How to Easily Count Distinct Values in SAS Using SQL and PROC FREQ

Authored by

stats writer

December 1, 2025

Determining the number of unique or non-repeated values within a column is a fundamental task in data analysis and preparation using the SAS system. Knowing the count of distinct keyword elements helps analysts understand data cardinality, identify potential data quality issues, and prepare inputs for statistical modeling. Fortunately, SAS provides robust and efficient procedures for achieving this, primarily through the use of PROC SQL and the powerful DISTINCT keyword.

This comprehensive guide explores the primary methodologies available for calculating distinct counts within your SAS dataset. While the focus remains on the highly versatile PROC SQL approach, we will also briefly discuss the alternative utility offered by the PROC FREQ procedure. Understanding the nuances of each method allows users to select the most appropriate syntax based on the specific requirements of their analytical workflow, whether they need a total distinct count or a count grouped by specific variables.

Why Counting Distinct Values is Crucial

The ability to quickly ascertain the number of unique values within a variable is essential for various data management tasks. For instance, when dealing with customer IDs, counting the distinct IDs confirms the actual number of individual customers, regardless of how many transactions they have generated. This metric, often referred to as **cardinality**, is critical for database optimization and data warehousing, ensuring that indexes are appropriately managed and resources are allocated effectively during complex queries.

Furthermore, calculating distinct values is an invaluable first step in data exploration and validation. High distinct counts in categorical variables might suggest the need for binning or aggregation before modeling, while unexpectedly low distinct counts in supposed identifiers (like product codes) could signal data entry errors or corruption. By prioritizing the calculation of these unique counts, analysts gain immediate insights into data quality and distribution characteristics, setting the foundation for reliable statistical analysis later in the pipeline.

Understanding the cardinality of variables helps define the scope of analysis. Variables with very few distinct values are often suitable for grouping or stratification, whereas variables with a high number of unique values (high cardinality) may require special handling or dimensionality reduction techniques to prevent models from becoming overfit or computationally expensive.

Method 1: Counting Distinct Values Using PROC SQL

The most common and arguably most flexible approach for calculating distinct counts in SAS involves using the powerful PROC SQL procedure. This method leverages standard ANSI SQL syntax, making it intuitive for those familiar with database querying languages. The core component of this technique is the use of the DISTINCT keyword placed inside the `COUNT`

aggregate function, instructing the procedure to only tally unique occurrences of the specified variable.

When implementing this method, the analyst specifies the target variable within the `COUNT(DISTINCT variable)` structure inside the `SELECT` statement. This efficient command processes the entire dataset and returns a single numerical result representing the total number of unique values for that variable. This is particularly useful when the goal is to determine the overall variability or domain size of a specific data field, providing a fast, one-line query solution for data exploration.

The standard syntax for counting distinct values in a single column (`var1`) from a dataset named `my_data` is demonstrated below. Note the use of the `AS` clause to assign a descriptive name to the resulting aggregated column, enhancing the clarity and readability of the output table:

The following is the general syntax for counting distinct values in a single column using PROC SQL:

```
proc sql;
select count(distinct var1) as distinct_var1
from my_data;
quit;
```

Method 2: Counting Distinct Values by Group using PROC SQL

A frequent requirement in analytical tasks is not just finding the total distinct count across the entire dataset, but rather finding the unique count of one variable conditional on the values of another variable. This is achieved using the `GROUP BY` clause within PROC SQL, allowing for sophisticated subgroup analysis. This approach is powerful for calculating metrics that must be summarized across different categories, such as finding the variability of sales prices broken down by product category or region.

To implement a grouped distinct count, you include the grouping variable (e.g., `var1`) in the `SELECT` statement, along with the distinct count calculation (e.g., `COUNT(DISTINCT var2)`). Crucially, the grouping variable must also be listed in the `GROUP BY` clause. PROC SQL then calculates the unique count of `var2` separately for every unique instance of `var1`, providing a detailed breakdown of variability across different categories. This is significantly more informative than a simple overall count when assessing data heterogeneity.

The following syntax demonstrates how to calculate the number of unique values in a second variable (`var2`), segmented by the values present in the grouping variable (`var1`). This is a cornerstone technique for generating summary reports where the complexity of one measure is

assessed relative to different categories:

```
proc sql;
select var1, count(distinct var2) as distinct_var2
from my_data
group by var1;
quit;
```

Setting Up the Example Dataset for Demonstration

To illustrate these two powerful PROC SQL methods, we will utilize a small, sample dataset containing basketball team information, specifically tracking the team name and associated points scored across several entries. This sample data structure allows us to clearly observe how the DISTINCT keyword operates in both simple and grouped scenarios, confirming the output manually against the raw data and validating the procedural logic.

The following SAS code uses the `DATA step` and `DATALINES` to create a temporary dataset named `my_data`. This dataset includes two variables: `team` (a character variable denoted by `$`) and `points` (a numeric variable). Notice that teams and points are repeated, creating the necessary redundancy against which the distinct count procedures will work, simulating real-world data where observations are seldom unique across all columns.

After creating the data, we use `PROC PRINT` to display the contents of the newly generated dataset. It is always best practice to review the input data structure before executing analytical procedures to ensure data integrity and variable type correctness, guaranteeing that the distinct calculations are applied to the intended columns.

```
/*create dataset: my_data*/
data my_data;
input team $ points;
datalines;
Mavs 10
Mavs 13
Mavs 13
Mavs 15
Mavs 15
Rockets 9
Rockets 10
Rockets 10
Spurs 18
```

```
Spurs 19
;
run;

/*view dataset contents*/
proc print data=my_data;
```

Obs	team	points
1	Mavs	10
2	Mavs	13
3	Mavs	13
4	Mavs	15
5	Mavs	15
6	Rockets	9
7	Rockets	10
8	Rockets	10
9	Spurs	18
10	Spurs	19

Practical Example 1: Calculating Total Distinct Values in a Single Column

Our first practical demonstration focuses on applying Method 1 to determine the total number of unique teams present in the `my_data` dataset. Despite the dataset containing 10 observations, many of these rows pertain to the same team. We are interested in identifying how many unique team entities exist, regardless of the repetition in their score entries.

We execute the PROC SQL statement, applying `COUNT(DISTINCT team)`. This instructs SAS to iterate through the `team` column, discard duplicate entries (treating 'Mavs' rows 1, 2, 3, 4, 5 as a single count), and ultimately return a count of the unique team names found. The result is stored under the alias `distinct_teams` for easy reference in the output table.

The output clearly shows the result of the aggregation. By manually inspecting our source data, we can see the unique teams are Mavs, Rockets, and Spurs. This confirms that the final count of distinct teams is indeed **3**. This simple calculation is powerful for establishing the domain size of categorical variables and ensuring that data entry has been consistent, preventing potential data quality issues like variations in spelling.

```
/*count distinct values in team column*/
```

```
proc sql;
select count(distinct team) as distinct_teams
from my_data;
quit;
```

distinct_teams
3

From the resulting output, we can observe that there are **3** distinct values identified in the `team` column.

Practical Example 2: Analyzing Distinct Scores per Team

In this second example, we apply Method 2 to count the unique scores (`points`) achieved by each individual team (`team`). This requires grouping the data by the `team` variable before applying the distinct count to the `points` column. The goal is to see the variety of point totals recorded for each franchise, which is a common task when normalizing or summarizing detailed transaction data.

The PROC SQL statement selects the grouping variable (`team`) and the calculated distinct count (`COUNT(DISTINCT points) AS distinct_points`). The mandatory `GROUP BY team` clause ensures that the aggregation function resets and performs the unique count calculation separately for 'Mavs', 'Rockets', and 'Spurs', rather than treating all 10 rows as one large group. This is the mechanism that generates row-level summaries based on the specified categorical variable.

The resulting table provides a clear performance profile based on scoring variability. We can observe that the Mavs recorded 3 distinct point totals (10, 13, 15), the Rockets recorded 2 distinct point totals (9, 10), and the Spurs recorded 2 distinct point totals (18, 19). This level of grouped analysis is invaluable for producing tailored summary statistics and key performance indicators (KPIs) in large-scale data environments where variability within subgroups is a critical measure.

```
/*count distinct values in points column, grouped by team*/
proc sql;
select team, count(distinct points) as distinct_points
from my_data
group by team;
quit;
```

team	distinct_points
Mavs	3
Rockets	2
Spurs	2

The final aggregated table provides the count of unique point values recorded for each respective team.

Alternative Approach: Utilizing PROC FREQ

While PROC SQL with the DISTINCT keyword is the most direct way to get a single numerical count of unique values, the PROC FREQ procedure offers a detailed alternative, particularly useful when you need the complete frequency distribution of all unique values. PROC FREQ generates one-way to n-way frequency tables, listing every unique value and the number of times it appears in the dataset, along with percentages and cumulative frequencies.

To determine the distinct count using PROC FREQ, one runs the procedure on the desired variable using the `TABLES` statement. The resulting output table lists all unique values; the count of rows in this output table (excluding totals) corresponds directly to the distinct count. Although not as efficient as PROC SQL for a simple tally, PROC FREQ provides the added benefit of showing the full distribution, which is often crucial for data validation and initial descriptive statistics, enabling the analyst to see exactly which values are unique and their respective occurrence counts.

For example, running the code `PROC FREQ DATA=my_data; TABLES team; RUN;` would output a table showing 'Mavs', 'Rockets', and 'Spurs', along with their respective frequency counts (5, 3, and 2). Simply counting the number of rows in this output table reveals the distinct count (3). Analysts typically choose PROC SQL for programmatic distinct counting and PROC FREQ when detailed frequency distribution statistics are also required for a comprehensive view of the data's composition.

Summary and Conclusion

Counting unique values is a core activity in data preparation using SAS. We have demonstrated that the most direct and versatile method for calculating distinct counts is through the use of the PROC SQL procedure combined with the `COUNT` aggregate function and the DISTINCT keyword. Whether you need a simple overall tally or a complex, grouped analysis using the `GROUP BY` clause, PROC SQL provides a clean, highly readable, and exceptionally efficient solution

consistent with standard SQL practices.

Mastering these two techniques--single column distinct counts and grouped distinct counts--empowers analysts to perform rapid diagnostic checks on their dataset cardinality and understand the variability within subgroups. Choosing between PROC SQL and alternatives like PROC FREQ depends entirely on whether the analysis requires just the count itself for reporting, or the full frequency distribution table for detailed statistical assessment.

For those looking to expand their knowledge of data manipulation in SAS, exploring other critical procedures such as `PROC SORT`, `PROC MEANS`, and `PROC TRANSPOSE` is highly recommended. These tools, used in conjunction with distinct counting techniques, form the backbone of advanced statistical programming within the SAS environment, providing the necessary foundation for robust data modeling and reporting.

The following tutorials explain how to perform other common tasks in SAS: