Learning Pandas: Descriptive Statistics by Group with the `describe()` Function

Authored by

Mohammed looti

October 27, 2025

RECOMMENDED CITATION

Mohammed looti (2025). *Learning Pandas: Descriptive Statistics by Group with the `describe()` Function*. PSYCHOLOGICAL STATISTICS. Retrieved from https://statistics.arabpsychology.com/?p=4430

In the realm of modern data analysis, the crucial first step is often generating rapid summaries to understand the underlying structure and distribution of a dataset. The pandas library, a cornerstone of the Python data science ecosystem, provides exceptionally powerful tools for this purpose. Chief among these is the built-in `describe()` function, which swiftly calculates core descriptive statistics--such as count, mean, standard deviation, and quartiles--for numerical columns within a DataFrame. This foundational technique offers an immediate high-level overview of the data's central tendency and dispersion.

However, raw datasets rarely benefit from analysis solely on the aggregated whole. True insight often lies in comparing specific subgroups, identifying patterns, and understanding variations across distinct categories. This necessity highlights the indispensable role of the `groupby()` function. By mastering the combination of `groupby()` and `describe()`, data analysts can move beyond generalized summaries to conduct nuanced, segmented analysis, making this technique fundamental to effective exploratory data analysis (EDA).

Core Mechanism: Combining groupby() and describe()

The synergy between `groupby()` and `describe()` allows for the application of statistical summarization independently to partitions of the data. This powerful approach adheres to the 'split-apply-combine' paradigm: first, the data is split into groups based on categorical variables; second, the descriptive statistics (the application) are calculated on a specified numerical column within each group; and finally, these results are combined into a single, comprehensive output structure.

This process is essential when seeking to compare performance metrics, demographic characteristics, or experimental outcomes across different segments of a population or dataset. For instance, instead of knowing the average salary of all employees, we can easily find the average salary broken down by department or job title, revealing critical structural differences that might influence business decisions.

Essential Syntax for Grouped Descriptive Statistics

Applying descriptive statistics on grouped data in pandas is achieved through a concise and highly efficient chained method call. The structure is designed to be readable and immediately understandable, reflecting the sequence of operations being performed on the DataFrame.

The fundamental syntax involves specifying the grouping variable, selecting the target numerical column, and then invoking the summarization method:

df.groupby('group_var').describe()

In this structure, `group_var` designates the column (or list of columns) used to segment the

DataFrame into distinct groups. The `values_var` is the numerical column upon which the statistical calculations will be performed. This chain ensures that the full suite of descriptive statistics is computed individually for every unique category identified in `group_var`, resulting in a multi-indexed summary table that clearly delineates the characteristics of each subgroup.

Setting Up Our Example DataFrame

To effectively illustrate the utility and implementation of grouped descriptive statistics, we will construct a simple, relatable scenario involving athletic performance. We aim to compare scoring and assisting capabilities between two distinct basketball teams by creating a representative pandas DataFrame.

This DataFrame will contain four key pieces of information for several players: the categorical variable **team** (A or B), and the numerical variables **points** scored and **assists** made. This setup provides the ideal framework for demonstrating how grouping by 'team' reveals significant differences in performance metrics.

import pandas as pd

#create DataFrame
df = pd.DataFrame({'team': ,
'points': ,
'assists':})

#view DataFrame
print(df)

team points assists
0 A 8 2
1 A 12 2
2 A 14 3
3 A 14 5
4 B 15 7
5 B 22 6
6 B 27 8
7 B 24 12

The resulting DataFrame, `df`, contains eight entries, representing four players on Team A and four players on Team B. This structure is perfectly balanced for our comparative statistical analysis, allowing us to proceed directly to grouping the data and summarizing the scoring output.

Applying describe() to Grouped Data for Comparison

With the data ready, we can now execute the core analysis: generating a comprehensive statistical summary of the **points** variable, segmented by the **team** variable. This step is crucial for comparing the scoring efficiency and distribution between Team A and Team B.

We initiate the operation by calling `df.groupby('team')`, which logically separates the DataFrame into two subsets based on unique team identifiers. Next, we isolate the numerical variable of interest, `'points'`, assuring that the statistical calculations focus exclusively on scoring performance. Finally, the `.describe()` method computes eight distinct descriptive statistics for the points scored within each individual team's subset.

#summarize points by team
df.groupby('team').describe()

count mean std min 25% 50% 75% max
team
A 4.0 12.0 2.828427 8.0 11.00 13.0 14.00 14.0
B 4.0 22.0 5.099020 15.0 20.25 23.0 24.75 27.0

The resulting table provides an immediate, highly effective side-by-side comparison. By placing the statistical metrics (count, mean, std, etc.) as column headers and the grouping variable ('team') as the index, we can instantly draw comparative conclusions. For instance, the difference in the **mean** score (12.0 for A versus 22.0 for B) clearly indicates that Team B demonstrates a significantly higher average scoring output.

Detailed Interpretation of Descriptive Statistics

Interpreting the rich output of the grouped `describe()` function is paramount to drawing actionable insights. Each statistic offers a unique perspective on the distribution of the **points** variable within the context of each team. Understanding these metrics allows analysts to characterize not only the typical performance but also the range and consistency of player scoring.

count: This metric confirms the number of valid (non-null) observations used in the calculation for each group. Both Team A and Team B have a count of 4.0, verifying that all four player entries per team were included.
mean: The average points value. Team B's mean of 22.0, compared to Team A's 12.0, is the most direct evidence of Team B's superior average offensive contribution.
std: The standard deviation quantifies the variability or spread of the data points around the mean. Team B's standard deviation (5.099) is significantly higher than Team A's (2.828). This suggests

that Team B's individual player scores are more diverse, perhaps indicating the presence of high-variance performers, whereas Team A's scoring is more tightly clustered and consistent.
min and max: These values represent the minimum and maximum scores, defining the full range of performance within each group. Team B's minimum score (15.0) is already higher than Team A's average score (12.0), and Team B's maximum (27.0) far exceeds Team A's maximum (14.0).
Quartiles (25%, 50%, 75%): These percentiles--often referred to as the first quartile, median, and third quartile--are crucial for understanding data skewness and distribution. The 50% percentile (median) provides a robust measure of central tendency, less sensitive to outliers than the mean. The quartile ranges further confirm the overall shift: 75% of Team B's players scored 24.75 points or less, a range that already encompasses all of Team A's individual scores.

In summary, the statistical profile suggests that while Team A's players are consistent scorers, Team B's players score much higher overall, though with greater individual performance variation.

Refining Output Structure with reset_index()

While the multi-indexed output generated by `groupby().describe()` is excellent for immediate visual interpretation, it can pose challenges for subsequent automated processing. When the grouping variable ('team') is part of the DataFrame index, operations such as filtering, merging, or exporting the results require specific indexing methods.

To produce a 'flat' DataFrame where the grouping variable is treated as a regular, accessible column, we can seamlessly chain the `reset_index()` method at the end of the operation. This technique converts the hierarchical index back into a standard column, assigning a new default integer index to the resulting DataFrame.

#summarize points by team and reset index
df.groupby('team').describe().reset_index()

team count mean std min 25% 50% 75% max
0 A 4.0 12.0 2.828427 8.0 11.00 13.0 14.00 14.0
1 B 4.0 22.0 5.099020 15.0 20.25 23.0 24.75 27.0

The revised output clearly shows that 'team' is now a standard data column, making the DataFrame substantially easier to integrate into automated workflows, save to external file formats (like CSV), or use in further complex data manipulation steps.

Further Resources for Pandas Mastery

The ability to calculate descriptive statistics by group is a cornerstone skill in pandas. Mastering the interplay between `groupby()` and `describe()` is essential for performing sophisticated data

segmentations and comparisons. For those seeking to deepen their knowledge of these critical pandas functions and explore more advanced data aggregation techniques, the following authoritative resources are highly recommended:

Pandas GroupBy: split-apply-combine (Official Documentation)
Pandas Descriptive Statistics (Official Documentation)
Pandas GroupBy: Your Guide to Grouping Data in Python (Real Python Tutorial)
Pandas Groupby Tutorial: Learn to Group and Aggregate Data (Dataquest Tutorial)

These resources offer comprehensive guidance to enhance your data preparation and analysis expertise using the powerful capabilities of the pandas library.