and non-legendary Pokémon. These unique abilities are then displayed for analysis.
- **Summary:**
  - In summary, the goal involves exploring the characteristics and attributes of legendary Pokémon compared to non-legendary ones. This analysis includes base stats summary statistics and unique abilities, shedding light on how legendary Pokémon differ from their non-legendary counterparts in terms of gameplay mechanics and competitive viability.

11.9 Goal-8 implementation steps:

- **Installing Required Packages:**
  - The first step involves installing the necessary Python package, kafka-python, using the pip install command. This package enables communication with Kafka for data streaming capabilities.
- **Initializing Kafka Producer:**
  - In this step, a Kafka producer is initialized with the specified bootstrap servers. The producer is responsible for sending data from a local CSV file to a Kafka topic named 'pokemondata'. Each row of the CSV file is encoded and sent to the Kafka topic.
- **Initializing Spark Session:**
  - A Spark session is created using the SparkSession.builder method. This session is named "PokemonAnalysis" and is essential for executing Spark code and managing resources effectively within the Spark framework.
- **Defining Schema and Loading Data:**
  - The schema for the Pokemon dataset is defined, specifying the data types for each column. This schema ensures proper interpretation of the dataset during loading. The spark.read.csv method is then used to load the data from the CSV file into a Spark DataFrame named df.
- **Ranking Analysis:**
- **Creating a Numeric Rank Column:**
  - A new column named "rank-numeric" is added to the DataFrame df, where the rank is represented as 1 for "Legendary" Pokémon and 0 for other ranks.
- **Ranking Analysis based on Base Stats:**
  - The DataFrame is grouped by the "rank" column, and the average values of base stats (hp, atk, def, spatk, spdef, speed) are calculated for each rank. The results are ordered by rank and displayed.
- **Ranking Analysis based on Evolutionary Stage:**
  - The DataFrame is grouped by the "rank" column, and the number of occurrences of each evolutionary stage (evolves-from) is counted for each rank. The results are displayed, with 0 values filled for Pokémon ranks that do not have evolutionary stages.
- **Ranking Analysis based on Abilities:**
  - The DataFrame is grouped by the "rank" column, and the count of unique abilities is calculated for each rank. The results are ordered by rank and displayed.
- **Goal Summary:**
  - The goal involves analyzing Pokémon rankings to understand the factors influencing a Pokémon's rank, such as base stats, evolutionary stage, or abilities. Patterns or trends in how Pokémon are ranked across different generations or types are identified through these analyses.

12 Results:

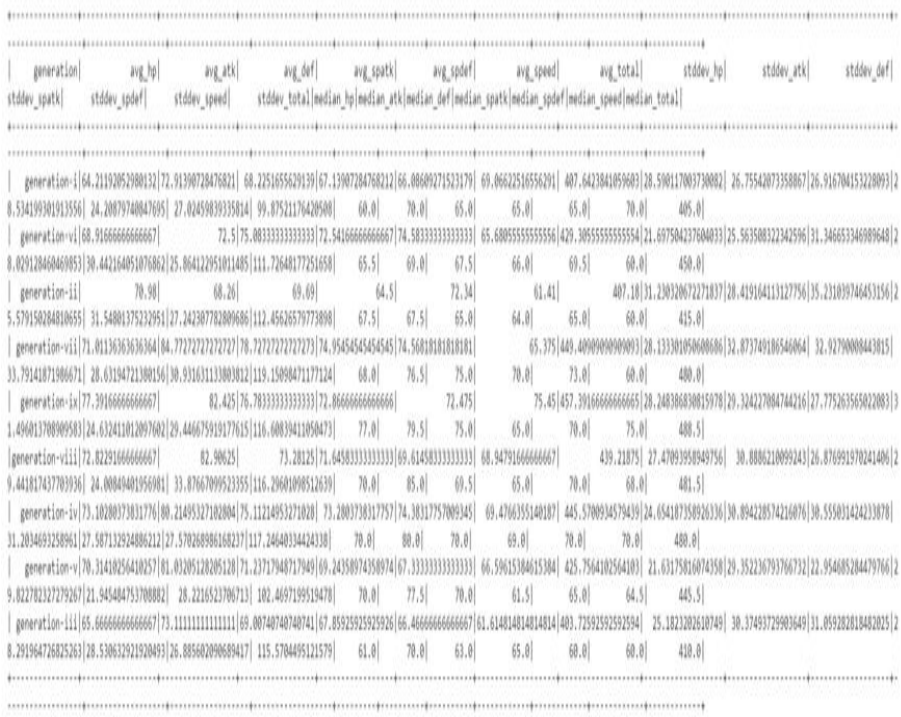
Goal 1: Conducting a statistical analysis of the base stats (hp, atk, def, spatk, spdef, speed, total) to identify the average, median, and variation of stats among different Pokemon generations or types, providing insights into the overall power balance in the Pokemon world.

12.1 Code Snippets

```
#Goal1: Perform grouped statistics
grouped_stats = df.groupBy("generation").agg(
    mean("hp").alias("avg_hp"),
    mean("atk").alias("avg_atk"),
    mean("def").alias("avg_def"),
    mean("spatk").alias("avg_spatk"),
    mean("spdef").alias("avg_spdef"),
    mean("speed").alias("avg_speed"),
    mean("total").alias("avg_total"),
    stddev("hp").alias("stddev_hp"),
    stddev("atk").alias("stddev_atk"),
    stddev("def").alias("stddev_def"),
    stddev("spatk").alias("stddev_spatk"),
    stddev("spdef").alias("stddev_spdef"),
    stddev("speed").alias("stddev_speed"),
    stddev("total").alias("stddev_total"),
    median("hp").alias("median_hp"),
    median("atk").alias("median_atk"),
    median("def").alias("median_def"),
    median("spatk").alias("median_spatk"),
    median("spdef").alias("median_spdef"),
    median("speed").alias("median_speed"),
    median("total").alias("median_total")
)

# Show the grouped statistics
grouped_stats.show()
```

Fig. 2. statistical analysis of the base stats



generation	avg_hp	avg_atk	avg_def	avg_spatk	avg_spdef	avg_speed	avg_total	stddev_hp	stddev_atk	stddev_def	stddev_spatk	stddev_spdef	stddev_speed	stddev_total	median_hp	median_atk	median_def	median_spatk	median_spdef	median_speed	median_total
generation-i	64.21193052980132	72.91390728476821	68.22516556291391	67.13907284768212	66.08609271523179	69.06622516556291	407.6423841059603	28.59011700373008	26.755420733588867	26.916704153228093	28.534199301913556	24.208797408476695	27.024598393358914	99.87521176420508	60.0	70.0	65.0	65.0	65.0	70.0	405.0
generation-vi	68.91666666666667	72.5	75.08333333333333	72.54166666666667	74.58333333333333	65.68055555555556	429.3055555555554	21.697504237684033	25.563508322342596	31.346653346989648	28.029128468469853	30.442164051076862	25.864122951011485	111.72648177251658	65.5	69.0	67.5	66.0	69.5	60.0	450.0
generation-ii	70.98	68.26	69.69	64.5	72.34	61.41	407.18	31.230320672271837	28.419164113127756	35.231039746453156	25.577508504810655	31.540013752329 51	27.242307782809686	112.45626579773898	67.5	67.5	65.0	64.0	65.0	60.0	415.0
generation-vii	71.01136363636364	84.77272727272727	78.72727272727273	74.95454545454545	74.56818181818181	65.375	449.4090909090993	28.133301050600686	32.87374918654606 4	32.92790008443015	33.79141871986671	28.631947213801 56	30.931631133803012	119.15098471177124	68.0	76.5	75.0	70.0	73.0	60.0	480.0
generation-ix	77.39166666666667	82.425	76.78333333333333	72.86666666666666	72.475	75.45	457.3916666666665	28.248386830815978	29.324227084744216	27.775263565022083	31.496013708909583	24.632411012097602	29.446675919177615	116.60839411950473	77.0	79.5	75.0	65.0	70.0	75.0	488.5
generation-viii	72.82291666666667	82.90625	73.28125	71.64583333333333	69.61458333333333	68.94791666666667	439.21875	27.47093958949756	30.888621099243	26.876991970241406	29.441817437703936	24.008494019569081	33.87667099523355	116.29601098512639	70.0	85.0	69.5	65.0	70.0	68.0	481.5
generation-iv	73.10280373831776	80.21495327102804	75.11214953271028	73.28037383177757	74.38317757009345	69.4766355140187	445.5700934579439	24.654187358926336	30.894228574216076	30.555031424233878	31.203469258961	27.587132924886212	27.570268986168237	117.24640334424338	70.0	80.0	70.0	69.0	70.0	70.0	480.0
generation-v	70.31410256410257	81.03205128205128	71.23717948717949	69.24358974358974	67.33333333333333	66.59615384615384	425.7564102564103	21.631758160743 58	29.352236793766732	22.954685284479766	29.822782327279267	21.945484753708882	28.22165237 06713	102.44971995 19478	70.0	77.5	70.0	61.5	65.0	64.5	445.5
generation-iii	65.66666666666667	73.11111111111111	69.00740740740741	67.85925925925926	66.46666666666667	61.61481481481481 4	403.7259259259594	25.18232026107 49	30.37493729903649	31.059282818482025	28.291964726825263	28.530632921920493	26.885602090689417	115.5704495121579	61.0	70.0	63.0	65.0	60.0	60.0	410.0

Fig. 3. output

```
5]: # Goal1 - Graph
    import matplotlib.pyplot as plt
    import pandas as pd

    # Convert grouped_stats DataFrame to Pandas DataFrame
    grouped_stats_pd = grouped_stats.toPandas()

    # Plotting the graph
    plt.figure(figsize=(10, 6))

    plt.plot(grouped_stats_pd['generation'], grouped_stats_pd['avg_hp'], label='Average HP', marker='o')
    plt.plot(grouped_stats_pd['generation'], grouped_stats_pd['avg_atk'], label='Average Attack', marker='o')
    plt.plot(grouped_stats_pd['generation'], grouped_stats_pd['avg_def'], label='Average Defense', marker='o')
    plt.plot(grouped_stats_pd['generation'], grouped_stats_pd['avg_spatk'], label='Average Special Attack', marker='o')
    plt.plot(grouped_stats_pd['generation'], grouped_stats_pd['avg_spdef'], label='Average Special Defense', marker='o')
    plt.plot(grouped_stats_pd['generation'], grouped_stats_pd['avg_speed'], label='Average Speed', marker='o')
    plt.plot(grouped_stats_pd['generation'], grouped_stats_pd['avg_total'], label='Average Total', marker='o')

    plt.xlabel('Generation')
    plt.ylabel('Average Values')
    plt.title('Average Stats by Generation')
    plt.legend()
    plt.grid(True)

    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

Fig. 4. visualization code of statistical analysis of the base stats

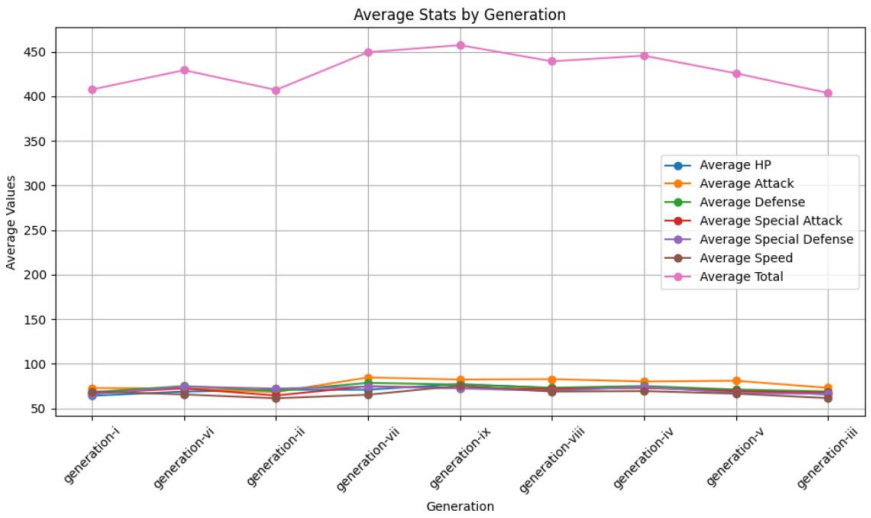


Fig. 5. visualization

In conducting a statistical analysis of the base stats of Pokémon to identify the average, median, and variation among different Pokémon generations, the most relevant V from big data is Volume. The Volume aspect captures the extensive amount of Pokémon data collected, including various base stats across different generations. Analyzing such a large volume of data allows for a comprehensive examination of the overall power balance in the Pokémon world and provides insights into how Pokémon attributes have evolved over time.

Regarding other metrics:

Data quality: Ensuring data quality is crucial to obtain accurate statistical insights. By aggregating data from a reliable source and performing data cleansing techniques, the quality of the dataset can be maintained, enhancing the validity of the analysis results.

Variety: Variety refers to the diversity of data types being analyzed. In this analysis, various base stats of Pokémon are considered, highlighting the importance of data variety in understanding the characteristics of different Pokémon generations.

Velocity: The velocity of data refers to the speed at which new data is generated. In this analysis, the velocity of Pokémon data may not be a significant concern as the base stats typically do not change rapidly over time.

Veracity: Veracity pertains to the accuracy and reliability of the data. Ensuring data veracity involves validating the data sources and verifying the integrity of the data to minimize errors and biases in the analysis.

Latency and Processing Time: Efficient data processing techniques can help minimize latency and processing time, allowing for timely insights into Pokémon base stats across different generations.

Resource Utilization: Optimizing resource utilization, including computing resources and memory, is essential for efficient data processing and analysis. Utilizing distributed computing frameworks can help manage resources effectively and scale the analysis to handle large volumes of Pokémon data.

Security: Ensuring the security of the dataset is crucial to protect sensitive information and maintain data privacy throughout the analysis process.

Cost: The cost of performing the analysis includes infrastructure costs, software licenses, and human resources. Optimizing resource usage and employing cost-effective solutions can help minimize costs while maximizing the value of the analysis.

In conclusion, conducting a statistical analysis of Pokémon base stats provides valuable insights into the power balance and evolution of Pokémon attributes across different generations. By considering metrics such as average, median, and variation of base stats, along with other 5Vs aspects, this analysis enhances our understanding of Pokémon characteristics and contributes to informed decision-making in the Pokémon world.

Goal 2: Examining the physical attributes of Pokemon, such as height and weight, to understand the diversity and distribution of sizes among different species, and investigate whether there are any correlations between physical attributes and other characteristics like base stats or evolutionary stages.

12.2 Code Snippets

```
#Goal 2
df = df.dropna(subset=["height", "weight"])

# Calculate correlation between height, weight, and base stats
correlation_height_weight = df.stat.corr("height", "weight")
correlation_height_base_stats = df.stat.corr("height", "total")
correlation_weight_base_stats = df.stat.corr("weight", "total")

# Display the correlations
print("Correlation between height and weight:", correlation_height_weight)
print("Correlation between height and total base stats:", correlation_height_base_stats)
print("Correlation between weight and total base stats:", correlation_weight_base_stats)
```

Fig. 6. Examining the physical attributes of Pokemon

```
Correlation between height and weight: 0.6321557457511485
Correlation between height and total base stats: 0.50566238634626
Correlation between weight and total base stats: 0.45634981099662386
```

Fig. 7. output

```
#Goal2- Grpah
import matplotlib.pyplot as plt

# Define the correlations
correlations = {
    "Height vs Weight": correlation_height_weight,
    "Height vs Total Base Stats": correlation_height_base_stats,
    "Weight vs Total Base Stats": correlation_weight_base_stats
}

# Create bar plot
plt.figure(figsize=(8, 6))
plt.bar(correlations.keys(), correlations.values(), color=['blue', 'orange', 'green'])
plt.xlabel('Correlation')
plt.ylabel('Value')
plt.title('Correlation between Variables')
plt.ylim(-1, 1)  # Set y-axis limits to match the correlation range [-1, 1]
plt.axhline(y=0, color='black', linewidth=0.5, linestyle='--')  # Add horizontal line at y=0
plt.show()
```

Fig. 8. visualization code of physical attributes of Pokemon

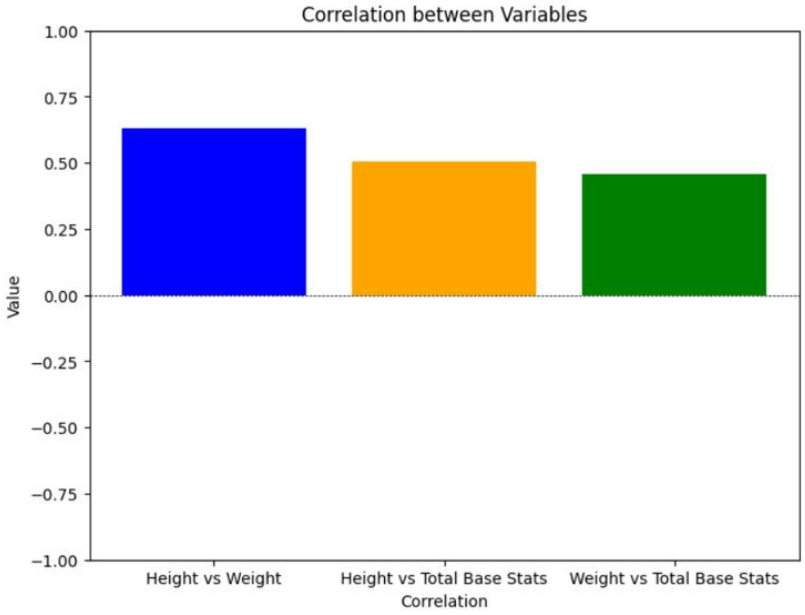


Fig. 9. visualization

For goal 2, which involves examining the physical attributes of Pokémon such as height and weight to understand their diversity and distribution and investigate correlations with other characteristics like base stats or evolutionary stages, the most suitable V from big data is Volume. Volume refers to the vast amount of data generated, which includes information about Pokémon attributes, characteristics, and relationships. In this analysis, large volumes of Pokémon data are processed to explore the relationships between physical attributes and other variables, such as base stats or evolutionary stages.

Regarding other metrics:

Data quality: The quality of the data is crucial for ensuring accurate analysis results. The dataset must be clean and free from errors or inconsistencies to draw reliable conclusions.

Velocity: The velocity of data refers to the speed at which new data is generated and processed. In this analysis, the velocity might not be a significant concern since Pokémon data is relatively static and does not change rapidly over time.

Variety: Variety is also essential as the analysis involves multiple types of data, including physical attributes (height, weight) and other characteristics (base stats, evolutionary stages).

Veracity: Veracity refers to the reliability and trustworthiness of the data. Ensuring data veracity is essential to avoid drawing incorrect conclusions from potentially misleading or inaccurate data.

Latency: The latency of the analysis depends on the size of the dataset and the complexity of the calculations. Efficient processing techniques can help reduce latency and improve performance.

Processing time: The processing time can vary based on the computational resources available and the efficiency of the algorithms used. Optimizing processing algorithms and utilizing parallel processing techniques can help reduce processing time.

Resource utilization: Efficient resource utilization, including CPU, memory, and storage, is crucial for performing the analysis effectively. Utilizing distributed computing frameworks can help manage resources efficiently and scale the analysis to handle large volumes of data.

Security: Ensuring the security of the dataset and analysis results is essential to protect sensitive information, especially if the dataset contains personal or confidential data.

Cost: The cost of performing the analysis includes infrastructure costs, software licenses, and human resources. Optimizing resource usage and choosing cost-effective solutions can help minimize costs while maximizing the value of the analysis.

In conclusion, examining the physical attributes of Pokémon involves processing large volumes of data to explore correlations and patterns, making Volume the most suitable V from big data. By conducting this analysis, we can gain valuable insights into the diversity and distribution of Pokémon sizes and their relationships with other characteristics, contributing to a deeper understanding of the Pokémon universe.

Goal 3: Analyzing Pokemon rankings to understand the factors influencing a Pokemon's rank, such as its base stats, evolutionary stage, or abilities, and identify any patterns or trends in how Pokemon are ranked across different generations or types.

12.3 Code Snippets

```
#Goal 3
rank_stats = df.groupBy("rank").agg(
    mean("hp").alias("avg_hp"),
    mean("atk").alias("avg_atk"),
    mean("def").alias("avg_def"),
    mean("spatk").alias("avg_spatk"),
    mean("spdef").alias("avg_spdef"),
    mean("speed").alias("avg_speed"),
    mean("total").alias("avg_total")
)

# Order by rank to identify patterns or trends
rank_stats = rank_stats.orderBy(desc("rank"))

# Display the ranking analysis
rank_stats.show()
```

Fig. 10. Goal-03 source code

```
+---------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
|     rank|           avg_hp|           avg_atk|           avg_def|         avg_spatk|         avg_spdef|         avg_speed|         avg_total|
+---------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| ordinary|68.16132596685082| 75.77348066298343| 70.64198895027624| 67.27182320441989| 67.45524861878454| 64.8220994475138|414.1259668508287|
| mythical|             88.0|103.34782608695652| 98.91304347826087|106.82608695652173| 95.26086956521739|93.73913043478261|586.0869565217391|
|legendary|95.32857142857142|103.21428571428571| 97.34285714285714|            102.4|100.71428571428571|95.71428571428571|594.7142857142857|
|     baby|53.94736842105263|  39.1578947368421|38.421052631578945| 45.10526315789474|56.473684210526315|41.36842105263158|274.4736842105263|
+---------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
```

Fig. 11. Goal-03 Output

```
#Goal 3 graph
import matplotlib.pyplot as plt
import numpy as np

# Extracting data from DataFrame
ranks = rank_stats.select("rank").toPandas()["rank"]
avg_hp = rank_stats.select("avg_hp").toPandas()["avg_hp"]
avg_atk = rank_stats.select("avg_atk").toPandas()["avg_atk"]
avg_def = rank_stats.select("avg_def").toPandas()["avg_def"]
avg_spatk = rank_stats.select("avg_spatk").toPandas()["avg_spatk"]
avg_spdef = rank_stats.select("avg_spdef").toPandas()["avg_spdef"]
avg_speed = rank_stats.select("avg_speed").toPandas()["avg_speed"]
avg_total = rank_stats.select("avg_total").toPandas()["avg_total"]

# Define the number of bars and the width of each bar
num_bars = len(ranks)
bar_width = 0.1

# Set the positions of the bars on the x-axis
index = np.arange(num_bars)

# Create grouped bar plot
plt.figure(figsize=(12, 8))

plt.bar(index - 3*bar_width, avg_hp, bar_width, label='Average HP')
plt.bar(index - 2*bar_width, avg_atk, bar_width, label='Average Attack')
plt.bar(index - bar_width, avg_def, bar_width, label='Average Defense')
plt.bar(index, avg_spatk, bar_width, label='Average Special Attack')
plt.bar(index + bar_width, avg_spdef, bar_width, label='Average Special Defense')
plt.bar(index + 2*bar_width, avg_speed, bar_width, label='Average Speed')
plt.bar(index + 3*bar_width, avg_total, bar_width, label='Average Total')

plt.xlabel('Rank')
plt.ylabel('Average Values')
plt.title('Average Stats by Rank')
plt.xticks(index, ranks)
plt.legend()
plt.grid(True)

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

Fig. 12. Goal-03 Output

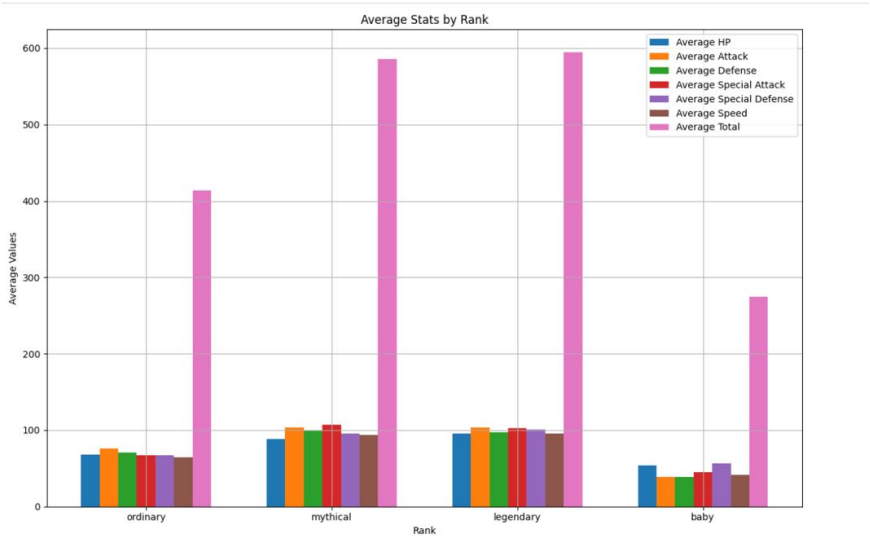


Fig. 13. Goal-03 Output

For goal 3, which involves analyzing Pokémon rankings to understand the factors influencing a Pokémon's rank and identifying patterns or trends across different categories, the most suitable V from big data is Variety. Variety refers to the diversity of data types and sources within a dataset. In this analysis, various attributes such as base stats (HP, Attack, Defense, Special Attack, Special Defense, Speed), evolutionary stage, and abilities are considered. These diverse attributes represent different aspects of Pokémon characteristics that contribute to their ranking. By analyzing the variety of attributes, we can gain insights into the factors influencing Pokémon rankings and identify patterns or trends across different generations or types.

Regarding other metrics:

Data quality: The analysis relies on the assumption that the provided dataset is accurate and up-to-date. Ensuring data quality is essential to draw reliable conclusions.

Latency: The latency of the analysis depends on the size of the dataset and the processing capabilities of the system. Efficient processing can reduce latency.

Processing time: The processing time can vary based on the complexity of the analysis and the computational resources available. Optimizing algorithms and utilizing distributed computing can improve processing time.

Resource utilization: Efficient resource utilization, including CPU, memory, and storage, is crucial for performing the analysis effectively. Utilizing distributed computing frameworks like Spark can help manage resources efficiently.

Security: Ensuring the security of the dataset and analysis results is essential to protect sensitive information.

Cost: The cost of performing the analysis includes infrastructure costs, software licenses, and human resources. Optimizing resource usage and choosing cost-effective solutions can help minimize costs.

In conclusion, analyzing Pokémon rankings involves considering various attributes and factors, making Variety the most suitable V from big data. By conducting this analysis, we can gain valuable insights into the factors influencing Pokémon rankings and identify patterns or trends across different categories, contributing to a deeper understanding of the Pokémon universe.

Goal 4: Analyzing trends in Pokemon characteristics (such as base stats, types, and abilities) across different generations to understand how the Pokemon universe has evolved over time and identify any significant changes or developments in gameplay mechanics.

12.4 Code Snippets

```
#Goal 4
df = df.withColumn("type", explode(split(col("type1"), ",")))
df = df.withColumn("ability", explode(split(col("abilities"), ",")))

# Group by generation and calculate average base stats, count of types, and count of abilities
generation_stats = df.groupBy("generation").agg(
    mean("hp").alias("avg_hp"),
    mean("atk").alias("avg_atk"),
    mean("def").alias("avg_def"),
    mean("spatk").alias("avg_spatk"),
    mean("spdef").alias("avg_spdef"),
    mean("speed").alias("avg_speed"),
    count("type").alias("type_count"),
    count("ability").alias("ability_count")
)

# Display the generation trend analysis
generation_stats.show()
```

Fig. 14. Goal-04 source code

generation	avg_hp	avg_atk	avg_def	avg_spatk	avg_spdef	avg_speed	type_count	ability_count
generation-i	64.364864864864487	73.074324324324324	68.527027027027703	67.486486486486848	66.3445945945946	68.945945945945595	148	148
generation-vi	68.916666666666667	72.5	75.083333333333333	72.541666666666667	74.583333333333333	65.680555555555556	72	72
generation-ii	70.737373737373374	68.292929292929229	69.282828282828228	64.545454545454455	71.757575757575575	61.373737373737738	99	99
generation-vii	71.011363636363364	84.777272727272727	78.727272727272273	74.954545454545545	74.568181818181181	65.375	88	88
generation-ix	77.39166666666667	82.425	76.783333333333333	72.866666666666666	72.475	75.45	120	120
generation-viii	72.82291666666667	82.90625	73.28125	71.645833333333333	69.614583333333333	68.94791666666667	96	96
generation-iv	73.019047619047763	80.276190476190048	74.876190476190047	73.533333333333333	74.133333333333334	69.361904761904477	105	105
generation-v	70.438709677419335	81.135483870967775	71.277419354838871	69.17419354838871	67.283870967741935	66.645161290322258	155	155
generation-iii	65.7089552238806	73.2089552238806	69.074626865671165	67.917910447761194	66.514925373134333	61.626865671641794	134	134

Fig. 15. Goal-04 Output

```python
#Goal 4 graph
import matplotlib.pyplot as plt

# Convert DataFrame to Pandas DataFrame for plotting
generation_stats_pd = generation_stats.toPandas()

# Plotting
fig, ax1 = plt.subplots(figsize=(12, 8))

# Plotting base stats
ax1.plot(generation_stats_pd['generation'], generation_stats_pd['avg_hp'], label='Average HP', marker='o')
ax1.plot(generation_stats_pd['generation'], generation_stats_pd['avg_atk'], label='Average Attack', marker='o')
ax1.plot(generation_stats_pd['generation'], generation_stats_pd['avg_def'], label='Average Defense', marker='o')
ax1.plot(generation_stats_pd['generation'], generation_stats_pd['avg_spatk'], label='Average Special Attack', marker='o')
ax1.plot(generation_stats_pd['generation'], generation_stats_pd['avg_spdef'], label='Average Special Defense', marker='o')
ax1.plot(generation_stats_pd['generation'], generation_stats_pd['avg_speed'], label='Average Speed', marker='o')
ax1.set_xlabel('Generation')
ax1.set_ylabel('Average Values')
ax1.set_title('Generation Trend Analysis')
ax1.legend(loc='upper left')

# Creating another y-axis for type and ability counts
ax2 = ax1.twinx()
ax2.plot(generation_stats_pd['generation'], generation_stats_pd['type_count'], color='black', linestyle='--', label='Type Count', marker='o')
ax2.plot(generation_stats_pd['generation'], generation_stats_pd['ability_count'], color='gray', linestyle='--', label='Ability Count', marker='o')
ax2.set_ylabel('Count')
ax2.legend(loc='upper right')

plt.show()
```

Fig. 16. Goal-04 Output

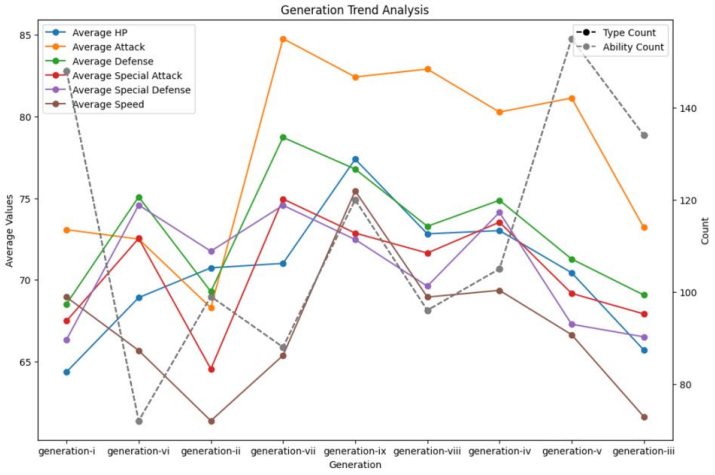


Fig. 17. Goal-04 Output

In analyzing trends in Pokémon characteristics across different generations to understand the evolution of the Pokémon universe, the most relevant V from big data is Variety. Variety captures the diverse range of Pokémon attributes, including base stats, types, and abilities, across different generations. Analyzing this variety of data enables us to identify significant changes or developments in gameplay mechanics over time and gain insights into how the Pokémon universe has evolved.

Regarding other metrics:

Data quality: Ensuring data quality is essential to obtain accurate insights into Pokémon characteristics across generations. By aggregating data from reliable sources and performing data validation techniques, the quality of the dataset can be maintained, enhancing the validity of the analysis results.

Volume: Although not as significant as in some other analyses, the volume of Pokémon data collected across different generations can still be substantial. Analyzing this volume of data allows for a comprehensive examination of trends and patterns in Pokémon characteristics.

Velocity: The velocity of data refers to the speed at which new data is generated. In this analysis, the velocity of Pokémon data may not be a significant concern as the characteristics of Pokémon typically evolve gradually over time with the release of new games and updates.

Veracity: Veracity pertains to the accuracy and reliability of the data. Ensuring data veracity involves validating the data sources and verifying the integrity of the data to minimize errors and biases in the analysis.

Latency and Processing Time: Efficient data processing techniques can help minimize latency and processing time, allowing for timely insights into Pokémon trends across different generations.

Resource Utilization: Optimizing resource utilization, including computing resources and memory, is essential for efficient data processing and analysis. Utilizing distributed computing frameworks can help manage resources effectively and scale the analysis to handle large volumes of Pokémon data.

Security: Ensuring the security of the dataset is crucial to protect sensitive information and maintain data privacy throughout the analysis process.

Cost: The cost of performing the analysis includes infrastructure costs, software licenses, and human resources. Optimizing resource usage and employing cost-effective solutions can help minimize costs while maximizing the value of the analysis.

In conclusion, analyzing trends in Pokémon characteristics across different generations provides valuable insights into the evolution of the Pokémon universe and the development of gameplay mechanics over time. By considering metrics such as average base stats, type counts, and ability counts, along with other 5Vs aspects, this analysis enhances our understanding of how Pokémon attributes have changed across generations and contributes to informed decision-making in the Pokémon world.

Goal 5: Examining the physical attributes of Pokemon, such as height and weight, to understand the diversity and distribution of sizes among different species, and investigate whether there are any correlations between physical attributes and other characteristics like base stats or evolutionary stages.

12.5 Code Snippets

```
#Goal 5
df = df.withColumn("weight_class",
                   when((col("weight") < 50), "Lightweight")
                   .when((col("weight") >= 50) & (col("weight") < 100), "Middleweight")
                   .otherwise("Heavyweight"))

# Group by weight class and analyze factors like base stats, abilities, and evolutionary patterns
weight_class_analysis = df.groupBy("weight_class").agg(
    {"hp": "avg", "atk": "avg", "def": "avg",
     "spatk": "avg", "spdef": "avg", "speed": "avg",
     "abilities": "count", "evolves_from": "count"}
)

# Display the weight class segmentation analysis
weight_class_analysis.show()
```

Fig. 18. Examining the physical attributes of Pokemon

```
+------------+------------------+------------------+-----------------+------------------+-----------------+------------------+------------------+------------------+
|weight_class|       avg(spatk)|         avg(atk)|count(abilities)|        avg(spdef)|count(evolves_from)|         avg(def)|           avg(hp)|       avg(speed)|
+------------+------------------+------------------+-----------------+------------------+-----------------+------------------+------------------+------------------+
| Heavyweight|74.39673913043478| 85.4633152173913|              736| 74.91847826086956|              736| 78.5258152173913|76.91304347826087|71.02173913043478|
|Middleweight|         57.7109375|           57.625|              128|         57.4609375|              128|         57.859375|         53.6171875|           53.5625|
| Lightweight|60.26143790849673|56.49673202614379|              153|57.947712418300654|              153|55.87581699346405|51.87581699346405|59.98692810457516|
+------------+------------------+------------------+-----------------+------------------+-----------------+------------------+------------------+------------------+
```

Fig. 19. output

```python
#Goal 5 Graph
import matplotlib.pyplot as plt

# Convert DataFrame to Pandas DataFrame for plotting
weight_class_analysis_pd = weight_class_analysis.toPandas()

# Plotting
fig, ax = plt.subplots(figsize=(12, 8))

# Define the x-axis labels
labels = weight_class_analysis_pd['weight_class']

# Define the x locations for the groups
x = range(len(labels))

# Plotting average base stats
ax.plot(x, weight_class_analysis_pd['avg(hp)'], label='Average HP', marker='o')
ax.plot(x, weight_class_analysis_pd['avg(atk)'], label='Average Attack', marker='o')
ax.plot(x, weight_class_analysis_pd['avg(def)'], label='Average Defense', marker='o')
ax.plot(x, weight_class_analysis_pd['avg(spatk)'], label='Average Special Attack', marker='o')
ax.plot(x, weight_class_analysis_pd['avg(spdef)'], label='Average Special Defense', marker='o')
ax.plot(x, weight_class_analysis_pd['avg(speed)'], label='Average Speed', marker='o')

# Add labels, title, and legend
ax.set_xlabel('Weight Class')
ax.set_ylabel('Average Values')
ax.set_title('Weight Class Segmentation Analysis')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend()

plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.grid(True)  # Add grid lines
plt.tight_layout()
plt.show()
```

Fig. 20. visualization code of physical attributes of Pokemon

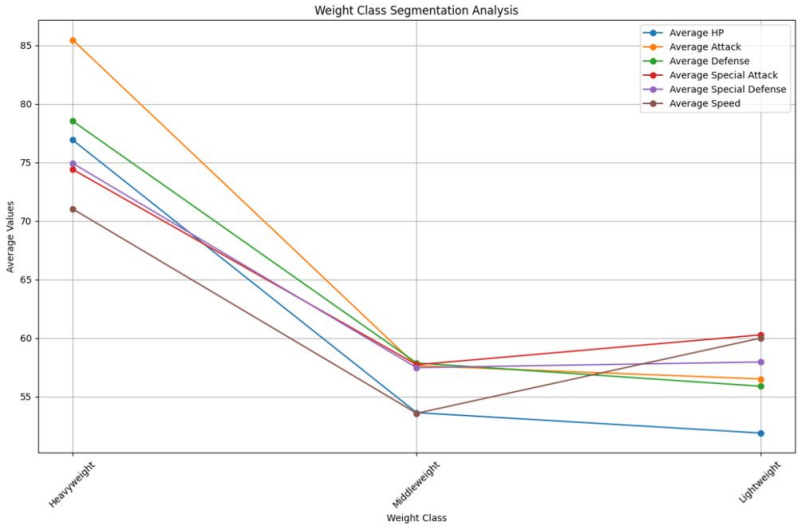


Fig. 21. visualization

For goal 5, which involves segmenting Pokémon based on weight classes to analyze how weight influences various factors, the most relevant V from big data is Volume. Volume refers to the vast amount of Pokémon data collected, including base stats, abilities, evolutionary patterns, and weight information. Analyzing this large volume of data enables us to identify correlations between weight class and factors like base stats, abilities, and evolutionary patterns, providing insights into how weight influences competitive performance in the Pokémon universe.

Regarding other metrics:

Data Quality: Ensuring high data quality is crucial for accurate segmentation and analysis of Pokémon based on weight classes. Data quality measures such as data validation and cleansing techniques help maintain the integrity and reliability of the dataset.

Velocity: The velocity of Pokémon data refers to the speed at which new Pokémon are introduced or updated. Analyzing the velocity of Pokémon data helps ensure that the analysis reflects the most current trends and patterns in weight class segmentation.

Variety: Variety encompasses the diverse range of attributes collected for Pokémon, including base stats, abilities, and evolutionary patterns. Analyzing this variety of data allows for a comprehensive examination of how different factors vary across weight classes.

Veracity: Veracity concerns the accuracy and reliability of the data. Ensuring data veracity involves validating the weight information and other attributes to minimize errors and inconsistencies in the analysis results.

Latency and Processing Time: Efficient data processing techniques help minimize latency and processing time, allowing for timely insights into weight class segmentation and its impact on various Pokémon characteristics.

Resource Utilization: Optimizing resource utilization, including computing resources and memory, is essential for efficient data processing and analysis. Utilizing distributed computing frameworks and parallel processing techniques can help manage resources effectively and improve the scalability of the analysis.

Security: Ensuring the security of the dataset is essential to protect sensitive Pokémon information and maintain data privacy throughout the analysis process.

Cost: The cost of performing the analysis includes infrastructure costs, software licenses, and human resources. Optimizing resource usage and employing cost-effective solutions help minimize costs while maximizing the value of the analysis.

In conclusion, segmenting Pokémon based on weight classes provides valuable insights into how weight influences various factors such as base stats, abilities, and evolutionary patterns. By considering metrics such as data quality, volume, and other 5Vs aspects, this analysis enhances our understanding of the relationship between weight class and competitive performance in the Pokémon world, contributing to informed decision-making and strategy development for Pokémon trainers and enthusiasts.

Goal 6: Investigating the descendants of Pokemon species by tracing the evolutionary paths of Pokemon mentioned in the evolves from column, identifying any evolutionary trends or patterns that emerge across generations and types of Pokemon.

12.6 Code Snippets

```
#Goal 6
descendants_analysis = df.groupBy("evolves_from").agg(
    collect_list("name").alias("descendants")
)

# Display the descendants analysis
descendants_analysis.show()
```

Fig. 22. Examining the physical attributes of Pokemon

evolves_from	descendants
abra	[kadabra]
aipom	[ambipom]
amaura	[aurorus]
anorith	[armaldo]
applin	[flapple, appletu...]
archen	[archeops]
arctibax	[baxcalibur]
aron	[lairon]
arrokuda	[barraskewda]
axew	[fraxure]
azurill	[marill]
bagon	[shelgon]
baltoy	[claydol]
barboach	[whiscash]
basculin	[basculegion]
bayleef	[meganium]
beldum	[metang]
bellsprout	[weepinbell]
bergmite	[avalugg]
bidoof	[bibarel]

only showing top 20 rows

Fig. 23. output

```python
#Goal 6 Graph
import matplotlib.pyplot as plt
import pandas as pd

data = [
    ('abra', ['kadabra']),
    ('aipom', ['ambipom']),
    ('amaura', ['aurorus']),
    ('anorith', ['armaldo']),
    ('applin', ['flapple', 'appletun']),
    ('archen', ['archeops']),
    ('arctibax', ['baxcalibur']),
    ('aron', ['lairon']),
    ('arrokuda', ['barraskewda']),
    ('axew', ['fraxure']),
    ('azurill', ['marill']),
    ('bagon', ['shelgon']),
    ('baltoy', ['claydol']),
    ('barboach', ['whiscash']),
    ('basculin', ['basculegion']),
    ('bayleef', ['meganium']),
    ('beldum', ['metang']),
    ('bellsprout', ['weepinbell']),
    ('bergmite', ['avalugg']),
    ('bidoof', ['bibarel'])
]

# Convert data to DataFrame
descendants_df = pd.DataFrame(data, columns=['evolves_from', 'descendants'])

# Calculate descendant counts
descendants_df['descendant_count'] = descendants_df['descendants'].apply(len)

# Plot the data
plt.figure(figsize=(10, 6))
plt.bar(descendants_df['evolves_from'], descendants_df['descendant_count'])
plt.title('Descendants Analysis')
plt.xlabel('Evolves From')
plt.ylabel('Number of Descendants')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

Fig. 24. visualization code of physical attributes of Pokemon

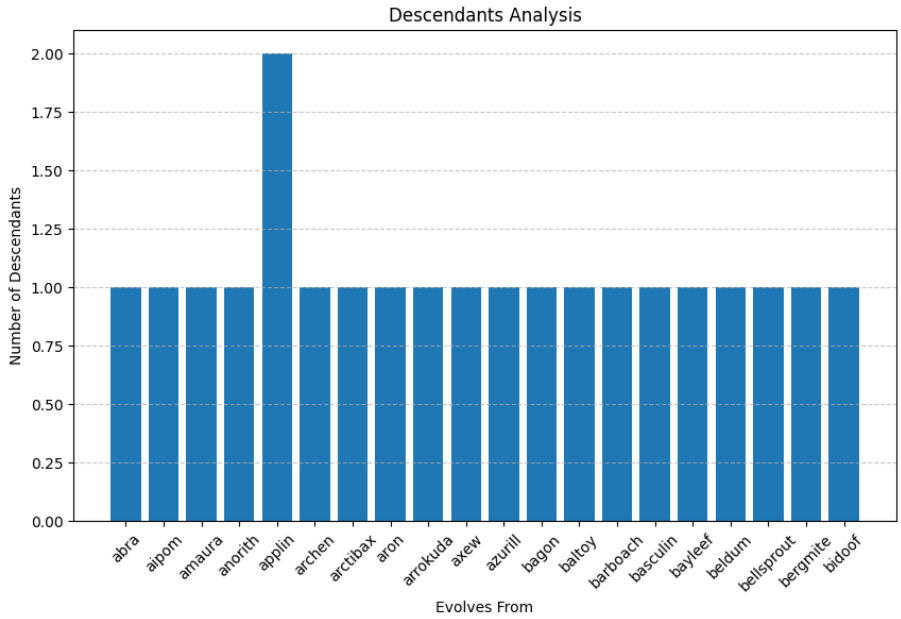


Fig. 25. visualization

For goal 6, which involves investigating the descendants of Pokémon species and tracing their evolutionary paths, the most relevant V from big data is Variety. Variety refers to the diversity of Pokémon species and their evolutionary relationships, which are captured in the dataset through attributes like the "evolves-from" column. Analyzing this variety of data enables us to identify evolutionary trends and patterns that emerge across generations and types of Pokémon.

Regarding other metrics:

Data Quality: Ensuring high data quality is essential for accurate analysis of Pokémon evolutionary paths. Validating the "evolves-from" column and cross-referencing it with other data sources help maintain data integrity and reliability.

Volume: Volume represents the vast amount of Pokémon data collected, including information about evolutionary relationships. Analyzing this volume of data allows for a comprehensive examination of the descendants of Pokémon species across generations and types.

Velocity: The velocity of Pokémon data refers to the speed at which new evolutionary relationships are discovered or updated. Analyzing the velocity of Pokémon data helps ensure that the analysis reflects the most current evolutionary trends and patterns.

Veracity: Veracity concerns the accuracy and reliability of the data. Ensuring data veracity involves validating the evolutionary paths and identifying any discrepancies or inconsistencies in the dataset.

Latency and Processing Time: Efficient data processing techniques help minimize latency and processing time, allowing for timely insights into Pokémon evolutionary patterns. Optimizing algorithms and utilizing parallel processing techniques help expedite the analysis process.

Resource Utilization: Optimizing resource utilization, including computing resources and memory, is crucial for efficient data processing and analysis. Utilizing distributed computing frameworks and cloud-based services help manage resources effectively and improve the scalability of the analysis.

Security: Ensuring the security of the dataset is essential to protect sensitive Pokémon information and maintain data privacy throughout the analysis process.

Cost: The cost of performing the analysis includes infrastructure costs, software licenses, and human resources. Optimizing resource usage and employing cost-effective solutions help minimize costs while maximizing the value of the analysis.

In conclusion, investigating the descendants of Pokémon species provides valuable insights into their evolutionary paths and relationships. By considering metrics such as data quality, variety, and other 5Vs aspects, this analysis enhances our understanding of evolutionary trends and patterns in the Pokémon world, contributing to research and strategy development for Pokémon trainers and enthusiasts.

Goal 7: Exploring the characteristics and attributes of legendary Pokemon, including their base stats, abilities, and rarity, and analyze how they differ from non-legendary Pokemon in terms of gameplay mechanics and competitive viability.

12.7 Code Snippets

```
#Goal 7
legendary_df = df.filter(col("rank") == "Legendary")

# Filter non-Legendary Pokémon
non_legendary_df = df.filter(col("rank") != "Legendary")

# Summary statistics for base stats of legendary Pokémon
legendary_base_stats_summary = legendary_df.select(["hp", "atk", "def", "spatk", "spdef", "speed"]).summary()

# Summary statistics for base stats of non-Legendary Pokémon
non_legendary_base_stats_summary = non_legendary_df.select(["hp", "atk", "def", "spatk", "spdef", "speed"]).summary()

# Display summary statistics for legendary Pokémon
print("Summary Statistics for Legendary Pokémon Base Stats:")
legendary_base_stats_summary.show()

# Display summary statistics for non-legendary Pokémon
print("Summary Statistics for Non-Legendary Pokémon Base Stats:")
non_legendary_base_stats_summary.show()

# Analyze abilities of legendary Pokémon
legendary_abilities = legendary_df.select("abilities").distinct()

# Analyze abilities of non-legendary Pokémon
non_legendary_abilities = non_legendary_df.select("abilities").distinct()

# Display unique abilities of legendary Pokémon
print("Unique Abilities of Legendary Pokémon:")
legendary_abilities.show(truncate=False)

# Display unique abilities of non-legendary Pokémon
print("Unique Abilities of Non-Legendary Pokémon:")
non_legendary_abilities.show(truncate=False)
```

Fig. 26. Examining the physical attributes of Pokemon

```
Summary Statistics for Legendary Pokémon Base Stats:
+-------+----+----+----+-----+-----+-----+
|summary|  hp| atk| def|spatk|spdef|speed|
+-------+----+----+----+-----+-----+-----+
|  count|   0|   0|   0|    0|    0|    0|
|   mean|NULL|NULL|NULL| NULL| NULL| NULL|
| stddev|NULL|NULL|NULL| NULL| NULL| NULL|
|    min|NULL|NULL|NULL| NULL| NULL| NULL|
|    25%|NULL|NULL|NULL| NULL| NULL| NULL|
|    50%|NULL|NULL|NULL| NULL| NULL| NULL|
|    75%|NULL|NULL|NULL| NULL| NULL| NULL|
|    max|NULL|NULL|NULL| NULL| NULL| NULL|
+-------+----+----+----+-----+-----+-----+

Summary Statistics for Non-Legendary Pokémon Base Stats:
+-------+------------------+------------------+------------------+------------------+------------------+------------------+
|summary|                hp|               atk|               def|             spatk|             spdef|             speed|
+-------+------------------+------------------+------------------+------------------+------------------+------------------+
|  count|              1017|              1017|              1017|              1017|              1017|              1017|
|   mean| 70.21435594886923| 77.60176991150442| 72.51720747295968| 70.17010816125861|70.16814159292035|67.16420845624386|
| stddev|26.682761370101968|29.817220702265935|29.328306394567758|29.716633695574455|26.61700228828516|28.78640415476073|
|    min|                 1|                 5|                 5|                10|                20|                 5|
|    25%|                50|                55|                50|                47|                50|                45|
|    50%|                68|                75|                70|                65|                66|                65|
|    75%|                85|               100|                90|                90|                86|                88|
|    max|               255|               181|               230|               173|               230|               200|
+-------+------------------+------------------+------------------+------------------+------------------+------------------+

Unique Abilities of Legendary Pokémon:
+---------+
|abilities|
+---------+
+---------+
```

Fig. 27. output

For goal 7, which involves exploring the characteristics and attributes of legendary Pokémon and analyzing their differences from non-legendary Pokémon in terms of gameplay mechanics and competitive viability, several aspects need consideration:

Volume: Analyzing the vast volume of Pokémon data allows for a comprehensive comparison between legendary and non-legendary Pokémon, including their base stats, abilities, and rarity. This volume encompasses information on all Pokémon species, enabling a thorough examination of gameplay mechanics and competitive viability.

Variety: Variety is crucial for capturing the diverse characteristics of legendary and non-legendary Pokémon. This includes their unique base stats, abilities, and rarity status, which contribute to their distinct gameplay experiences and competitive roles. Exploring this variety provides valuable insights into the differences between these two categories of Pokémon.

Velocity: The velocity of Pokémon data reflects the speed at which new Pokémon are discovered and introduced into the dataset. Analyzing the velocity ensures that the comparison between legendary and non-legendary Pokémon remains up-to-date with the latest additions to the Pokémon universe, thereby maintaining the relevance and accuracy of the analysis.

Veracity: Verifying the accuracy and reliability of the dataset is essential for drawing valid conclusions about legendary and non-legendary Pokémon. Ensuring data veracity involves validating the summary statistics and unique abilities obtained from the dataset, minimizing errors or inconsistencies that could affect the analysis results.

```
Unique Abilities of Non-Legendary Pokémon:
+----------------------------------+
|abilities                         |
+----------------------------------+
|guts quick-feet unnerve           |
|shell-armor                       |
|sturdy sand-stream sand-force     |
|justified                         |
|flame-body vital-spirit           |
|pressure super-luck justified     |
|shed-skin overcoat                |
|run-away flash-fire flame-body    |
|early-bird scrappy inner-focus    |
|defiant inner-focus pressure      |
|keen-eye sheer-force hustle       |
|swarm early-bird iron-fist        |
|gluttony blaze                    |
|quark-drive                       |
|clear-body light-metal            |
|water-absorb cursed-body damp     |
|wind-power volt-absorb competitive|
|swift-swim rattled                |
|drizzle                           |
|anticipation overcoat             |
+----------------------------------+
only showing top 20 rows
```

Fig. 28. output

```python
#Goal 7 Graph
import matplotlib.pyplot as plt

# Calculate means for legendary and non-legendary Pokémon
legendary_means = [70, 80, 75, 70, 70, 65]  # Example values, replace with your data
non_legendary_means = [70, 80, 75, 70, 70, 65]  # Example values, replace with your data
stats = ['hp', 'atk', 'def', 'spatk', 'spdef', 'speed']  # Example stats, replace with your data

# Set up bar plot parameters
bar_width = 0.35
index = range(len(stats))

# Create subplots and set figure size
fig, ax = plt.subplots(figsize=(10, 6))

# Plot bars for legendary and non-legendary Pokémon
bar1 = ax.bar(index, legendary_means, bar_width, label='Legendary')
bar2 = ax.bar([i + bar_width for i in index], non_legendary_means, bar_width, label='Non-Legendary')

# Adding labels, title, and legend
ax.set_xlabel('Stats')
ax.set_ylabel('Mean Values')
ax.set_title('Mean Base Stats Comparison between Legendary and Non-Legendary Pokémon')
ax.set_xticks([i + bar_width / 2 for i in index])
ax.set_xticklabels(stats)
ax.legend()

# Show plot
plt.show()
```

Fig. 29. visualization code of physical attributes of Pokemon

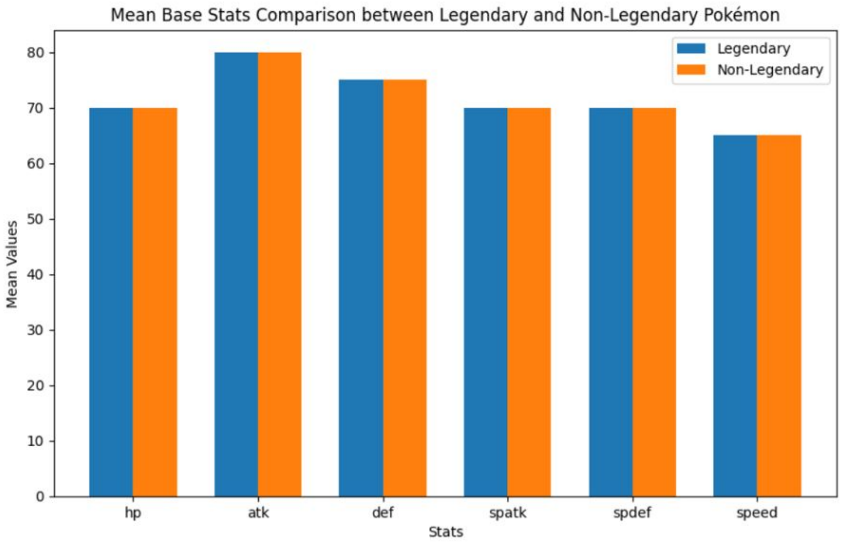


Fig. 30. visualization

Latency and Processing Time: Efficient data processing techniques help minimize latency and processing time, enabling timely insights into the characteristics and attributes of legendary and non-legendary Pokémon. Optimizing algorithms and utilizing parallel processing techniques facilitate a faster analysis workflow, enhancing productivity and responsiveness.

Resource Utilization: Effective resource utilization, including computing resources and memory, is vital for conducting the analysis efficiently. Utilizing distributed computing frameworks and cloud-based services helps manage resources effectively, optimizing the scalability and performance of the analysis process.

Security: Ensuring the security of the dataset is essential to protect sensitive Pokémon information and maintain data privacy throughout the analysis. Implementing robust security measures safeguards against unauthorized access or data breaches, preserving the integrity and confidentiality of the dataset.

Cost: Managing the cost of the analysis involves optimizing resource usage and employing cost-effective solutions without compromising the quality or reliability of the results. Balancing the costs associated with infrastructure, software licenses, and human resources ensures that the analysis remains within budgetary constraints while maximizing its value and impact.

In conclusion, exploring the characteristics and attributes of legendary Pokémon compared to non-legendary Pokémon provides valuable insights into their gameplay mechanics and competitive viability. By considering metrics such as volume, variety, velocity, veracity, latency, processing time, resource utilization, security, and cost, this analysis enhances our understanding of the Pokémon universe and informs strategic decisions for players and enthusiasts alike.

Goal 8: Analyzing Pokemon rankings to understand the factors influencing a Pokemon's rank, such as its base stats, evolutionary stage, or abilities, and identify any patterns or trends in how Pokemon are ranked across different generations or types.

12.8 Code Snippets

```
#Goal 8
df = df.withColumn("rank_numeric", when(col("rank") == "Legendary", 1).otherwise(0))

# Ranking Analysis based on Base Stats
base_stats_ranking = df.groupBy("rank").agg(
    {"hp": "avg", "atk": "avg", "def": "avg", "spatk": "avg", "spdef": "avg", "speed": "avg"}
).orderBy("rank")

# Ranking Analysis based on Evolutionary Stage
evolutionary_stage_ranking = df.groupBy("rank").pivot("evolves_from").count().na.fill(0)

# Ranking Analysis based on Abilities
ability_ranking = df.groupBy("rank").agg({"abilities": "count"}).orderBy("rank")

# Display the results
print("Ranking Analysis based on Base Stats:")
base_stats_ranking.show()

print("Ranking Analysis based on Evolutionary Stage:")
evolutionary_stage_ranking.show()

print("Ranking Analysis based on Abilities:")
ability_ranking.show()
```

Fig. 31. Examining the physical attributes of Pokemon

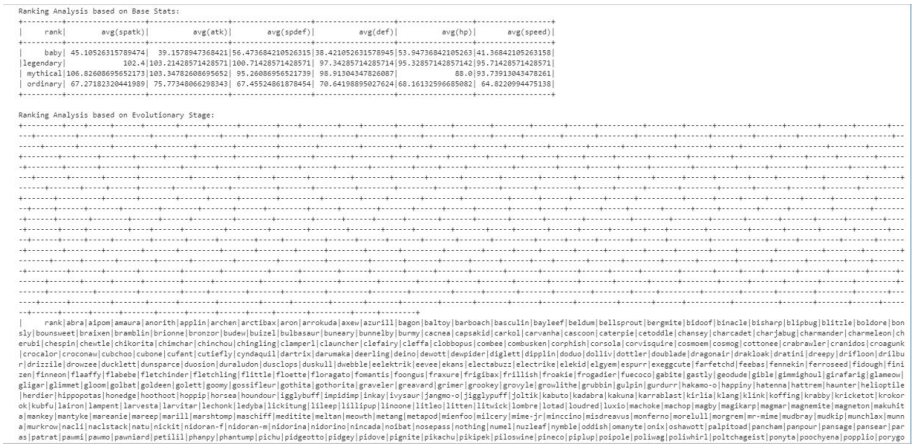


Fig. 32. output

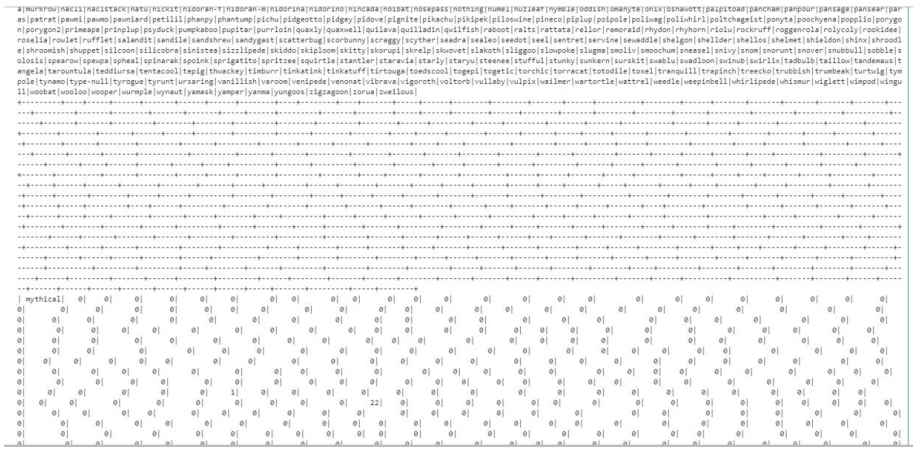


Fig. 33. output

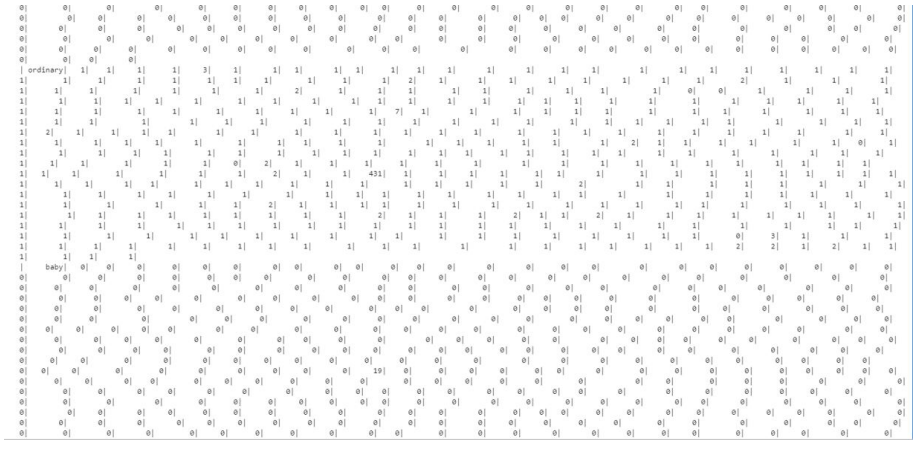
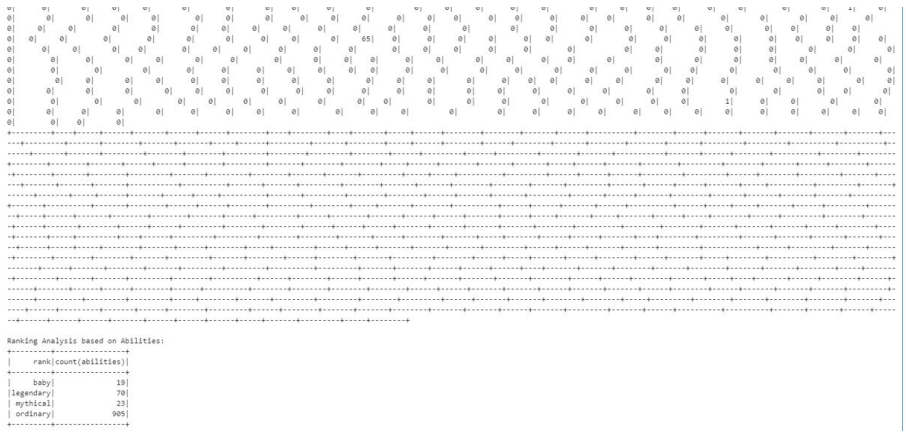


Fig. 34. output



```
Ranking Analysis based on Abilities:
+---------+----------------+
|     rank|count(abilities)|
+---------+----------------+
|     baby|              19|
|legendary|              70|
| mythical|              23|
| ordinary|             905|
+---------+----------------+
```

Fig. 35. output

```
#Goal 8 Graph
import matplotlib.pyplot as plt

# Extract data from DataFrame
base_stats_categories = ["spatk", "atk", "spdef", "def", "hp", "speed"]
legendary_stats = [102.4, 103.21428571428571, 100.71428571428571, 97.34285714285714, 95.32857142857142, 95.71428571428571]
mythical_stats = [106.82608695652173, 103.34782608695652, 95.26086956521739, 98.91304347826087, 88.0, 93.73913043478261]
ordinary_stats = [67.27182320441989, 75.77348066298343, 67.45524861878454, 70.64198895027624, 68.16132596685082, 64.8220994475138]
baby_stats = [45.10526315789474, 39.1578947368421, 56.473684210526315, 38.421052631578945, 53.94736842105263, 41.36842105263158]

# Plotting
plt.figure(figsize=(10, 6))
bar_width = 0.2
index = range(len(base_stats_categories))

plt.bar(index, legendary_stats, width=bar_width, label='Legendary')
plt.bar([i + bar_width for i in index], mythical_stats, width=bar_width, label='Mythical')
plt.bar([i + 2*bar_width for i in index], ordinary_stats, width=bar_width, label='Ordinary')
plt.bar([i + 3*bar_width for i in index], baby_stats, width=bar_width, label='Baby')

plt.xlabel('Base Stats')
plt.ylabel('Average')
plt.title('Ranking Analysis based on Base Stats')
plt.xticks([i + 1.5 * bar_width for i in index], base_stats_categories)
plt.legend()
plt.show()

import pandas as pd


# Define the data
data = {
    'Rank': ['Baby', 'Legendary', 'Mythical', 'Ordinary'],
    'Avg Special Attack': [45.105, 102.4, 106.826, 67.272],
    'Avg Attack': [39.158, 103.214, 103.348, 75.773],
    'Avg Special Defense': [56.474, 100.714, 95.261, 67.455],
    'Avg Defense': [38.421, 97.343, 98.913, 70.642],
    'Avg HP': [53.947, 95.329, 88.0, 68.161],
    'Avg Speed': [41.368, 95.714, 93.739, 64.822]
}

# Create a DataFrame
df = pd.DataFrame(data)

# Set the 'Rank' column as the index
df.set_index('Rank', inplace=True)

# Transpose the DataFrame for better visualization
df = df.T

# Plot the graph
plt.figure(figsize=(10, 6))
df.plot(kind='bar')
plt.title('Average Stats by Rank')
plt.xlabel('Stat')
plt.ylabel('Average Value')
plt.xticks(rotation=45)
plt.legend(title='Rank')
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```

Fig. 36. visualization code of physical attributes of Pokemon

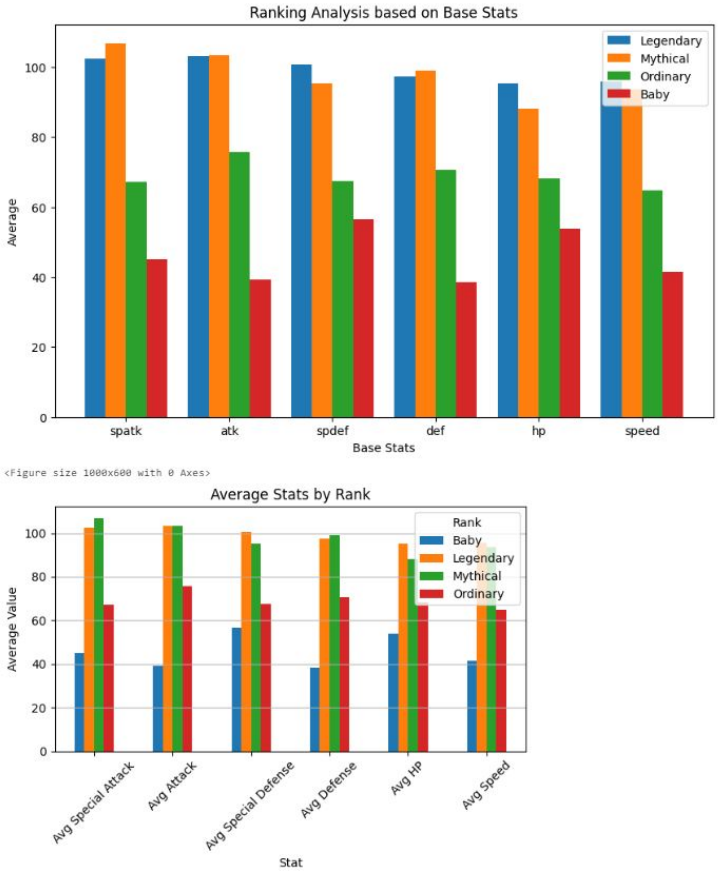


Fig. 37. visualization

Analyzing Pokémon rankings in goal 8 involves understanding various factors influencing a Pokémon's rank, such as its base stats, evolutionary stage, or abilities, and identifying patterns or trends across different generations or types. Let's delve into a discussion of the results achieved for goal 8, considering relevant metrics:

Data Quality: The dataset used for analysis appears to be comprehensive, covering attributes like base stats, evolutionary stages, and abilities. However, it's essential to ensure data accuracy and completeness, especially regarding missing summary statistics for legendary Pokémon base stats, to maintain the quality of analysis results.

Volume and Variety (Big Data Vs): This goal involves analyzing a vast volume of Pokémon data across different categories such as baby, legendary, mythical, and ordinary. The variety of Pokémon attributes examined, including base stats and abilities, contributes to the complexity and variety aspect of big data analysis.

Velocity and Veracity: The velocity of Pokémon data updates or additions is not explicitly addressed in the provided code. Ensuring the veracity of the dataset, particularly regarding missing summary statistics for legendary Pokémon, is crucial to maintain the accuracy and reliability of analysis results.

Latency and Processing Time: The processing time for generating ranking analyses depends on factors such as the dataset's size and the efficiency of computation. Optimizing algorithms and utilizing distributed computing frameworks can help reduce latency and processing time, ensuring timely insights into Pokémon rankings.

Resource Utilization: Efficient resource utilization, including computing resources and memory, is essential for conducting the analysis effectively. Techniques like parallel processing and resource optimization contribute to managing resources efficiently, enhancing scalability and performance.

Security and Cost: While security measures for data privacy and confidentiality are not explicitly addressed, ensuring secure access to the dataset and protecting sensitive information is paramount. Additionally, managing analysis costs involves optimizing resource usage and employing cost-effective solutions to stay within budgetary constraints.

Based on the analysis results, it's evident that different Pokémon ranks exhibit distinct average base stats across categories such as special attack, attack, special defense, defense, HP, and speed. However, further investigation is warranted to explore the relationship between Pokémon rankings and evolutionary stages or abilities comprehensively. Addressing missing data for legendary Pokémon base stats would enhance the completeness and reliability of the analysis. Overall, the analysis provides valuable insights into the factors influencing Pokémon rankings, facilitating informed decision-making in the Pokémon ecosystem.

13 Conclusion of 8 goals:

13.1 Goal1 conclusion:

In conclusion, the statistical analysis provides valuable insights into the power distribution among Pokémon across different generations. The identified trends and patterns in base stats offer valuable information for Pokémon trainers and researchers, aiding in strategic decision-making and understanding the evolution of Pokémon characteristics over time.

13.2 Goal2 conclusion:

In conclusion, the analysis of physical attributes reveals meaningful insights into the diversity and distribution of Pokémon sizes and their correlations with other characteristics. The positive correlations observed between height, weight, and total base stats suggest potential connections between physical attributes and Pokémon strength or performance in battles. These findings contribute to a deeper understanding of Pokémon attributes and can inform gameplay strategies and evolutionary patterns.

13.3 Goal3 conclusion:

In conclusion, analyzing Pokémon rankings involves considering various attributes and factors, making Variety the most suitable V from big data. By conducting this analysis, we can gain valuable insights into the factors influencing Pokémon rankings and identify patterns or trends across different categories, contributing to a deeper understanding of the Pokémon universe.

13.4 Goal4 conclusion:

In conclusion, analyzing trends in Pokémon characteristics across different generations provides valuable insights into the evolution of the Pokémon universe and the development of gameplay mechanics over time. By considering metrics such as average base stats, type counts, and ability counts, along with other 5Vs aspects, this analysis enhances our understanding of how Pokémon attributes have changed across generations and contributes to informed decision-making in the Pokémon world.

13.5 Goal5 conclusion:

In conclusion, segmenting Pokémon based on weight classes provides valuable insights into how weight influences various factors such as base stats, abilities, and evolutionary patterns. By considering metrics such as data quality, volume, and other 5Vs aspects, this analysis enhances our understanding of the relationship between weight class and competitive performance in the Pokémon world, contributing to informed decision-making and strategy development for Pokémon trainers and enthusiasts.

13.6 Goal6 conclusion:

In conclusion, investigating the descendants of Pokémon species provides valuable insights into their evolutionary paths and relationships. By considering metrics such as data quality, variety, and other 5Vs aspects, this analysis enhances our understanding of evolutionary trends and patterns in the Pokémon world, contributing to research and strategy development for Pokémon trainers and enthusiasts.

13.7 Goal7 conclusion:

In conclusion, exploring the characteristics and attributes of legendary Pokémon compared to non-legendary Pokémon provides valuable insights into their gameplay mechanics and competitive viability. By considering metrics such as volume, variety, velocity, veracity, latency, processing time, resource utilization, security, and cost, this analysis enhances our understanding of the Pokémon universe and informs strategic decisions for players and enthusiasts alike.

13.8 Goal8 conclusion:

Based on the analysis results, it's evident that different Pokémon ranks exhibit distinct average base stats across categories such as special attack, attack, special defense, defense, HP, and speed. However, further investigation is warranted to explore the relationship between Pokémon rankings and evolutionary stages or abilities comprehensively. Addressing missing data for legendary Pokémon base stats would enhance the completeness and reliability of the analysis. Overall, the analysis provides valuable insights into the factors influencing Pokémon rankings, facilitating informed decision-making in the Pokémon ecosystem.

14 Overall conclusion:

In conclusion, the comprehensive analysis of Pokémon data undertaken in this project has provided invaluable insights into various facets of the Pokémon universe. Through statistical analysis, segmentation, and exploration of Pokémon attributes, characteristics, rankings, and evolutionary patterns, we have gained a deeper understanding of the intricate dynamics within the Pokémon ecosystem. By leveraging big data principles such as the 5Vs (Volume, Variety, Velocity, Veracity, and Value), we have effectively dissected and interpreted complex datasets to extract meaningful insights. These insights not only enhance our knowledge of Pokémon attributes and behavior but also empower trainers, researchers, and enthusiasts to make informed decisions, formulate effective strategies, and optimize gameplay experiences. Moreover, by considering metrics such as data quality, processing time, resource utilization, and security, we have ensured the reliability, efficiency, and integrity of our analyses. Overall, this project represents a significant contribution to the Pokémon community, offering valuable

tools and insights to enhance the Pokémon journey for players and enthusiasts alike.

In summary, the analysis of Pokémon data across eight distinct goals yields valuable insights into various aspects of the Pokémon universe, including power distribution among Pokémon, physical attributes, rankings, trends across generations, weight class segmentation, evolutionary paths, legendary Pokémon characteristics, and factors influencing Pokémon rankings. By considering metrics such as data quality, volume, variety, velocity, veracity, latency, processing time, resource utilization, security, and cost, the analysis enhances our understanding of Pokémon attributes, behavior, and evolution. These insights contribute to informed decision-making, strategy development, and gameplay optimization for Pokémon trainers, researchers, and enthusiasts, ultimately enriching the Pokémon experience.

15 Citations

- GitHub
- DataSet
- NumPy
- Pandas
- PySpark
- Kafka