PassQueen
a.k.a dumpsqueen.com

SAS Certified Associate: Programming Fundamentals Using SAS 9.4

SAS Institute A00-215

Version Demo

Total Demo Questions: 13

Total Premium Questions: 130

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Accessing and Creating Data Sets	31
Topic 2, Data Management	18
Topic 3, Data Manipulation	25
Topic 4, Reports	42
Topic 5, Identifying and Correcting Data, Syntax, and Programming Logic Errors	14
Total	130

QUESTION NO: 1

Which PROC SORT option allows you to create an output data set of the sorted data?

A. Data=

B. SORTOUT=

C. OUTPUT=

D. OUT=

ANSWER: D

Explanation:

`OUT=` is the PROC SORT option used to write the sorted observations to a separate output SAS data set. It is specified in the PROC SORT statement followed by the name of the target data set, such as `proc sort data=work.sales out=work.sales_sorted; by region; run;`. In this example, PROC SORT reads `WORK.SALES`, sorts the observations by `REGION`, and stores the sorted result in `WORK.SALES_SORTED`. This preserves the original input data set rather than replacing it. The output data set can then be used by later DATA and PROC steps, and it receives the sorting metadata associated with the BY variables. If an `OUT=` data set is not specified, PROC SORT normally writes the sorted observations back to the input data set identified by `DATA=`. SAS documents `OUT=` as the option that names the output data set created by PROC SORT; it can be a one-level name in the default WORK library or a two-level name for a permanent library. See the SAS PROC SORT statement documentation and the SAS PROC SORT overview.

QUESTION NO: 2

Which statement is true regarding the SET statement?

A. The SET statement specifies an input data set in the DATA step.

B. The SET statement specifies an output data set in the PROC SORT step.

C. The SET statement specifies an input data sot in the PROC SORT step

D. The SET statement specifies an output data 5et in the DAI A step.

ANSWER: A

Explanation:

The SET statement specifies an input SAS data set in a DATA step. It is an executable DATA-step statement that reads observations from an existing SAS data set and makes their variables available in the program data vector, where subsequent DATA-step statements can process, transform, subset, or otherwise manipulate them. For example, `data work.new;` defines the output data set, while `set work.old;` identifies the existing input data set whose observations are read. The DATA step normally iterates once for each observation read by SET, writing observations to the output data set unless program logic prevents output. SET can also name multiple input data sets, allowing observations to be read sequentially from each named data set. In contrast, output data set creation is specified in the DATA statement, not SET. SAS documents SET as a DATA-step statement used to read observations from one or more SAS data sets; its syntax and processing behavior are described in the SAS SET Statement documentation. For a broader explanation of how SAS DATA steps read input and create output data sets, see the SAS DATA Statement documentation.

QUESTION NO: 3

Given the program shown below:

Given the partial PROC PRINT report below:

Why are the labels for msbp, MPG_city, and MPG_Highway NOT displaying in the PROC PRINT report^

A. You must use the LABEL option on the PROC PRINT statement

B. You must put the LABEL statement in the PROC PRINT step

C. You must put the LABEL statement after the KEEP statement In the DATA stop

D. You must use a single LABEL statement for each variable.

ANSWER: A

Explanation:

You must use the LABEL option on the PROC PRINT statement. A LABEL statement in a DATA step associates descriptive text with variables as permanent variable-label metadata in the output data set. For example, it can assign a display label such as “Manufacturer’s Suggested Retail Price” to MSRP or more reader-friendly text to the city and highway mileage variables. However, PROC PRINT displays variable names as column headings by default, even when labels are available in the data set’s descriptor information. Specifying the LABEL option on the PROC PRINT statement instructs the procedure to use the stored variable labels for report headings instead of the SAS variable names. The DATA step LABEL statement does not need to be relocated or repeated for each variable; one LABEL statement can assign labels to multiple variables. Once the data set contains the intended labels, adding the option produces the desired labeled headings, for example: `proc print data=work.cars label;`. SAS documents LABEL as a PROC PRINT statement option that uses variable labels as column headings. See the SAS PROC PRINT documentation and the SAS LABEL statement documentation.

QUESTION NO: 4

Which two statements are true about data set options such as KEEP= and RENAME =? (Choose two.)

A. The options must be placed in parentheses.

B. The options are not allowed in PROC steps.

C. The options can be used only on the output data set.

D. The options are placed after the data set name.

ANSWER: A D

Explanation:

“The options must be placed in parentheses.” and “The options are placed after the data set name.” are true because SAS data set options use the syntax `SAS-data-set-name (data-set-options)`. The parenthesized option list is directly associated with the data set reference, whether that reference identifies an input data set being read or an output data set being created. For example, `set sales(keep=id amount);` instructs SAS to read only `id` and `amount` from `sales`. Similarly, `data summary(rename=(oldname=newname));` applies the RENAME= option to the output data set named `summary`. Multiple data set options can be included in the same parentheses when needed, such as `data work.new(keep=id total rename=(total=amount));`. This syntax lets SAS distinguish data set options from DATA-step statements and procedure options, and it makes clear precisely which data set is affected. SAS documents data set options as options specified in parentheses following a SAS data set name; their availability depends on the context in which that data set is referenced. See the official SAS data set reference documentation and SAS Statements: Reference.

QUESTION NO: 5

Which statement is true regarding the DATA step?

A. The DATA step can only read raw data files.

B. The DATA step reads, processes, and creates data.

C. The DATA step can output only one data set.

D. The DATA step must be the first step in a program.

ANSWER: B

Explanation:

The DATA step reads, processes, and creates data is correct. In SAS, the DATA step is the primary programming construct for building and manipulating SAS data sets. It can read observations from existing SAS data sets or from external/raw data files, then apply programming statements to calculate values, recode variables, subset observations, combine sources, and otherwise transform the data. When the step executes, SAS processes observations iteratively using the program data vector and writes the resulting observations to the output SAS data set named in the DATA statement.

This read-process-create model is a core SAS programming concept: input supplies data to the step, DATA-step statements perform the required processing, and output produces one or more SAS data sets. The wording must say *data*, rather than "date," because the DATA step handles data values and data sets generally; it is not limited to date values. SAS documentation describes the DATA step as the component used to read and manipulate data and create SAS data sets. See the official SAS Language Reference: Concepts and the SAS 9.4 documentation for DATA-step concepts and processing behavior.

QUESTION NO: 6 - (SIMULATION)

____ steps typically report, manage, or analyze data.

Enter your answer in the space above. Case is ignored.

ANSWER: See the explanation for the answer

Explanation:

Answer: `PROC`

`PROC` steps (procedure steps) typically report, manage, or analyze data. In contrast, `DATA` steps are generally used to create or modify SAS data sets.

QUESTION NO: 7

Which two correctly create a SAS date value? (Choose two.)

A. "10/19/2019"d

B. mdy (10, 19, 2019)

C. "19Oct2019"d

D. mdy (19, Oct, 2019)

ANSWER: B C

Explanation:

`mdy (10, 19, 2019)` correctly creates a SAS date value by using the MDY function. MDY accepts numeric month, day, and year arguments and returns the numeric count of days from 01 January 1960, which is SAS's internal representation of a date. The result can subsequently be displayed in a readable form by applying a date format, such as DATE9. or MMDDYY10.

`"19Oct2019"d` correctly creates a SAS date constant. A SAS date constant is a quoted value followed immediately by `d`; its date text uses the day, a three-letter month abbreviation, and a two- or four-digit year. SAS converts this literal directly to its internal numeric date value when compiling the statement. Date constants are useful when a known, fixed date is needed in

an expression, comparison, assignment, or WHERE condition. The corrected month text is `Oct`, using the letter `O`, rather than the digit zero.

For SAS documentation on date constants and their required form, see SAS Language Reference: Date, Time, and Datetime Constants. For the function's argument rules and returned value, see SAS MDY Function documentation.

QUESTION NO: 8

Which LABEL statement has correct syntax?

A. Label FName=' First Name' ;LName =; Last Name' ;

B. Label FName='First Name' LName='Last Name';

C. Label FName=' First Name' andLName =' Last Name' ;

D. Label FName=' First Name' ,LName =; Last Name' ;

ANSWER: B

Explanation:

`Label FName='First Name' LName='Last Name';` has correct SAS syntax. A LABEL statement begins with the `LABEL` keyword and is followed by one or more variable-label assignments. Each assignment uses a valid SAS variable name, an equals sign, and a quoted character string containing the descriptive label. Multiple assignments may appear in the same LABEL statement, separated by whitespace; a semicolon terminates the complete statement. The labels in this statement associate `First Name` with `FName` and `Last Name` with `LName`. This statement can be used in a DATA step to permanently store descriptive labels in the output data set's metadata. SAS permits either single or double straight quotation marks for character strings, provided that the opening and closing quotation marks match. The option text has been normalized to use straight single quotation marks, which are valid SAS delimiters. For the formal syntax and usage of LABEL statements, see the SAS LABEL Statement documentation. Additional examples of assigning permanent variable labels in a DATA step are available in the SAS DATA step documentation.

QUESTION NO: 9

Which statement is true about the SUM statement?

A. The SUM statement includes an equal sign.

B. Multiple accumulating variables can be created with one SUM statement.

C. The SUM statement ignores missing values.

D. By default, the accumulating variable is initialized to 1.

ANSWER: C

Explanation:

The SUM statement ignores missing values. In a DATA step, a SUM statement uses the form `variable + expression;` and maintains a running total across observations. Its behavior is specifically designed for accumulation: SAS automatically retains the accumulating variable from one iteration of the DATA step to the next and initializes it to zero before processing begins. When the expression on the right side evaluates to a missing numeric value, the SUM statement treats that value as zero for purposes of the running total. Thus, a missing value does not cause the accumulated result to become missing; processing continues with the total unchanged for that observation.

For example, with `total + amount;`, an observation whose `amount` is missing leaves `total` at its previously accumulated value. This differs from ordinary arithmetic addition, where adding a missing value generally produces a missing result. The special missing-value handling, automatic retention, and zero initialization are defining characteristics of

the SUM statement and make it useful for cumulative totals. SAS documents this behavior in the SUM Statement reference and the SUM function reference.