```sas

%macro print\_hello(name);

%put Hello, &name.! Welcome to SAS Macros!;

%mend;

%print\_hello(John);

```

```sas

%macro summarize\_variable(dataset, var);

proc means data=&dataset;

var &var;

run;

%mend;

%summarize\_variable(MyData, Salary);

%summarize\_variable(MyData, Age);

```

```sas

%let var = value;

```

```sas

data \_null\_;

set dataset;

call symput('var', value);

run;

proc sql noprint;

select value into :var

from table;

quit;

```

```sas

data \_null\_;

/\* Assigning the value of a data step variable to a macro variable \*/

x = 10;

%symput('var', x);

run;

/\* Displaying the value of the macro variable \*/

%put &var;

```

```sas

data x;

x=10;

call symput('var',x);

proc print;

title "Report for &var";

run;

```

```sas

%macro generate\_var;

%let var = value;

%mend;

%generate\_var;

```

```sas

/\* Sample dataset \*/

data SalesData;

input Product $ Sales;

datalines;

A 100

B 150

C 80

D 120

E 200

;

run;

/\* Applying conditional processing \*/

data SalesSummary;

set SalesData;

/\* Example 1: Creating a new variable based on a condition \*/

if Sales >= 100 then Category = 'High';

else Category = 'Low';

/\* Example 2: Filtering observations based on a condition \*/

if Product ne 'C' then output;

/\* Example 3: Using WHERE statement for conditional processing \*/

/\* Creating a dataset with only observations where Sales > 100 \*/

where Sales > 100;

run;

/\* Output dataset SalesSummary will contain:

- A new variable 'Category' categorizing sales as High or Low based on the sales amount.

- Observations excluding Product 'C'.

- Observations where Sales > 100.

\*/

```

```sas

/\* Sample dataset \*/

data Sales;

input Product1 $ Product2 $ Product3 $;

datalines;

A B C

D E F

G H I

;

run;

/\* Array processing \*/

data SalesArray;

set Sales;

/\* Define array to reference multiple variables \*/

array Products[3] $ Product1-Product3;

/\* Example 1: Concatenating values of multiple variables \*/

concat\_products = catx(',', of Products[\*]);

/\* Example 2: Calculating total sales across multiple variables \*/

total\_sales = sum(of Products[\*]);

/\* Example 3: Assigning values to multiple variables using array \*/

do i = 1 to dim(Products);

Products[i] = 'NewValue';

end;

/\* Example 4: Printing values of array elements \*/

do i = 1 to dim(Products);

put Products[i];

end;

/\* Output dataset SalesArray will contain:

- New variables concat\_products and total\_sales.

- Original variables updated with 'NewValue'.

\*/

run;

```

```sas

/\* Sample datasets \*/

data Employees;

input ID Name $ Department $;

datalines;

1 John Sales

2 Alice Marketing

3 Bob HR

;

run;

data Salaries;

input ID Salary;

datalines;

1 50000

2 60000

3 55000

;

run;

/\* Example 1: Basic MERGE statement \*/

data Combined;

merge Employees(in=emp) Salaries(in=sal);

by ID;

if emp and sal;

run;

/\* Example 2: SQL Join \*/

proc sql;

create table Combined\_SQL as

select a.\*, b.Salary

from Employees as a

left join Salaries as b

on a.ID = b.ID;

quit;

/\* Example 3: BY-group processing \*/

proc sort data=Employees;

by Department;

run;

data AverageSalary;

set Salaries;

by ID;

retain SumSalary Count;

if first.ID then do;

SumSalary = Salary;

Count = 1;

end;

else do;

SumSalary + Salary;

Count + 1;

end;

if last.ID;

AverageSalary = SumSalary / Count;

drop Salary;

run;

/\* Output datasets Combined, Combined\_SQL, and AverageSalary demonstrate different advanced merging techniques in SAS. \*/

```

```sas

/\* Sample dataset \*/

data Sales;

input Product $ Month $ Sales;

datalines;

A Jan 100

B Jan 150

A Feb 120

B Feb 180

A Mar 130

B Mar 170

;

run;

/\* Example 1: BY-group processing \*/

proc sort data=Sales;

by Product;

run;

data MonthlyTotal;

set Sales;

by Product;

retain TotalSales;

if first.Product then TotalSales = 0;

TotalSales + Sales;

if last.Product;

drop Sales;

run;

/\* Example 2: Data partitioning with SQL \*/

proc sql;

create table MonthlyTotal\_SQL as

select Product, sum(Sales) as TotalSales

from Sales

group by Product;

quit;

/\* Example 3: Parallel processing \*/

proc sort data=Sales out=Sales\_sorted;

by Product;

run;

proc sort data=Sales\_sorted;

by Month;

run;

data ParallelProcessing;

set Sales\_sorted;

run;

/\* Output datasets MonthlyTotal, MonthlyTotal\_SQL, and ParallelProcessing demonstrate different data partitioning and processing techniques in SAS. \*/

```

```sas

/\* Sample dataset \*/

data Sales;

input Product $ Month $ Sales;

datalines;

A Jan 100

B Jan 150

C Jan .

A Feb 120

B Feb 180

C Feb 200

A Mar 130

B Mar .

C Mar 220

;

run;

/\* Example 1: Missing value handling \*/

proc sql;

create table Sales\_no\_missing as

select \*

from Sales

where not missing(Sales);

quit;

/\* Example 2: Outlier detection \*/

proc univariate data=Sales noprint;

var Sales;

output out=Sales\_summary p5(pctlpts=5) p95(pctlpts=95);

run;

data Sales\_no\_outliers;

set Sales;

if Sales >= 95th\_percentile or Sales <= 5th\_percentile then delete;

run;

/\* Example 3: Error handling with conditional logic \*/

data Adjusted\_Sales;

set Sales;

if Sales < 0 then Sales = .; /\* Set negative sales values to missing \*/

run;

/\* Output datasets Sales\_no\_missing, Sales\_no\_outliers, and Adjusted\_Sales demonstrate different data validation and error handling techniques in SAS. \*/

```

```sas

proc sql;

connect to odbc (dsn='SalesDB' uid='username' pwd='password');

create table SalesData as

select \*

from connection to odbc (

select \*

from SalesTransactions

where TransactionDate >= '2023-01-01'

);

disconnect from odbc;

quit;

```

```sas

proc sql;

create table SalesSummary as

select Region,

sum(SalesAmount) as TotalSales

from SalesData

where Year(Date) = 2023

group by Region;

quit;

```

```sas

/\* Applying SAS Formats \*/

data SalesReport;

set SalesData;

format DateSold date9. SalesAmount dollar10.2;

run;

/\* Using SAS Functions \*/

data SalesSummary;

set SalesData;

TotalSales = sum(SalesAmount, Discounts);

run;

```

```sas

/\* Array Processing Example \*/

data SalesData;

input Product1 Product2 Product3;

datalines;

100 150 200

80 120 180

;

run;

data TotalSales;

set SalesData;

array Products{3} Product1-Product3;

TotalSales = sum(of Products[\*]);

run;

```

```sas

/\* Advanced ODS Techniques Example \*/

ods html file="output.html" style=sasweb;

title 'Sales Analysis Report';

proc print data=sales\_data(obs=10) noobs;

var Product Sales Region;

run;

proc sgplot data=sales\_data;

scatter x=Product y=Sales / group=Region;

xaxis label='Product' values=('Product1' 'Product2' 'Product3');

yaxis label='Sales';

title 'Product Sales by Region';

run;

ods html close;

```

```sas

/\* Performance Tuning Example \*/

options compress=yes; /\* Enable data compression \*/

data sales\_data;

set large\_sales\_data;

/\* Data processing steps \*/

run;

```

```sas

/\* Error Handling Example \*/

%macro process\_data(input\_ds);

%local rc;

/\* Check if dataset exists \*/

%if %sysfunc(exist(&input\_ds)) %then %do;

/\* Data processing steps \*/

data output\_ds;

set &input\_ds;

/\* Data transformation logic \*/

run;

%end;

%else %do;

%put ERROR: Dataset &input\_ds does not exist.;

%let rc = %sysfunc(sysmsg());

%end;

/\* Return error code \*/

&rc

%mend;

/\* Usage \*/

%let error\_code = %process\_data(input\_data);

%if &error\_code ne 0 %then %put ERROR: Data processing failed.;

```