Quiz Score: 0%

*You answered **0 out of 50** questions correctly. To see any answer, scroll down or click a question in the grid below. When you select links on this page, the information appears in the original browser window.*

Question #	1	2	3	4	5	6	7	8	9	10
Correct / Incorrect	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	11	12	13	14	15	16	17	18	19	20
	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	21	22	23	24	25	26	27	28	29	30
	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	31	32	33	34	35	36	37	38	39	40
	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	41	42	43	44	45	46	47	48	49	50
	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

✗ **1.** Given the following SAS data sets **One** and **Two**:

One

Year	Qtr	Budget
2001	3	500
2001	4	400
2002	1	700

Two

Year	Qtr	Sales
2001	4	300
2002	1	600

The following SAS program is submitted:

```
proc sql;
   select one.*, sales
      from one, two
      where one.year=two.year and
            one.qtr=two.qtr;
quit;
```

Which of the following reports is generated?

a.

Year	Qtr	Budget	Sales
2001	3	500	300

2002	1	700	600

b.

Year	Qtr	Budget	Sales
2001	4	400	300
2002	1	700	600

c.

Year	Qtr	Budget	Sales
2001	3	500	.
2001	4	400	300
2002	1	700	600

d.

Year	Qtr	Budget	Sales
2001	3	500	300
2001	4	400	300
2002	1	700	300
2001	3	500	600
2001	4	400	600
2002	1	700	600

Correct answer: b
Your answer:

A PROC SQL inner join combines and displays only the rows from the first table that match rows from the second table, based on the matching criteria (join conditions) that are specified in the WHERE clause. In this example, there are two rows that have matching values of `Year` and `Qtr` across the two tables.

You can learn about using the SQL procedure for inner joins in **Combining Tables Horizontally Using PROC SQL**.

---

2. Given the following SAS data sets **One** and **Two**:

One

Year	Qtr	Budget
2001	3	500
2001	4	400
2002	1	700

Two

Year	Qtr	Sales
2001	4	300
2002	1	600

The following SAS program is submitted:

```
proc sql;
   select one.*, sales
   from one right join two
   on one.year=two.year;
quit;
```

Which one of the following reports is generated?

a.

Year	Qtr	Budget	Sales
2001	3	500	.

b.

Year	Qtr	Budget	Sales
2001	4	400	300
2002	1	700	600

c.

Year	Qtr	Budget	Sales
2001	3	500	.
2001	4	400	300
2002	1	700	600

d.

Year	Qtr	Budget	Sales
2001	3	500	300
2001	4	400	300
2002	1	700	600

Correct answer: d
Your answer:

A PROC SQL right outer join combines and displays all rows that match across tables, based on the specified matching criteria (join conditions), plus non-matching rows from the right table (the second table specified in the FROM clause). In this example, there are two rows that have matching values of `Year` across the two tables. There are no non-matching rows in the second table, **Two**.

You can learn about using the SQL procedure for right outer joins in **Combining Tables Horizontally Using PROC SQL**.

---

3. Which one of the following programs is syntactically correct?

a.
```
proc sql;
   create table forecast as
      select a.*, b.sales
         from actual a, budget b
         where a.dept=b.dept and
               a.month=b.month;
quit;
```

b.
```
proc sql;
   create table forecast as
      select a.* b.sales
         from actual as a, budget as b
         where a.dept=b.dept and
               a.month=b.month;
quit;
```

c.
```
proc sql;
```

```
      create table forecast as
         select a.*, b.sales
            from actual as a, budget as b
            where dept=dept and
                  month=month;
   quit;
```

d.
```
   proc sql;
      create table forecast as
         select a.*, b.sales
            from actual, budget
            where a.dept=b.dept and
                  a.month=b.month;
   quit;
```

Correct answer: **a**
Your answer:

This program must use qualified table names in the SELECT and WHERE clauses to distinguish between the two tables that are being joined. To make the query easier to read, a table alias (a temporary, alternate name for a table) is used in each qualified table name instead of the full table name. A table alias must be specified in the FROM clause; the keyword AS is optional. In answers *c* and *d* above, the syntactical error is related to the use of table aliases. In answer *b*, the syntactical error is a missing comma in the SELECT clause.

You can learn about

- table aliases in **Combining Tables Horizontally Using PROC SQL**
- the SELECT clause in **Performing Advanced Queries Using PROC SQL**.

---

4. Given the following SAS data sets **One** and **Two**:

One

Num	Var1
1	A
1	A
2	B
3	C

Two

Num	Var2
1	A
4	Y
4	Z

The following SAS program is submitted:

```
proc sql;
   select *
      from one
   except
   select *
      from two;
quit;
```

Which of the following reports is generated?

a.

Num
2
3

b.

Num	Var1
2	B
3	C

c.

Num	Var1
1	A
2	B
3	C

d.

Num	Var2
1	A
4	Y
4	Z

Correct answer: b

Your answer:

The set operator EXCEPT selects unique rows from the first table (the table specified in the first query) that are **not** found in the second table (the table specified in the second query). PROC SQL overlays columns based on their relative position in the SELECT clause and uses the column names from the first table (the table referenced in the first query). In this example, the result set contains the third and fourth rows in table **One**, which are the only unique rows that are not found in table **Two**.

You can learn about the EXCEPT set operator in **Combining Tables Vertically Using PROC SQL**.

---

5. Given the following SAS data sets **One** and **Two**:

One

Num	Var1
1	A
1	A
2	B
3	C

Two

Num	Var2
1	A
4	Y
4	Z

The following SAS program is submitted:

```
proc sql;
   select *
      from one
```

```
   intersect all
   select *
      from two;
quit;
```

Which of the following reports is generated?

a.

Num
1

b.

Num
1
2

c.

Num	Var1
1	A

d.

Num	Var1
1	A
1	A

Correct answer: c
Your answer:

The set operator INTERSECT with the keyword ALL selects all rows that are common to both tables, and overlays columns. In this example, the result set contains the one row that is common across the two tables. Although table **One** also contains a duplicate of the common row, there is no matching duplicate row in table **Two**, so the duplicate row is not included in the result set.

You can learn about the INTERSECT set operator in **Combining Tables Vertically Using PROC SQL**.

---

6. Given the following SAS data sets **One** and **Two**:

One

Num	Product
1	Computer
2	Printer
3	Monitor
4	Keyboard
5	Modem

Two

Num	Product
1	Portable
5	External

The following SAS program is submitted:

```
proc sql;
```

```
      select product
         from one
         where exists
            (select *
               from two
               where one.num=two.num);
quit;
```

Which one of the following reports is generated?

a.

Product
Computer
Modem

b.

Product
Computer
Printer

c.

Product
Printer
Monitor
Keyboard

d.

Product
Computer
Printer
Monitor
Keyboard
Modem

Correct answer: a
Your answer:

A correlated subquery requires one or more values to be passed to it by the outer query before the subquery can return a value to the outer query. In this example, the outer query passes each value of `One.Num` to the subquery, and the subquery compares the value of `One.Num` to the value of `Two.Num`. The report displays the values of `Product` for the two rows that have matching values of `One.Num` and `Two.Num`.

You can learn about correlated subqueries in **Performing Advanced Queries Using PROC SQL**.

---

7. How many columns can an SQL procedure subquery within a WHERE or HAVING clause return to the outer query?

a. 0
b. 1
c. 2

d. the same number of columns that are in the table

Correct answer: b
Your answer:

A subquery in the WHERE or HAVING clause can return values for multiple rows but only for a single column.

You can learn about using a subquery in a WHERE or HAVING clause in **Performing Advanced Queries Using PROC SQL**.

---

8. Given the following SAS data sets **One** and **Two**:

One

Num	Char1
1	A
2	B
4	D

Two

Num	Char2
2	X
3	Y
5	V

The following SAS program is submitted creating the output table **Three**:

```
data three;
   merge one (in=in1) two (in=in2);
   by num;
   if in2;
run;
```

Three

Num	Char1	Char2
2	B	X
3		Y
5		V

Which of the following SQL programs creates an equivalent SAS data set **Three**?

a.
```
proc sql;
   create table three as
      select two.num, char1, char2
         from one left join two
         on one.num=two.num;
quit;
```

b.
```
proc sql;
   create table three as
      select two.num, char1, char2
         from one right join two
         on one.num=two.num;
```

```
quit;
```

c.
```
proc sql;
   create table three as
      select two.num, char1, char2
         from one full join two
         on one.num=two.num;
quit;
```

d.
```
proc sql;
   create table three as
      select two.num, char1, char2
         from one, two
         where one.num=two.num;
quit;
```

Correct answer: **b**
Your answer:

In the DATA step shown, the use of only the IN= variable in the IF statement specifies that only matching observations are taken from the first data set, while all observations are taken from the second data set. Similarly, a right outer join retrieves all rows that match across tables, based on the specified matching criteria (join conditions), plus non-matching rows from the right table (the second table specified in the FROM clause).

You can learn about

- using the temporary IN= variables in a DATA step merge in **Combining Data Horizontally**
- using the SQL procedure to perform a right outer join in **Combining Tables Horizontally Using PROC SQL**.

---

9. Table **One** has five million observations. Table **Two** has one thousand observations. These tables have identical column attributes. Concatenating tables **One** and **Two** should result in 5,001,000 observations.

Which one of the following SAS techniques uses the least CPU time and I/O operations to process?

a. the APPEND procedure
b. the SET statement in the DATA step
c. the INSERT INTO statement in the SQL procedure
d. the OUTER UNION CORR operator in the SQL procedure

Correct answer: **a**
Your answer:

Both the SET statement in the DATA step and the OUTER UNION CORR operator in the SQL procedure read all observations from all input data sets, while the APPEND procedure reads only the observations in the appended data set. Thus, I/O and CPU usage are both greater with either the SET statement or the OUTER UNION CORR statement than with the APPEND procedure. While the INSERT INTO statement reads only the observations in the inserted data set, just as the APPEND procedure does, the SQL procedure in general requires more procedure code and therefore more I/O and CPU usage than the APPEND procedure.

You can learn about

- using the SET statement to concatenate SAS data sets, as well as the APPEND procedure in **Combining Data Vertically**
- the OUTER UNION CORR operator in the SQL procedure in **Combining Tables Vertically Using PROC SQL**
- the INSERT statement in the SQL procedure in **Creating and Managing Tables Using PROC SQL**.

---

10. The following SAS program is submitted:

```
proc sql noprint;
   select name
      into : info separated by ' '
      from dictionary.columns
      where libname="SASHELP" and
            memname="CLASS";
quit;
```

Given that a SAS data set **Sashelp.Class** exists, which one of the following is generated by the above program?

a. a list of names
b. a syntax error in the log
c. a macro variable called `info`
d. a report showing metadata information

Correct answer: **c**
Your answer:

The INTO clause in a SELECT statement enables you to create or update macro variables during execution of a PROC SQL step. In this example, all of the values of the data set variable `name` are concatenated (and separated by spaces) to form the value of the new macro variable `info`.

You can learn about using the INTO clause in a SELECT statement to create macro variables in a PROC SQL step in **Processing Macro Variables at Execution Time**.

---

11. The following SAS program is submitted:

```
%let lib=SASUSER;
%let dsn=CLASS;
proc sql;
   select name
      from dictionary.columns
         where libname="&lib" and
               <insert code here>
quit;
```

Which one of the following conditions completes the WHERE clause to create a subset from the data set **Dictionary.Columns** about **Sasuser.Class**?

a. `table="&dsn"`
b. `name="&dsn"`
c. `member="&dsn"`
d. `memname="&dsn"`

Correct answer: **d**
Your answer:

The dictionary table **Dictionary.Columns** stores the names of data sets in the column `memname`. The %LET statement creates a macro variable and assigns a value to it. In this example, the reference `&dsn` resolves to *CLASS*.

You can learn about

- dictionary tables in **Managing Processing Using PROC SQL**
- using the %LET statement to create macro variables in **Introducing Macro Variables**.

---

12. Which one of the following statements is true about an SQL procedure view?

a. It stores large amounts of data.
b. It updates data from multiple database tables.
c. It contains both the data and the descriptor portions.
d. It accesses the most current data in changing tables.

Correct answer: **d**
Your answer:

A PROC SQL view is a stored query that is executed when you use the view in a SAS procedure, DATA step, or function. Remember, a PROC SQL view contains only the logic for accessing the data, not the data itself.

You can learn about PROC SQL views in **Creating and Managing Views Using PROC SQL**.

---

13. The following SAS program is submitted:

```
proc sql noprint;
   select name
      into : name1 -: name19
      from one;
quit;
```

Given that table **One** has 19 observations and 5 variables, how many macro variables are created by the above program?

a. 5
b. 6
c. 19
d. 20

Correct answer: **c**
Your answer:

You can use the INTO clause to create one new macro variable for each observation that the SELECT statement of a PROC SQL step reads from the input data set. In this example, the SQL step selects each observation from table **One**, so the result is 19 new macro variables.

You can learn about using the INTO clause to create macro variables in a PROC SQL step in **Processing Macro Variables at Execution Time**.

---

14. Which one of the following SAS programs removes the index **Jobcode** from the table **Staff**?

a.
```
proc sql;
   drop index jobcode from work.staff;
quit;
```
b.
```
proc sql;
   delete index jobcode from work.staff;
quit;
```
c.
```
proc sql;
   drop index from work.staff;
quit;
```
d.
```
proc sql;
   delete index from work.staff;
quit;
```

Correct answer: **a**
Your answer:

In a PROC SQL step, you can use the DROP INDEX statement to delete an index from a table. You use the `drop index` keywords, then name the index that you want to delete. There is no DELETE INDEX statement in the SQL procedure.

You can learn about using the DROP INDEX statement to delete an index in a PROC SQL step in **Creating Samples and Indexes**.

---

15. A batch (non-interactive) job is submitted on Wednesday, 20 January 1999 (SAS date 14264). The program completes the following day.

The following code is the last step in the SAS program:

```
data _null_;
```

```
   call symput('mdate', input("&sysdate", date7.));
run;
```

Which of the following is the value of the macro variable `mdate`?

a. *14264*
b. *14265*
c. *20JAN99*
d. *21JAN99*

Correct answer: **a**
Your answer:

The SYMPUT routine enables you to create a macro variable and to assign to that variable any value that is available in the DATA step. The SYSDATE automatic macro variable records the date when the current SAS session began in DATE7. format. The INPUT statement within the CALL SYMPUT routine in this example converts the value of SYSDATE to a SAS date value.

You can learn about

- the SYMPUT routine in **Processing Macro Variables at Execution Time**
- the SYSDATE automatic macro variable in **Introducing Macro Variables**.

---

16. Given the following SAS data set **Mylib.Mydata**:

Mylib.Mydata

Name	Animal	Age
Max	Cat	9
Brown	Dog	22
Large	Pig	1

The following SAS program is submitted:

```
data _null_;
   set mylib.mydata;
   call symput('animal' || left(_n_), name);
run;
%let i=2;
title "The value is &&animal&i";
```

Which one of the following does the TITLE statement resolve to?

a. The value is Dog
b. The value is animal2
c. The value is Brown
d. The value is &animal2

Correct answer: c
Your answer:

The CALL SYMPUT statement in this step concatenates animal with the value of _n_ as the name of the macro variable, and assigns the value of the data set variable `name` to the macro variable `animal`*n*. When the macro processor encounters the indirect reference in the TITLE statement, the multiple ampersands preceding the macro variable reference resolve to one ampersand, `&i` resolves to 2, the reference `&animal2` is re-scanned and resolves to *Brown*.

You can learn about the CALL SYMPUT statement and indirect macro references in **Processing Macro Variables at Execution Time**.

---

17. The following SAS program is submitted:

```
%let sun=SHINE;
%let shine=MOON;

proc print data=weather(keep=sun shine moon);
   sum &&&&sun;
run;
```

Which one of the following variables is summed in the resulting report?

a. `sun`
b. `MOON`
c. `SHINE`
d. None, an ERROR message will be written to the log.

Correct answer: c
Your answer:

When the macro processor encounters multiple ampersands preceding a macro variable reference, it resolves two ampersands into one ampersand and re-scans the reference. In this example, the macro processor resolves the reference `&&&&sun` to `&&sun` on the first scan. Then, the macro processor re-scans and resolves the reference to `&sun`, and re-scans again, finally resolving the reference to *SHINE*.

You can learn about using multiple ampersands in a macro variable reference in **Processing Macro Variables at Execution Time**.

---

18. The following SAS program is submitted:

```
%let code=set old (keep=A B);
   Newvar=a || b;
run;

data new;
```

```
   &code
run;

proc print data=new;
run;
```

Which one of the following is the result of the above program?

a. There will be errors in the log and no report.
b. There will be no report and no errors in the log.
c. There will be a report that has only the variables `A` and `B`.
d. There will be a report that has the variables `A`, `B`, and `Newvar`.

Correct answer: a
Your answer:

The %LET statement creates a macro variable named `code` and assigns it a value of *set old (keep=A B)*. In the DATA step, `&code` resolves to *set old (keep=A B)* but the semicolon is missing after `&code`. Therefore the DATA step compiler sees *set old (keep A B) run;* and interprets **Run** as the name of a data set. Because a data set named **Run** does not exist, the step fails during compilation and the data set **New** is not created.

You can learn about using the %LET statement to create a macro variable, as well as referencing a macro variable, in **Introducing Macro Variables**.

---

19. The following SAS program is submitted:

```
data towns;
   input City $ pop year;
cards;
ANDOVER 75000 1998
CARY 80000 1998
ANDOVER 14000 1977
CARY 71000 1986
;

%macro a(location);
   data a;
      set towns;
   %if &location=UK %then %do;
      where city='ANDOVER';
   %end;
   %if &location=NC %then %do;
      where city='CARY';
   %end;
   if pop>70000;
   run;
%mend;
%a(UK)
```

Which one of the following represents the resulting output in the SAS data set **A**?

a. 1 observation with the value of *CARY* for the variable `City`
b. 2 observations with the value of *CARY* for the variable `City`

c. 1 observation with the value of *ANDOVER* for the variable `City`

d. 2 observations with the value of *ANDOVER* for the variable `City`

Correct answer: **c**
Your answer:

You can use %DO and %END statements in conjunction with a %IF-%THEN statement in order to conditionally process a section of a macro program. In this example, the WHERE statement is processed only when the value of the `location` macro variable is *UK*. Because the value of `location` is set by the positional parameter, its value is *UK* when **A** executes, and the WHERE statement is processed.

You can learn about using %DO and %END statements in conjunction with a %IF-%THEN statement, as well as using positional parameters in a macro definition, in **Creating and Using Macro Programs**.

---

20. The following SAS program is submitted:

```
%let a=cat;

%macro animal;
   %let a=dog;
%mend;
%animal
%put a is &a;
```

Which one of the following is written to the SAS log?

a. a is

b. a is &a

c. a is cat

d. a is dog

Correct answer: **d**
Your answer:

The first %LET statement creates a macro variable named a in the global symbol table and assigns a value of *cat* to it. When the **Animal** macro executes, the macro processor encounters the second %LET statement. The macro processor checks the symbol table that is local to **Animal** for a macro variable named a. Since the local symbol table does not contain a macro variable named a, the macro processor checks the global symbol table for a macro variable named a. The macro variable a does exist in the global symbol table, and the macro processor updates its value to *dog*. When the %PUT statement executes, the macro processor resolves the value of the macro variable reference to *dog*.

You can learn about

- the global symbol table and local symbol tables in **Creating and Using Macro Programs**
- the %PUT statement in **Introducing Macro Variables**.

21. The following SAS program is submitted at the start of a new SAS session:

```
%macro trans;
   %let type=Airplane;
   proc print data=sasuser.activities;
      where trans="&type";
   run;
   %location(Automobile)
   %put type is &type;
%mend;

%macro location(type);
   data _null_;
      call symput('type', 'Train');
   run;
%mend;
%trans
```

Which one of the following is written to the log by the %PUT statement?

a. type is &type
b. type is Train
c. type is Airplane
d. type is Automobile

Correct answer: **c**
Your answer:

The **Location** macro is nested inside the **Trans** macro. The %LET statement in the **Trans** macro creates a macro named `type` in the local symbol table. When the nested **Location** macro executes, the macro processor creates a new macro variable named `type` in the symbol table that is local to **Location** because `type` is a positional parameter in the definition of **Location**. Therefore, when **Location** finishes and the %PUT statement in **Trans** executes, the macro processor looks up the value for `type` in the symbol table that is local to **Trans**, and returns *Airplane*.

You can learn about nested macros, local symbol tables, and positional parameters in **Creating and Using Macro Programs**.

▲

---

22. The following SAS program is submitted:

```
%let day=Tuesday;
%macro loop(day);
   %do day=2 %to 6;
      proc print data=sasuser.houses;
         where sellday="&day";
      run;
   %end;
   %put day is &day;
%mend;
```

```
%loop(Friday)
```

Which one of the following is written to the log by the %PUT statement?

a. day is 7
b. day is &day
c. day is Friday
d. day is Tuesday

Correct answer: **a**
Your answer:

The %LET statement creates a macro variable named `day` in the global symbol table and assigns a value of *Tuesday* to it. When the **Loop** macro executes, the parameter in the macro definition creates a new macro variable in the local symbol table for the **Loop** macro and assigns a value of *Friday* to it. Then, the %DO statement uses the local `day` macro variable as the index variable and assigns a value of *2* to it. As the %DO loop iterates, the value of the local `day` increases by *1* each time. When the value of the local `day` reaches *7*, the condition on the %DO loop fails and the %PUT statement executes, writing the value of the local `day` to the SAS log.

You can learn about

- the global symbol table, as well as using parameters and the %DO statement in a macro definition, in **Creating and Using Macro Programs**
- the %PUT statement in **Introducing Macro Variables** .

▲

---

23. Which of the following SAS functions retrieves the value of a macro variable during DATA step execution?

a. GET
b. SYMGET
c. %SYMGET
d. &MVARNAME

Correct answer: **b**
Your answer:

The SYMGET function enables you to obtain a macro variable's current value during DATA step execution.

You can learn about the SYMGET function in **Processing Macro Variables at Execution Time**.

▲

---

24. The following SAS program is submitted:

```
%let type=RANCH;
```

```
proc sql;
   create view houses as
      select *
         from sasuser.houses
            where style="&type";
quit;
%let type=CONDO;
proc print data=houses;
run;
```

The report that is produced displays observations whose value of `Style` are all equal to *RANCH*. Which one of the following functions on the WHERE clause resolves the current value of the macro variable `type`?

a. GET
b. SYMGET
c. %SYMGET
d. &RETRIEVE

Correct answer: **b**
Your answer:

The macro variable reference `&type` is resolved during creation of the PROC SQL view **Houses**, resulting in a constant value of *RANCH* for `type` whenever the view is used. In order to resolve the macro variable reference in the WHERE statement to the current value of `type` whenever the **Houses** view is used, you use the SYMGET function to obtain the value of `type`.

You can learn about

- using macro variables with PROC SQL views in **Creating and Using Macro Programs**
- the SYMGET function in **Processing Macro Variables at Execution Time**.

---

25. Which one of the following is the purpose of the %STR function?

a. to extract part of a macro character value
b. to treat semicolons and other selected tokens as text
c. to concatenate the values of two macro variables into one
d. to extract part of a macro variable based on matching criteria

Correct answer: **b**
Your answer:

The %STR function hides the normal meaning of a semicolon and other special tokens and mnemonic equivalents of comparison or logical operators so that they appear as constant text to the macro processor.

You can learn about the %STR function in **Introducing Macro Variables**.

26. The following SAS program is submitted:

```
%let mvar=bye;
data _null_;
   var1='hi';
   var2='hello';
   call symput('var1' || "&mvar", var2);
run;
```

Which one of the following is the name of the macro variable created by the CALL SYMPUT statement?

a. `var1`
b. `hibye`
c. `var1bye`
d. No macro variable is created because the CALL SYMPUT statement is not valid.

Correct answer: **c**
Your answer:

The CALL SYMPUT statement in this example uses the concatenation operator to join the text string `var1` with the resolved value of the macro variable reference `&mvar`, which is *bye*. Therefore, the name of the new macro variable is `var1bye`.

You can learn about the CALL SYMPUT statement in **Processing Macro Variables at Execution Time**.

---

27. Given the following partial SAS log:

```
11   %macro a(location);
12     %if &location = FL %then
13        libname &location 'c:\state';
14   %mend;
15   %a(FL)
NOTE: SCL source line.
16   libname mylib 'c:\mydata';
     -------
     22
ERROR: Libname FL is not assigned.
ERROR: Error in the LIBNAME statement.
ERROR 22-7: Invalid option name LIBNAME.
```

Which of the following SAS system options can be used to display the results of resolving the macro variable reference `&location`?

a. MPRINT only

b. MLOGIC only
c. SYMBOLGEN only
d. MPRINT and MLOGIC

Correct answer: c
Your answer:

The MLOGIC option prints messages that trace macro execution to the SAS log. The MPRINT option prints the text that is sent to the compiler after all macro resolution has taken place. The SYMBOLGEN option writes the resolved value of macro variables to the SAS log.

You can learn about

- the MLOGIC option and the MPRINT option in **Creating and Using Macro Programs**
- the SYMBOLGEN option in **Introducing Macro Variables**.

---

28. Which one of the following statements displays all user-defined macro variables in the SAS log?

a. `%put user=;`
b. `%put user;`
c. `%put _user_;`
d. `options mprint`

Correct answer: c
Your answer:

To display all user-defined macro variables and their values in the SAS log, you use the %PUT statement with the _USER_ argument. The MPRINT option specifies that text that is sent to the compiler as a result of macro execution is printed in the SAS log.

You can learn about the _USER_ argument and the %PUT statement in **Introducing Macro Variables**.

---

29. Given the following SAS log:

```
55   %let lib = sasuser.;
56   %let dsn = class;
57   %put dsn is &dsn, and lib is &lib;
dsn is class, and lib is sasuser.
58   <insert statement here>
59   %put dsn is &dsn, and lib is &lib;
WARNING: Apparent symbolic reference DSN not resolved.
dsn is &dsn, and lib is sasuser.
```

Which one of the following was used in the code shown above to remove the macro variable `dsn`

from the symbol table?

a. `%SYMDEL dsn;`
b. `%REMOVE dsn;`
c. `%DELETE dsn;`
d. `%MACDEL dsn;`

Correct answer: a
Your answer:

You use the %SYMDEL statement to delete a macro variable from the global symbol table during a SAS session.

You can learn about the macro variables in the global symbol table and the %SYMDEL statement in **Creating and Using Macro Programs**.

---

30. When using the KEY= option in the SET statement in the DATA step, which one of the following is the purpose of the automatic variable `_IORC_`?

a. to determine the amount of I/O usage
b. to determine the success of the I/O operation
c. to determine the length of the master observation
d. to determine the location of the master observation

Correct answer: b
Your answer:

SAS creates the automatic variable _IORC_ whenever you use the KEY= option in the DATA step. If the value of _IORC_ is zero, the I/O function was successful. If the value of _IORC_ is not zero, the I/O function was not successful.

You can learn about the automatic variable _IORC_ in **Combining Data Horizontally**.

---

31. Which one of the following statements is true regarding the TRANSPOSE procedure?

a. The TRANSPOSE procedure produces printed output by default.
b. The TRANSPOSE procedure rearranges all variables by default.
c. The TRANSPOSE procedure does not create an output data set by default.
d. The TRANSPOSE procedure creates one observation for each restructured variable for each BY group.

Correct answer: d
Your answer:

The TRANSPOSE procedure creates an output data set by restructuring the values in a SAS data

set. When the data set is restructured, selected variables are transposed into observations.

You can learn about the TRANSPOSE procedure in **Using Lookup Tables to Match Data**.

---

32. Which one of the following is true with respect to when a DATA step with a MERGE statement successfully combines data in SAS?

a. when the data sources are indexed SAS data sets
b. when the data sources are unsorted raw data files
c. when the data sources are unsorted, unindexed SAS data sets
d. when not all the data sources contain the same BY variable(s)

Correct answer: a
Your answer:

You can use a DATA step with a MERGE statement to combine data sets that have a common BY variable. In order for the MERGE statement to produce expected results, your input data sets must be either sorted by or indexed on the common BY variable. You cannot use the MERGE statement to combine raw data files.

You can learn about using the MERGE statement in a DATA step in **Combining Data Horizontally**.

---

33. Which one of the following statements is true regarding compressed SAS data sets?

a. Deleted observation space is reused by default.
b. Observations are addressable by observation number.
c. More input operations are necessary to read from the data set during processing.
d. More disk space is usually required to store compressed data sets than to store uncompressed data sets.

Correct answer: b
Your answer:

The REUSE= option controls the reuse of deleted observations, and the default value for REUSE= is NO. Because input operations are a measure of how many pages must be read from the SAS data set, and because a compressed SAS data file occupies fewer data set pages than the same data set in uncompressed form, input operations are reduced when reading a compressed SAS data set. Also, less disk space is required to store a compressed SAS data set than the equivalent uncompressed SAS data set.

You can learn about the REUSE= option and compressed SAS data files in **Controlling Data Storage Space**.

34. Which one of the following statements is true regarding the use of a SAS numeric variable of less than 8 bytes?

a. It requires no additional processing when reading the SAS data set.
b. It might reduce the precision when the value is written to the SAS data set.
c. It has an effect on how the values are stored in the program data vector (PDV).
d. It has no effect on the precision of the largest number that can be accurately stored.

Correct answer: **b**
Your answer:

Numeric variables of less than 8 bytes are expanded to 8 bytes by padding with binary zeros in the program data vector and when they are used by SAS procedures; therefore, reduced-length numeric variables do require additional processing. Also, the precision of the largest number that can be accurately stored is directly related to the number of bytes of storage.

You can learn about the use of numeric variables of less than 8 bytes in **Controlling Data Storage Space**.

---

35. Which one of the following programs creates a SAS data view named **Sasuser.Ranch**?

a.
```
data _null_ view=sasuser.ranch;
   set sasuser.houses;
   where style='RANCH';
run;
```
b.
```
data _null_ / view=sasuser.ranch;
   set sasuser.houses;
   where style='RANCH';
run;
```
c.
```
data sasuser.ranch view=sasuser.ranch;
   set sasuser.houses;
   where style='RANCH';
run;
```
d.
```
data sasuser.ranch / view=sasuser.ranch;
   set sasuser.houses;
   where style='RANCH';
run;
```

Correct answer: **d**
Your answer:

The name of the data set and the data view must match. In addition, a slash is required between the name(s) of the data set(s) being created and the VIEW= option.

You can learn about SAS data views and the VIEW= option in **Controlling Data Storage Space**.

36. By default, in which one of the following SAS catalogs are user-defined formats stored?

a. **Work.Format**
b. **Work.Formats**
c. **Sasuser.Format**
d. **Sasuser.Formats**

Correct answer: b
Your answer:

When the LIB= option is not specified in the FORMAT procedure, formats are stored in the **Work.Formats** catalog and exist for the duration of the SAS session.

You can learn about the FORMAT procedure in **Formatting Data**.

---

37. Which one of the following FORMAT procedure options is used to create a format from the values in a SAS data set?

a. FMTLIB
b. INVALUE
c. CNTLIN=
d. CNTLOUT=

Correct answer: c
Your answer:

You use the CNTLIN= option to read and create a format from a SAS data set that contains value information. The CNTLOUT= option enables you to create a SAS data set from a format. The FMTLIB keyword displays a list of all formats for a specified catalog. INVALUE is not a valid option in the FORMAT procedure.

You can learn about PROC FORMAT and the CNTLIN= option in **Formatting Data**.

---

38. Which one of the following FORMAT procedure options is used to create a SAS data set from an existing format?

a. FMTLIB
b. CREATE
c. CNTLIN=
d. CNTLOUT=

Correct answer: d
Your answer:

You use the CNTLOUT= option to create a SAS data set from a format. The FMTLIB keyword displays a list of all formats for a specified catalog. The CNTLIN= option enables you to create a format from a SAS data set. CREATE is not a valid option in the FORMAT procedure.

You can learn about the PROC FORMAT and the CNTLOUT= option in **Formatting Data**.

---

39. Which one of the following represents the resource(s) saved by the BUFSIZE= and BUFNO= options?

a. I/O only
b. CPU only
c. disk space only
d. I/O, CPU, and disk space

Correct answer: a
Your answer:

While there are some CPU resources saved with the BUFSIZE= and the BUFNO= options, the primary resource conserved is I/O. The BUFSIZE= option and the BUFNO= option have no effect on disk space.

You can learn about the BUFSIZE= option and the BUFNO= option in **Controlling Memory Usage**.

---

40. Which one of the following represents the resource(s) conserved by the SASFILE statement if the entire SAS data set fits in memory?

a. I/O only
b. CPU only
c. memory only
d. CPU and I/O

Correct answer: d
Your answer:

Using the SASFILE statement conserves I/O if the entire data set fits in memory. The SASFILE statement also saves CPU resources for every input/output operation that is not performed to access the data. Using the SASFILE statement increases the memory used, because the entire data set is read into memory at once.

You can learn about the SASFILE statement in **Controlling Memory Usage**.

41. Which one of the following represents where the input is measured when using SAS data sets as input to a DATA step?

a. between the storage on disk and utility swap files
b. between the storage on disk and the input buffers
c. between the input buffers and the program data vector (PDV)
d. between the program data vector (PDV) and the output buffers

Correct answer: **b**
Your answer:

Input is measured between the physical storage location and the input buffers in memory.

You can learn about how input is measured in SAS in **Controlling Memory Usage**.

42. The SAS data set **One** consists of five million observations and has 25 variables. Which one of the following SAS programs requires the least CPU time to be processed?

a.
```
data two;
   set one;
   if year(date)=1999 and state in
      ('NY' 'NJ' 'CT') then
      sales=sales98*1.1;
   else if year(date)=2000 and state in
      ('NY' 'NJ' 'CT') then
      sales=sales98*1.15;
   else if year(date)=2001 and state in
      ('NY' 'NJ' 'CT') then
      sales=sales98*1.2;
run;
```

b.
```
data two;
   set one;
   if year(date)=1999 and state in
      ('NY' 'NJ' 'CT') then
      sales=sales98*1.1;
   if year(date)=2000 and state in
      ('NY' 'NJ' 'CT') then
      sales=sales98*1.15;
   if year(date)=2001 and state in
      ('NY' 'NJ' 'CT') then
      sales=sales98*1.2;
run;
```

c.
```
data two;
   set one;
   if state in ('NY' 'NJ' 'CT')then do;
      year=year(date);
   if year=1999 and state in('NY' 'NJ' 'CT') then
      sales=sales98*1.1;
   if year=2000 and state in('NY' 'NJ' 'CT') then
      sales=sales98*1.15;
   if year=2001 and state in('NY' 'NJ' 'CT') then
```

```
         sales=sales98*1.2
      end;
   run;
```

d.
```
   data two;
      set one;
      if state in('NY' 'NJ' 'CT') then do;
         year=year(date);
         select(year);
            when(1999) sales=sales98*1.1;
            when(2000) sales=sales98*1.15;
            when(2001) sales=sales98*1.2;
            otherwise;
         end;
      end;
   run;
```

Correct answer: **d**
Your answer:

Answer *a* is not as efficient as possible because the YEAR function is invoked and the value for the `state` variable is tested a minimum of one time and a maximum of three times for each observation that is read from the data set **One**. Answer *b* is not as efficient as possible because the YEAR function is invoked and the value for `state` is tested three times for each observation that is read from the data set **One**. In answer *c*, the YEAR function is invoked only once, but the values for `year` and `state` are tested three times for each observation that is read from the data set **One**. Answer *d* is the most efficient because the YEAR function is invoked only once and the value of `state` is checked only once.

You can learn about efficient use of the IF-THEN statement, the IF-THEN/ELSE statement. and the SELECT statement in **Utilizing Best Practices**.

---

43. The SAS data set **Temp** contains 1,000 observations and 10 variables. Which one of the following SAS DATA steps writes the variables a, b, and c to both SAS data sets **One** and **Two**?

a.
```
   data one two(keep=a b c);
      set temp(keep=a b c d);
   run;
```

b.
```
   data one two;
      set temp(keep=a b c);
   run;
```

c.
```
   data one(keep=a b c) two;
      set temp(keep=a b c d);
   run;
```

d.
```
   data one(keep=a b c) two(keep=a c);
      set temp(keep=a b c);
   run;
```

Correct answer: **b**
Your answer:

The program in answer *a* writes a, b, c, d to **One** and a, b, c to **Two**. The program in answer *b* writes a, b, c to both data sets. The program in answer *c* writes a, b, c to **One** and a, b, c, d to

Two. The program in answer *d* writes a, b, c to **One** and a, c to **Two**.

You can learn about using the KEEP= data set option in the DATA statement versus using it in the SET statement in **Utilizing Best Practices**.

---

44. Which one of the following statements is true?

a. A WHERE statement can select observations only from SAS data sets.
b. Efficiency is affected by the placement of the WHERE statement in the DATA step.
c. The FIRSTOBS= and OBS= data set options are never allowed in a step with a WHERE expression.
d. There is a significant efficiency difference between a WHERE statement and a WHERE= data set option in the input data set.

Correct answer: **a**
Your answer:

The placement of the WHERE statement in the DATA step has no effect on efficiency. The WHERE statement is always processed between the input buffers in memory and the program data vector (PDV). Beginning in SAS 8.1, the FIRSTOBS= and OBS= data set options can be used in a step with a WHERE expression. There is no difference in efficiency between a WHERE statement and a WHERE= data set option in the input data set.

You can learn about how efficiency relates to WHERE statements and the WHERE= data set option in **Utilizing Best Practices**.

---

45. The following SAS program is submitted:

```
data one;
   do pt=1 to 14 by 2;
      set temp point=pt
               nobs=totobs;
      output;
   end;
   stop;
run;
```

The SAS data set **Temp** has 2,500,000 observations. Which one of the following represents the observations included in the data set **One**?

a. the first 14 observations from **Temp**
b. all 2,500,000 observations from **Temp**
c. a random sample of the observations from **Temp**
d. a systematic sample of the observations from **Temp**

Correct answer: d
Your answer:

One will contain 7 observations that are taken from regular intervals in **Temp** by using the POINT= option in the SET statement. Because the observations are taken from regular intervals, the sample is systematic rather than random.

You can learn about the POINT= option, random samples, and systematic samples in **Creating Samples and Indexes**.

---

46. Which one of the following SAS SORT procedure options eliminates observations with identical BY values?

a. NODUP
b. UNIQUE
c. DISCRETE
d. NODUPKEY

Correct answer: d
Your answer:

NODUPKEY is the correct answer. The NODUP option of the SORT procedure eliminates consecutive duplicate observations in which all variable values are identical between observations. UNIQUE and DISCRETE are not valid options for the SORT procedure.

You can learn about the SORT procedure and the NODUPKEY option in **Selecting Efficient Sorting Strategies**.

---

47. The SAS data set **Sasdata.Sales** has a simple index for the variable `Date` and a variable named `Revenue` with no index. In which one of the following SAS programs is the **Date** index considered for use?

a.
```
proc print data=sasdata.sales;
   by date;
run;
```
b.
```
proc print data=sasdata.sales;
   where month(date)=3;
run;
```
c.
```
data march;
   set sasdata.sales;
   if '01mar2002'd < date < '31mar2002'd;
run;
```
d.
```
data march;
   set sasdata.sales;
   where date < '31mar2002'd or revenue > 50000;
run;
```

Correct answer: a
Your answer:

An index is **not** used with a subsetting IF statement. SAS does **not** use an index to process a WHERE condition that uses the MONTH function. SAS also does **not** use an index to optimize a WHERE statement that uses two conditions which are joined by OR but do not reference the same variable. However, an index **is** used to sort observations for BY processing.

You can learn about

- using indexes in **Creating Samples and Indexes**
- WHERE conditions that cannot be optimized in **Querying Data Efficiently**.

---

48. The following SAS program is submitted:

```
data new(sortedby=date);
   infile rawdata;
   input date mmddyy10. employeeID $ salary;
run;
```

Which one of the following is the purpose of the SORTEDBY= data set option?

a. to specify how the data set is currently sorted
b. to sort the data values in `Date` order in the SAS data set **New**
c. to issue a call to the SORT procedure after the DATA step completes
d. to sort the data in **Rawdata** by `Date` before reading the external file

Correct answer: a
Your answer:

The SORTEDBY= data set option specifies how a data set is currently sorted. SAS uses the sort information to improve efficiency for BY-group processing, index creation, PROC SQL querying, and further sorting.

You can learn about the SORTEDBY= option in **Selecting Efficient Sorting Strategies**.

---

49. Which one of the following SAS programs is the most efficient in producing and removing an index from a SAS data set?

a.
```
proc datasets lib=sasuser;
   modify houses;
   index create style;
   index delete location;
quit;
```

b.
```
proc datasets lib=sasuser;
```

```
      modify houses;
      index delete location;
      index create style;
   quit;
```

c.
```
   proc datasets lib=sasuser;
      modify houses;
      delete index location;
   quit;
   proc datasets lib=sasuser;
      modify houses;
      create index style;
   quit;
```

d.
```
   proc datasets lib=sasuser;
      modify houses;
      delete index location;
      create index style;
   quit;
```

Correct answer: **b**
Your answer:

You can delete an index and create a new index in one PROC DATASETS step, which is more efficient than using separate steps. If you delete the old index first, SAS can reuse the disk space to create the new index.

You can learn about deleting and creating indexes in **Creating Samples and Indexes**.

---

50. Which one of the following is used to create an index for a SAS data set?

a. INDEX function
b. INDEX= data set option
c. CREATE INDEX statement in the INDEX procedure
d. CREATE INDEX statement in the DATASETS procedure

Correct answer: **b**
Your answer:

You use the INDEX= data set option to create an index at the same time that you create a data set. The INDEX function is a character function that is not related to index files. There is no INDEX procedure, and you use the INDEX CREATE statement rather than the CREATE INDEX statement in the DATASETS procedure.

You can learn about creating indexes in **Creating Samples and Indexes**.

Return to Quiz