Learning to Analyze Categorical Data Using Pandas describe()

Authored by

Mohammed looti

November 15, 2025

In the essential phase of data exploration, the initial summary statistics set the foundation for all subsequent analysis. The **pandas** library, a foundational element of Python's data science toolkit, offers the highly efficient `describe()` function. By default, this function excels at providing a rapid quantitative summary--including the mean, standard deviation, and quartiles--specifically tailored for a **DataFrame**'s numerical features. However, complex, real-world datasets rarely consist solely of numbers; they inevitably contain **categorical variables**, which define qualitative characteristics and require a distinct analytical approach to uncover their structure and patterns.

This comprehensive guide is designed for data professionals seeking to unlock the full potential of the **pandas** `describe()` method when dealing with non-numeric data. We will meticulously detail the two most effective strategies for extracting meaningful descriptive statistics from these features. By mastering these techniques, analysts can ensure a complete and holistic understanding of their dataset, regardless of the underlying variable type, moving beyond a strictly quantitative perspective.

The Standard Behavior of the describe() Method

The **pandas** `describe()` method serves as the primary starting point for initial exploratory data analysis (EDA). When invoked without specifying parameters, its functionality is strictly limited to numerical columns. It furnishes eight core statistics: the **count** of non-missing values, the arithmetic mean, standard deviation, minimum value, the 25th percentile, the median (50th percentile), the 75th percentile, and the maximum value. These metrics are crucial for diagnosing the distribution, identifying central tendency, and quickly spotting potential outliers or skewness within numerical features.

It is important to recognize that these standard quantitative measures--such as calculating a mean or standard deviation--are conceptually inappropriate when applied to **categorical variables**. These variables represent qualitative characteristics like names, groups, or classifications, where mathematical operations like averaging are meaningless. For instance, calculating the average of a list of colors or the spread of nominal identifiers yields nonsensical results. Consequently, to analyze these types of columns effectively, we must explicitly instruct **pandas** to calculate descriptive measures that are specifically designed for qualitative, non-numerical data.

The necessary flexibility for this task is provided by the powerful `include` and `exclude` parameters within the `describe()` method. These parameters empower the user to filter which data types (referred to as **dtypes** within the **pandas** ecosystem) should be included in or excluded from the resulting calculation. This capability is paramount when confronting real-world datasets that invariably contain a complex mixture of quantitative measures and qualitative attributes.

Strategy 1: Focused Categorical Analysis using include='object'

The most precise and targeted method for summarizing **categorical variables** involves setting the `include` parameter to `'object'`. Within the **pandas** data model, columns that store generic strings, mixed data types, or other complex non-numeric data are commonly represented using the **`object` dtype**, making it the de facto indicator for text-based or categorical features.

By executing `include='object'`, the analyst instructs **pandas** to strictly analyze only those columns whose data type matches the `object` indicator. For these selected columns, the **`describe()`** function intelligently switches its output format, delivering a specialized set of statistics tailored for qualitative analysis. This adapted output prioritizes insights into frequency, variety, and dominant categories, rather than traditional metrics like mean or standard deviation.

df.describe(include='object')

The result of this operation provides four crucial statistics for every column identified as a **categorical variable**: the **count** (total non-null entries), the number of **unique** categories present, the **top** (the most frequently occurring or modal) category, and its corresponding **freq**uency of occurrence. These metrics are indispensable tools for rapidly assessing the cardinality, completeness, and identifying the dominant groups within the categorical features of the dataset.

Strategy 2: Universal Categorical Summary via Type Coercion

An alternative, more expansive strategy proves useful when the objective is to obtain categorical summary statistics across the entire **DataFrame**. This includes columns that, while numerically encoded, conceptually function as discrete categories (e.g., specific status codes, year identifiers, or ID numbers). This method involves temporarily treating every column in the dataset as if it were a qualitative, **categorical variable**.

This universal summary is accomplished by chaining the **`astype()` method** to convert all column data types to `object` before subsequently calling `describe()` without any filtering parameters. By explicitly forcing this data type conversion, we compel **pandas** to calculate the specialized categorical summary statistics (count, unique, top, freq) for all features, irrespective of their original **dtype**.

df.astype('object').describe()

While this technique provides a powerful, rapid, and uniform summary of all fields--yielding insights into the number of distinct values, the most common value, and its frequency--analysts must exercise a degree of caution. Converting large numerical columns to the `object` type can

temporarily increase memory consumption and, if not executed carefully (perhaps on a temporary copy), might obscure the true quantitative properties of those columns, leading to misinterpretation if the context is lost.

Practical Demonstration: Constructing a Mixed-Type DataFrame

To clearly demonstrate the functional distinction between these two analytical methods, we will construct a small, representative **DataFrame**. This dataset details fictional basketball player statistics and is intentionally designed to include a mixture of data types: the qualitative **categorical variable** (the team name) and standard quantitative measures (points scored, assists recorded, and rebounds collected).

The following Python code snippet creates the sample data structure. This structure is essential for observing precisely how the output of the `describe()` function changes when different parameters are applied, illustrating the shift from quantitative to qualitative summarization.

import pandas as pd

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

#view DataFrame
print(df)

team points assists rebounds
0 A 18 5 11
1 B 22 7 8
2 C 19 7 10
3 D 14 9 6
4 E 14 12 6
5 F 11 9 5
6 G 20 9 9
7 H 28 4 12

In this defined setup, the **`team`** column, containing distinct string labels, is naturally identified as a categorical feature. Conversely, **`points`**, **`assists`**, and **`rebounds`** are treated as standard numerical features. This distinction allows us to proceed with the specific methods for conducting categorical analysis.

Case Study 1: Isolating Categories with include='object'

Applying the first method, **`df.describe(include='object')`**, permits us to isolate the analysis strictly to the **`team`** column. Since this column stores categorical data and is represented by the `object` data type, the resulting summary will focus exclusively on its qualitative properties.

#calculate descriptive statistics for categorical variables only
df.describe(include='object')

team
count 8
unique 8
top A
freq 1

The resulting output delivers a highly concise summary, optimized for qualitative understanding:

`count`: The value of **8** confirms that all 8 entries in the dataset have a non-missing team value.
`unique`: A value of **8** confirms that there are eight distinct team names. This observation highlights the column's high cardinality, suggesting that each observation represents a unique category.
`top`: The value **`'A'`** is designated as the most frequent category. This metric, often called the **mode**, identifies the most common qualitative observation.
`freq`: The frequency of the top value is **1**. Since all categories are unique in this particular column, the maximum frequency is predictably 1.

This technique is highly recommended for analysts who require a clean, focused summary of only their predefined categorical features, efficiently avoiding the complexity and potential noise introduced by numerical metrics.

Case Study 2: Comprehensive Summary using astype('object')

Next, we demonstrate the second method, which involves temporarily converting the entire **DataFrame** to the `object` data type immediately before calling `describe()`. This approach forces the categorical summary format (count, unique, top, freq) to be calculated for all columns, including **`points`**, **`assists`**, and **`rebounds`**, even though they originally hold numerical data.

#calculate categorical descriptive statistics for all variables
df.astype('object').describe()

team points assists rebounds
count 8 8 8 8

unique 8 7 5 7
top A 14 9 6
freq 1 2 3 2

By interpreting the output for the numerical columns as if they were categorical features, we gain interesting insights into their internal value distribution:

`points` column analysis:

`unique`: 7. This indicates that out of 8 players, 7 scored different point totals, meaning exactly one point total was repeated within the dataset.
`top`: 14. The most frequently occurring point total is 14.
`freq`: 2. The point total of 14 appears twice, confirming its status as the modal value.

`assists` column analysis:

`unique`: 5. There are only five distinct assist totals recorded across the eight players.
`top`: 9. The modal assist total is 9.
`freq`: 3. The assist total of 9 appears three times, representing the highest frequency achieved by any single value in this column.

This method is exceptionally valuable for conducting initial data quality and distribution checks, enabling analysts to rapidly identify surprisingly low cardinality in numerical fields or detect accidental encoding issues where a quantitative column might truly be better analyzed as a discrete factor.

Selecting the Optimal Method for Exploratory Data Analysis

The choice between these two powerful methods depends entirely on the specific analytical objective and the inherent composition of the data. Both are effective tools for summarizing categorical characteristics, but their scope of application differs significantly, necessitating a thoughtful selection.

Choose `df.describe(include='object')` when:

You require a precise and clean analysis solely focused on columns that **pandas** has formally classified as having the `object` data type (i.e., known strings or mixed types).
Your primary analytical interest lies exclusively in the inherent qualitative properties of the data, and you explicitly wish to exclude numerical variables from this phase of the summary.
You need a computationally and memory-efficient operation that avoids the overhead of temporary data type conversions.

Choose `df.astype('object').describe()` when:

You need a uniform categorical summary (count, unique, top, freq) applied to **all** columns in the **DataFrame**, including those that are technically numerical.
You are performing a quick, initial scan to assess the value diversity (cardinality) of every single field, irrespective of its original data type.
You suspect that some numerical columns might function more effectively as discrete categories due to a low number of unique values or highly repetitive patterns.

A firm grasp of these distinct operational behaviors ensures that the analyst selects the most appropriate method for the task, leading to more accurate data interpretations and establishing a robust and comprehensive exploratory data analysis pipeline.

Conclusion

The `describe()` function is far more versatile than its default settings suggest, extending well beyond the summarization of numerical features. By strategically leveraging the `include` parameter or employing temporary type coercion via the **`astype()` method**, data analysts can seamlessly extend its descriptive capabilities to summarize qualitative variables effectively.

Mastering these two complementary techniques ensures efficient and comprehensive summarization of the fundamental properties of any mixed-type **DataFrame**. This dual capability is an indispensable skill for any professional navigating the complexities of real-world data science, guaranteeing that no variable type is inadvertently overlooked during the crucial exploratory phase.

Additional Resources

To further advance your proficiency in **pandas** and deepen your understanding of exploratory data analysis best practices, we highly recommend consulting authoritative resources.

For in-depth documentation on any specific function, including the intricacies of `describe()` and working with **dtypes**, the official **pandas documentation** remains the most reliable and complete source of information.