How to Easily Calculate Quartiles in Pandas

Authored by

stats writer

November 21, 2025

Analyzing data distributions is a fundamental step in statistics and data science. A highly effective method for understanding the spread and central tendency of a dataset is calculating its quartiles. The Pandas library, a cornerstone tool for data manipulation in Python, offers a straightforward and powerful mechanism for performing this calculation: the `.quantile()` function.

The `.quantile()` function is extremely versatile. While its primary use for descriptive statistics often centers around determining the 25th, 50th, and 75th percentiles--which correspond directly to the three main quartiles--it can be leveraged to calculate any arbitrary percentile required. This capability allows data analysts to quickly assess data asymmetry, identify potential outliers, and understand where the bulk of the data lies, making it an indispensable tool when working with a DataFrame.

This comprehensive guide will explore how to effectively utilize the **Pandas `quantile()` function**. We will demonstrate two primary methods: calculating quartiles for a specific column (Series) and calculating them across all relevant numeric columns in a DataFrame simultaneously, providing practical examples and detailed explanations for robust data analysis.

Introduction to Quartiles and Descriptive Statistics

In the field of statistics, **quartiles** are crucial measures of position that divide a dataset into four segments, each containing 25% of the total observations. These divisions are essential for building the five-number summary (minimum, Q1, median, Q3, maximum) and visualizing data spread using box plots. Understanding how data clusters around the central tendency and how it spreads towards the extremes is vital for accurate modeling and reporting.

When performing exploratory data analysis (EDA) using Pandas, calculating quartiles provides instant insight into the distribution's shape. For instance, comparing the distance between Q1 and the minimum versus Q3 and the maximum can immediately highlight skewness in the data. The robust nature of quartiles makes them less susceptible to extreme outliers compared to the mean and standard deviation, offering a stable perspective on data variability.

Typically, analysts focus on three key quartiles when assessing a distribution:

First Quartile (**Q1**): This value represents the 25th percentile. It signifies the point below which 25% of the data falls.
Second Quartile (**Q2**): This value is equivalent to the 50th percentile. It is also known as the **Median**, dividing the data exactly in half.
Third Quartile (**Q3**): This value corresponds to the 75th percentile. It marks the point below which 75% of the data falls.

The Power of the Pandas `quantile()` Function

The Pandas library simplifies the calculation of these statistical measures through the built-in `.quantile()` function. This method is available on both DataFrame objects (to apply the calculation across multiple columns) and Series objects (to target a single column). It is designed to be highly flexible, accepting a single float or a list of floats representing the desired quantile levels.

To calculate the standard quartiles (Q1, Q2, Q3), we pass a list of floating-point numbers corresponding to the 25%, 50%, and 75% marks, which are 0.25, 0.5, and 0.75, respectively. The function handles the interpolation and calculation automatically, adhering to specific statistical methods for defining quantiles (which can be configured using the `interpolation` parameter, though the default settings are often sufficient for standard analysis). Understanding the input structure is critical for efficient use of the function.

When applying `.quantile()` to a large dataset, its vectorization capabilities ensure performance remains high, a characteristic advantage of utilizing Pandas functions over manual statistical looping. Below, we introduce the two main structural methods for applying this function based on the required scope: targeting one column or applying it broadly across the dataset.

Method 1: Calculating Quartiles for a Single Column (Series)

If your analysis requires understanding the distribution of a single numerical variable within your DataFrame, applying the `.quantile()` method directly to the Pandas Series (column) is the most direct approach. This method is ideal when focusing on metrics such as sales figures, test scores, or, as in our example, player statistics like 'points'.

The syntax involves selecting the column using bracket notation and chaining the `.quantile()` method. Inside the method, we provide the list of quantile values. Note that when applied to a Series, the output is another Series indexed by the quantile levels provided (0.25, 0.5, 0.75), making the results instantly readable and usable for further calculation, such as the Interquartile Range (IQR).

Here is the general structure for calculating quartiles for a specific column:

df.quantile()

This concise approach ensures that only the relevant data points are processed, optimizing resource usage when dealing with wide DataFrames containing many non-numeric columns.

Method 2: Calculating Quartiles Across All Numeric Columns (DataFrame)

When conducting a comprehensive exploratory data analysis, it is often necessary to obtain the quartiles for every numeric column simultaneously. Applying `.quantile()` directly to the entire DataFrame achieves this efficiently. By default, Pandas calculates the quantile along `axis=0` (row-wise), meaning it computes the statistic for each column.

A crucial parameter here is `numeric_only=True`. While often set to True by default depending on the Pandas version, explicitly including this argument ensures that the function only attempts to calculate quartiles for columns containing numerical data, preventing errors that arise from attempting to find the median of categorical or string data types.

The output of this method is a DataFrame where the indices represent the percentile levels (e.g., 0.25, 0.50, 0.75), and the columns represent the original numeric columns of the input DataFrame. This organized output is perfect for generating summary tables or feeding data directly into visualization tools.

The structure for this broad calculation method is as follows:

df.quantile(q=, axis=0, numeric_only=True)

This method significantly speeds up the initial statistical assessment of a complex dataset, providing the full quartile distribution summary in a single command execution.

Setting Up the Example Dataset

To illustrate both methods in practice, we will use a sample DataFrame containing fictional team performance statistics. This DataFrame includes both categorical data ('team') and two numerical columns ('points' and 'assists'), allowing us to demonstrate how Pandas handles these different data types during quartile calculation.

We first import the required library and then construct the DataFrame, populating it with 10 rows of sample data. Observing the structure of this initial data is essential before applying statistical transformation. The raw data appears sorted by 'points', which helps illustrate the concept of finding the midpoints (quartiles) in an ordered dataset.

The following code sets up our working environment and displays the resulting DataFrame structure:

import pandas as pd

#create DataFrame

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

#view DataFrame
print(df)

team points assists
0 A 12 2
1 B 14 2
2 C 14 3
3 D 16 3
4 E 24 4
5 F 26 6
6 G 28 7
7 H 30 8
8 I 31 10
9 J 35 15

With the DataFrame defined, we can now proceed to apply the specific quartile calculation methods to gain statistical insights into player performance.

Example 1: Calculate Quartiles for One Column

Detailed Calculation for the 'points' Column

Our first practical example focuses exclusively on the distribution of 'points'. By isolating this variable, we can determine the central tendencies and spread specific to scoring performance, ignoring other variables in the dataset. This isolation is particularly useful when generating summary metrics for individual key performance indicators (KPIs).

We use Method 1 syntax, passing the list to the `.quantile()` function applied to the 'points' Series. Since the number of observations (N=10) is even, the median (Q2) and the quartiles (Q1 and Q3) are calculated using interpolation, finding the average of the two middle values bounding the specified percentile.

Executing the calculation yields the following output:

#calculate quartiles for points column
df.quantile()

0.25 14.5
0.50 25.0
0.75 29.5
Name: points, dtype: float64

The resulting Series clearly defines the quartile boundaries for the 'points' variable, giving us the following breakdown:

The first quartile (Q1) is located at **14.5**. This means 25% of the teams scored 14.5 points or less.
The second quartile (Q2), or the median, is located at **25.0**. Half of the teams scored 25 points or less.
The third quartile (Q3) is located at **29.5**. This indicates that 75% of the teams scored 29.5 points or less.

By understanding these three values, we gain a rapid and accurate representation of how the 'points' values are distributed. For instance, the Interquartile Range (IQR), calculated as Q3 - Q1 (29.5 - 14.5 = 15.0), tells us the range covered by the middle 50% of the data, which is a robust measure of statistical dispersion.

Example 2: Calculate Quartiles for Each Numeric Column

Simultaneous Analysis of 'points' and 'assists'

For a holistic view of team performance, we apply Method 2 to calculate quartiles for both 'points' and 'assists' simultaneously. This approach saves time and ensures consistency in the calculation methodology across all numerical features, allowing for direct comparison of their respective distributions.

When applying `df.quantile()`, Pandas calculates the specified quantiles (Q1, Q2, Q3) for every column that is deemed numeric, efficiently handling the DataFrame structure. The `axis=0` parameter, while default, explicitly reinforces the column-wise computation.

The execution provides a well-organized DataFrame result:

#calculate quartiles for each numeric column in DataFrame
df.quantile(q=, axis=0, numeric_only=True)

points assists
0.25 14.5 3.00
0.50 25.0 5.00
0.75 29.5 7.75

The output clearly displays the quartiles for the two numeric columns in the DataFrame. Notice that the 'points' results match Example 1, confirming the consistency of the calculation. We now also have the distribution for 'assists': Q1 is 3.00, Q2 (median) is 5.00, and Q3 is 7.75.

Interpreting and Applying Quartile Results

The power of calculating quartiles lies in their interpretability. By comparing the calculated quartile values, data scientists can quickly identify if a dataset is symmetric, positively skewed (Q3 is further from the median than Q1), or negatively skewed (Q1 is further from the median than Q3). Furthermore, quartiles are the basis for determining potential outliers using the standard 1.5 * IQR rule.

For the 'assists' column, Q1 (3.00) is close to the minimum (2.0), while Q3 (7.75) is closer to the median (5.0) than the maximum (15). The IQR for assists is 4.75. If we look at the difference between Q3 and the maximum (15 - 7.75 = 7.25), compared to the IQR (4.75), we might suspect the maximum value of 15 is an outlier, suggesting one team had exceptionally high assist performance relative to the rest of the dataset.

In contrast, the 'points' distribution appears more balanced, with Q1 (14.5) and Q3 (29.5) being roughly equidistant from the median (25.0). This comparative analysis highlights the importance of using multiple descriptive statistics rather than relying solely on the mean or count. The consistent, efficient calculation provided by the Pandas `quantile()` function makes this in-depth analysis practical even for massive datasets.

Advanced Considerations: Quantile Calculation Methods

It is important for advanced analysis to recognize that calculating quartiles and quantiles is not universally standardized across all statistical packages. Different methods exist for handling discrete versus continuous data and for interpolating values when the requested percentile falls between two data points (especially with small sample sizes).

The Pandas `quantile()` function uses a default method ('linear') but provides flexibility through the `interpolation` parameter, which supports several types: 'linear', 'lower', 'higher', 'nearest', and 'midpoint'. Each method applies a slightly different rule for determining the quantile value, particularly when exact values are not present in the dataset.

Analysts should refer to the official Pandas documentation to see the various methods that the Pandas **quantile()** function uses to calculate quartiles. Choosing the appropriate interpolation method ensures that the calculated quartiles align with the specific statistical methodology required for the project or academic context, thereby maintaining analytical rigor.