Learning SAS: Understanding Dataset Structure with PROC CONTENTS

Authored by

Mohammed looti

May 9, 2026

In the crucial domains of statistical analysis and data management, the ability to rapidly understand the underlying structure and characteristics of your **dataset** is a non-negotiable prerequisite for successful analysis. Before any sophisticated modeling or reporting can commence, analysts must obtain a comprehensive structural overview of the data at hand. This fundamental requirement is perfectly addressed by the **PROC CONTENTS** procedure in **SAS**, which serves as an indispensable tool by providing a detailed summary of a SAS dataset without the need to load or view the raw data values.

This robust procedure empowers users to quickly inspect vital metadata elements, such as the total count of **observations** (rows), the number of **variables** (columns), and the specific **data type** assigned to each variable, alongside other critical structural details. Possessing this information is paramount for maintaining data integrity, performing initial data validation checks, and efficiently planning subsequent analytical steps. By routinely utilizing **PROC CONTENTS**, analysts can ascertain the scope and inherent nature of their data, solidifying its role as the routine first command in any thorough data exploration workflow within **SAS**.

The following guide is designed to walk you through the practical application of **PROC CONTENTS**, illustrating its diverse utility through clear, step-by-step examples. We will focus on how to accurately interpret the generated output and explore various procedural options that substantially enhance its functionality. Ultimately, this article aims to equip you with the expertise needed to efficiently characterize and validate any **dataset** you encounter.

Establishing the Sample Dataset for Demonstration

To effectively illustrate the powerful capabilities of **PROC CONTENTS**, our initial step involves constructing a readily accessible sample **dataset** within the **SAS** environment. This dataset, which we will name `original_data`, will house hypothetical statistical information related to basketball players, specifically tracking their team assignment, points scored, and total rebounds. This simple, controlled structure will provide a practical and unambiguous foundation for understanding how the **PROC CONTENTS** procedure processes and subsequently summarizes data metadata.

The construction of this sample data relies on executing a **DATA step**, where we first define the necessary variables and then populate them with static raw data using the `INPUT` and `DATALINES` statements. This is the standard, efficient methodology for creating small, illustrative **datasets** directly embedded within a SAS program script. Following the successful execution of the **DATA step**, we will employ **PROC PRINT**. The purpose of **PROC PRINT** is to visually confirm the content and integrity of our newly generated dataset before we move forward to analyze its structural properties using **PROC CONTENTS**.

The complete SAS code block required for generating and subsequently displaying our sample

basketball player dataset is provided below:

```
/*create dataset*/
data original_data;
input team $ points rebounds;
datalines;
A 12 8
A 12 8
A 12 8
A 23 9
A 20 12
A 14 7
A 14 7
B 20 2
B 20 5
B 29 4
B 14 7
B 20 2
B 20 2
B 20 5
;
run;

/*view dataset*/
proc print data=original_data;
```

Obs	team	points	rebounds
1	A	12	8
2	A	12	8
3	A	12	8
4	A	23	9
5	A	20	12
6	A	14	7
7	A	14	7
8	B	20	2
9	B	20	5
10	B	29	4
11	B	14	7
12	B	20	2
13	B	20	2
14	B	20	5

As clearly demonstrated by the output generated by **`PROC PRINT`**, our `original_data` dataset has been correctly instantiated. The dataset encompasses data for 14 individual player entries (rows) and is defined by three distinct **variables**: `team`, `points`, and `rebounds`. This necessary visual confirmation establishes that the data is structured exactly as anticipated, allowing us to confidently transition to the metadata analysis phase.

Executing the Basic `PROC CONTENTS` Command

With our sample dataset successfully validated and prepared, we can now proceed to execute the most fundamental form of **`PROC CONTENTS`**. The foundational purpose of this procedure is to deliver a concise yet comprehensive descriptive summary of the specified SAS dataset, focusing entirely on its inherent structural characteristics rather than displaying its actual data values. This distinction makes it exceptionally valuable when dealing with large-scale datasets, where viewing the complete content would be inefficient, time-consuming, or simply unnecessary for metadata inspection.

To obtain this descriptive summary, the analyst simply needs to invoke the **`PROC CONTENTS`** statement followed by the mandatory `DATA=` option, specifying the target dataset name. Upon execution, the procedure dynamically interrogates the dataset's metadata, compiling and presenting this information in a highly structured format. This output typically consists of several distinct tables, each offering a different facet of the dataset's composition. This initial, high-level

overview is often sufficient for performing a quick check of the dataset's basic properties and confirming both its existence and overall structural integrity.

The code snippet below illustrates the most straightforward and common application of **PROC CONTENTS** when applied to our `original_data` dataset:

```
/*view contents of dataset*/
proc contents data=original_data;
```

The CONTENTS Procedure

Data Set Name	WORK.ORIGINAL_DATA	Observations	14
Member Type	DATA	Variables	3
Engine	V9	Indexes	0
Created	11/18/2022 10:25:35	Observation Length	24
Last Modified	11/18/2022 10:25:35	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	NO
Label			
Data Representation	SOLARIS_X86_64, LINUX_X86_64, ALPHA_TRU64, LINUX_IA64		
Encoding	utf-8 Unicode (UTF-8)		

Engine/Host Dependent Information	
Data Set Page Size	131072
Number of Data Set Pages	1
First Data Page	1
Max Obs per Page	5431
Obs in First Data Page	14
Number of Data Set Repairs	0
Filename	/saswork/SAS_work75D70001500F_odaws04-usw2.oda.sas.com/SAS_workFD330001500F_odaws04-usw2.oda.sas.com/original_data.sas7bdat
Release Created	9.0401M6
Host Created	Linux
Inode Number	537009054
Access Permission	rw-r--r--
Owner Name	u1311781
File Size	256KB
File Size (bytes)	262144

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
2	points	Num	8
3	rebounds	Num	8
1	team	Char	8

Once executed, **PROC CONTENTS** produces a detailed and multi-sectioned output, the components of which we will analyze systematically in the next section. This output is logically organized into sections that detail the dataset's macro characteristics and the micro-characteristics of its constituent variables. Mastering the interpretation of these structured tables is absolutely essential for effective data validation and comprehensive data exploration.

Detailed Interpretation of the `PROC CONTENTS` Output

The output generated by the **`PROC CONTENTS`** procedure is consistently partitioned into three primary sections or tables. Each section contributes a unique set of information, and a holistic understanding of all three is necessary to form a complete picture of the dataset's metadata and structure.

The first crucial section, typically titled "Data Set Information," delivers a high-level summary of the dataset itself, often proving to be the most immediately actionable component of the output. This section provides critical statistics regarding the dataset's size and general operational attributes. Essential values to scrutinize here include the **Data Set Name** (confirming the library and name, e.g., `WORK.ORIGINAL_DATA`), the **Member Type** (confirming it is a `DATA` dataset), and the **Engine** (the mechanism used by SAS, such as `V9`). Furthermore, it provides timestamps for **Created/Last Modified**, which are vital for tracking data lineage, and the overall counts: **Observations**, which tells us there are **14** player records, and **Variables**, confirming the presence of **3** columns. Other details like whether the data is **Compressed** or **Sorted** are also listed here.

The second section focuses on "Engine/Host Information," providing technical specifications related to the **engine** and the underlying **host** environment where the dataset was created and is currently stored. While this technical metadata is highly relevant for system administrators, particularly in complex or distributed computing environments, it generally holds less immediate importance for typical data analysts focused solely on the analytical properties of the data. This section details aspects such as the operating system, machine type, and file system characteristics, which primarily pertain to the infrastructure that supports the SAS operations.

The third and arguably most detailed table, "Variables in Alphabetical Order," lists every **variable** contained within the dataset, arranged alphabetically by default. For each listed variable, this table provides indispensable metadata essential for subsequent data preparation and analysis. Key columns include the variable's **#** (position in the dataset), the **Variable** name, and critically, the **Type**, which definitively specifies the **data type** as either **`Numeric`** or **`Character`**. In our example: `points` and `rebounds` are **numeric variables**, while `team` is a **character variable**. The table also specifies **Length** (storage in bytes), **Format** (display rules), **Informat** (input reading rules), and the descriptive **Label**.

Data Set Information Key Metrics:

Data Set Name (e.g., `WORK.ORIGINAL_DATA`)
Engine Type (e.g., `V9`)
Number of **Observations** (Rows)
Number of **Variables** (Columns)

Variable Attributes:

Variable Name and Position
Type: `Numeric` or `Character`
Length, Format, Informat, and Label

Modifying Variable Display Order with `ORDER=VARNUM`

By default, when **`PROC CONTENTS`** generates the third output table detailing the variables, the list is conventionally sorted in alphabetical order. While this standard alphabetical arrangement is beneficial for quickly isolating a variable by its name, it frequently fails to reflect the logical or physical structure of the data file itself. In many analytical scenarios, particularly when consuming data from external systems, the precise order in which **variables** appear in the dataset holds significant conceptual importance and can be more intuitive for validation purposes.

To address this need and present the **variables** according to their physical sequence within the data file--the order in which they were originally defined or read into SAS--**`PROC CONTENTS`** provides the useful option: `ORDER=VARNUM`. This option overrides the default sorting mechanism, ensuring that the variable list is displayed based on its numerical position within the dataset structure. Employing `ORDER=VARNUM` is especially helpful when working with fixed-format legacy data or when verifying that the input variables align precisely with the definitions established in the preceding **`DATA step`** layout. It provides an immediate and accurate visual representation of the dataset's physical arrangement.

The SAS code snippet below demonstrates the necessary modification to the **`PROC CONTENTS`** statement, illustrating how to incorporate the `ORDER=VARNUM` option:

```
/*view contents of dataset and retain original order of variables*/
proc contents data=original_data order=varnum;
```

Variables in Creation Order

#	Variable	Type	Len
1	team	Char	8
2	points	Num	8
3	rebounds	Num	8

As clearly shown in the resulting output table, the **variables** are now correctly listed in the order they were initially defined in the **`DATA step`**: `team` (position 1), followed by `points` (position 2), and finally `rebounds` (position 3). This seemingly minor adjustment in presentation offers a significant

advantage in terms of readability and facilitates efficient validation checks, especially when managing datasets where the column sequence carries intrinsic meaning.

Advanced Options for Tailored Metadata Retrieval

Moving beyond its core function, **PROC CONTENTS** incorporates several advanced options that facilitate more customized and efficient metadata retrieval, allowing users to fine-tune the procedure's output to match specific programmatic or informational needs. These options are instrumental in streamlining the data exploration process by either limiting output detail or providing highly specialized structural information.

One highly practical option is the use of the `SHORT` keyword, which instructs the procedure to generate a highly abbreviated output. This concise report focuses primarily on listing variable names and their corresponding types, intentionally suppressing the detailed information such as formats, lengths, and dataset history found in the standard output. The `SHORT` option is perfect for situations demanding only a rapid inventory of the variables present. Conversely, the `POSITION` option can be deployed to include the starting byte position of each variable within the observation record. This detailed, low-level information is particularly relevant for intricate data manipulation tasks or when debugging complex input definitions within the **DATA step**.

Perhaps the most powerful advanced feature is the `OUT=` option, which enables the user to capture the entire **PROC CONTENTS** output and save it directly into a new, accessible SAS dataset. This capability is invaluable for programmatic manipulation of metadata, as it allows subsequent SAS procedures to dynamically read and react to information about variable names, **data types**, and lengths. For instance, an analyst could use the resulting metadata dataset to conditionally process variables based on their properties or automatically generate data validation code. When leveraging `OUT=` for programmatic purposes, it is common practice to also include the `NOPRINT` option, which suppresses the display of the standard output in the log and output windows, ensuring that only the desired metadata dataset is created.

Conclusion: Summarizing the Power of `PROC CONTENTS`

This tutorial has thoroughly explored the functionality and critical importance of the **PROC CONTENTS** procedure within the **SAS** programming environment. This essential procedure acts as a fundamental mechanism for acquiring a comprehensive summary of a **dataset**'s structure, supplying crucial metadata that is indispensable for effective data governance, validation, and advanced analysis. Its core strength lies in its ability to rapidly characterize any dataset, making it a critical initial step in virtually every data-driven project.

Specifically, we have highlighted that **PROC CONTENTS** is uniquely suited for extracting and

confirming the following key structural details:

The overall dimension of a dataset, quantified by the total number of **variables** (columns) and **observations** (rows).
The specific names and fundamental **data type** (either `Numeric` or `Character`) for every variable in the dataset, along with their precise storage lengths and any associated formats or informats.
Technical details pertaining to the dataset's creation, modification history, the underlying **engine** utilized, and other relevant environmental parameters.

In professional practice, integrating the execution of **`PROC CONTENTS`** into your workflow before proceeding with any statistical modeling, data manipulation, or reporting is universally regarded as a highly recommended best practice. This proactive approach ensures a robust understanding of data size and structure, aids in the early identification of potential data quality issues, and guarantees a smoother, more reliable progression through all subsequent SAS programming tasks.

Further Learning and Essential Resources

To continue building proficiency in SAS programming and enhance your data management techniques, it is beneficial to explore additional tutorials and resources that cover a breadth of common data processing and analytical tasks. Developing strong competency across various procedures and manipulation techniques is essential for becoming an accomplished SAS user.

The following resources offer guidance on how to perform other common and necessary data manipulation tasks within the SAS environment: