SAS® Programming 2: Data Manipulation Techniques

Case Study

SAS® Programming 2: Data Manipulation Techniques Case Study was developed by Greg Treiman. Instructional design, editing, and production support was provided by the Learning Design and Development team.

SAS® Programming 2: Data Manipulation Techniques Case Study

Course code PG2V2CS, prepared date 13Dec2022. PG2V2CS_001

Lesson 1 Case Study

1.1 Case Study Introduction

In this case study, you solve a real-world business problem by applying concepts that you learned in the SAS Programming 2: Data Manipulation Techniques course. Be aware that there are numerous solutions to this problem, and some can include concepts that are outside the scope of this course.

Background Information

In your current role, it is your responsibility to use your SAS programming skills to acquire, organize, and prepare data for your company's business analysts to use. The business analysts will use the data that you provide to create reports, visualizations, and statistical models designed to grow your company's market share.

How to Attempt the Case Study

There are three ways to complete this case study. Follow the method that fits your skill set.

Advanced Method

- If you feel comfortable with the topics in the SAS Programming 2: Data Manipulation Techniques course and want to treat this as a real-world problem, follow the "Advanced Method" section. Only project requirements are defined, and you must determine the SAS code necessary to complete each task. During the process, feel free to use your course notes, SAS documentation, or other online resources. If you are stuck on a specific task, you can find suggestions in the "Intermediate Method" section and solution code in the "Beginner Method" section.

Intermediate Method

- If you think that you might need a bit of assistance in the case study, follow the "Intermediate Method" section. The guide does not give you the solutions. Instead, it provides a roadmap for how to solve the problem. If you are stuck on a specific task, you can find solution code in the "Beginner Method" section.

Beginner Method

- If you are not familiar with the SAS programming language and want to use the case study as a walk-through demo, follow the "Beginner Method" section. A detailed description of steps and the solution code are provided. After you walk through the case study as a demo, we recommend that you go back and attempt it on your own.

If you have any questions regarding the case study, if you came up with different solutions and want to show them off, or if you want to share additional visualizations, post in the SAS Training Forum. We would love to hear from you!

Business Problem

Your company is interested in analyzing 2014 global tourism data. Your supervisor has sent you two SAS data sets with an email that contains a list of data requirements and an example of the desired outcome. It is your job to follow your supervisor's requirements and prepare the data for use.

Here are the three tables that you will deliver upon successful completion of the project:

- **cleaned_tourism**
- **final_tourism**
- **nocountryfound**

Specifications for these tables can be found in the "Requirements" subsection of this document.

Data Information

In this case study, two source SAS data sets are provided:

- **tourism**
  contains information about the arrivals of nonresident tourists/visitors, departures, and tourism expenditure in the country and in other countries. The raw data was downloaded from UNdata at https://data.un.org/
- **country_info**
  contains country names with the associated continent IDs.

Resources

To create the tables for the Tourism case study, open the **case_study** folder located in the **S:/workshop** folder in the virtual lab.

1. Open the program called **PG2_CaseStudy_cre8data.sas**.
2. Run the program to create the **tourism** and **country_info** tables.
3. Confirm that both tables have been created in your **Work** library.

The **LogScanner.sas** SAS program (bonus) is also available in the **case_study** folder located in the **S:/workshop** folder in the virtual lab.

Data Layout

Here is the column information for the two tables that you will use:

tourism

Column	Type	Description
A	Num	Numeric ID when a country appears in the **Country** column.
Country	Char	This column contains country names, tourism type, tourism category, and the conversion type.
Series	Char	**Inbound tourism:** *TF* = Arrivals of non-resident tourists at national borders *VF* = Arrivals of non-resident visitors at national borders *THS* = Arrivals of non-resident tourists in hotels and similar establishments *TCE* = Arrivals of non-resident tourists in all types of accommodation establishments **Outbound tourism:** *TF* = Departures - trips abroad by resident tourists *VF* = Departures - trips abroad by resident visitors **Monetary amounts:** *IMF* = International Monetary Fund *CB* = Central Bank *country* = From national tourism bureau or organization within country .. = Data not available
_1995 through **_2014**	Char	Scaled numeric data stored as text. The **Country** column contains the information necessary to properly convert this data to a numeric value. Here are the values: • dollar amounts (in millions) for rows containing expenditure data • passenger count in thousands for rows containing arrival or departure data. .. = Data not available

country_info

Column	Type	Description
Continent	Num	Continent ID associated with the country. 1 = North America 2 = South America 3 = Europe 4 = Africa 5 = Asia 6 = Oceania 7 = Antarctica
Country	Char	A character country name.

Requirements

Your supervisor has emailed you the requirements and desired outcome for this project. Be sure to read through all the requirements thoroughly for information about how to manipulate the data to create the desired output.

Raw Data

Here is a partial image of the raw data that was imported into SAS for you to use:

A	COUNTRY	Series	1995	1996	1997	1998	1999	2000	2001	2002	2003
826	UNITED KINGDOM										
	Inbound tourism										
	Arrivals - Thousands	TF	21,719	22,936	23,215	23,710	23,341	23,212	20,982	22,307	22,787
	Tourism expenditure in the country - US$ Mn	IMF	**27,577**	**29,181**	**30,483**	**31,658**	**30,807**	**29,978**	**26,137**	**27,819**	**30,736**
	Travel - US$ Mn	IMF	20,487	21,389	22,586	23,689	22,716	21,769	18,864	20,549	22,668
	Passenger transport - US$ Mn	IMF	7,090	7,792	7,897	7,969	8,091	8,209	7,273	7,270	8,068
	Outbound tourism										
	Departures - Thousands	TF	41,345	42,050	45,957	50,872	53,881	56,837	58,281	59,377	61,424
	Tourism expenditure in other countries - US$ Mn	IMF	**30,749**	**32,298**	**35,954**	**41,458**	**45,536**	**47,009**	**46,410**	**51,125**	**58,627**
	Travel - US$ Mn	IMF	24,926	25,962	28,529	33,452	37,034	38,262	37,931	41,744	47,853
	Passenger transport - US$ Mn	IMF	5,823	6,336	7,425	8,006	8,502	8,747	8,479	9,381	10,774

Desired Outcome

The final table should be a combination of the **cleaned_tourism** table and the **country_info** table. It should resemble the example below.

COUNTRY_NAME	TOURISM_TYPE	CATEGORY	SERIES	Y2014	CONTINENT
UNITED KINGDOM	Inbound tourism	Arrivals	TF	32,613,000	Europe
UNITED KINGDOM	Inbound tourism	Tourism expenditure in the country - US$	IMF	**62,830,000,000**	Europe
UNITED KINGDOM	Inbound tourism	Travel - US$	IMF	46,723,000,000	Europe
UNITED KINGDOM	Inbound tourism	Passenger transport - US$	IMF	16,107,000,000	Europe
UNITED KINGDOM	Outbound tourism	Departures	TF	60,082,000	Europe
UNITED KINGDOM	Outbound tourism	Tourism expenditure in other countries - US$	IMF	**79,935,000,000**	Europe
UNITED KINGDOM	Outbound tourism	Travel - US$	IMF	63,424,000,000	Europe
UNITED KINGDOM	Outbound tourism	Passenger transport - US$	IMF	16,511,000,000	Europe

Data Requirements

The requirements have been split into two parts.

Part 1: cleaned_tourism table

Here are the requirements that your supervisor specified when asking you to create the **cleaned_tourism** table:

- **Tourism_Type** – Create a new column that contains the type of tourism by reorganizing the original **Country** column. Valid values are *Inbound tourism* or *Outbound tourism*.
- **Category** – Create a new column that contains the category names in the data by extracting and modifying values from the original **Country** column. Original categories values need to be updated to the new values shown below. In total, there should be six distinct values.

Original Category Distinct Values	New Distinct Values
Arrivals - Thousands	*Arrivals*
Departures - Thousands	*Departures*
Passenger transport - US$ Mn	*Passenger transport - US$*
Tourism expenditure in other countries - US$ Mn	*Tourism expenditure in other countries - US$*
Tourism expenditure in the country - US$ Mn	*Tourism expenditure in the country - US$*
Travel - US$ Mn	*Travel - US$*

- **Series** – Convert all values to uppercase and change data that is not available (..) to a missing character value.
- **Y2014** – Create a new column that changes the scaled character values in the original **2014** year column to the full numeric value. To create the numeric values, determine the conversion type and multiply with the scaled original values. The row category determines whether the value for **Y2014** should be in the millions (abbreviated Mn) or thousands. The new values should be formatted with the COMMA format.
  - For example, if the category is *Travel - US$ MN* and the value for **_2014** is 4.26, **Y2014** is equal to 4.26 * 1000000, or 4,260,000.
- Include only **Country_Name**, **Tourism_Type**, **Category**, **Series**, and **Y2014** in the **cleaned_tourism** output table.
- ***Bonus task***: After creating the **cleaned_tourism** table, run the **Log Checker.sas** program on your log to ensure that your code meets our regulatory requirements.

Part 2: Merging and formatting

Lastly, merge the **cleaned_tourism** table with the **country_info** table and do the following:

- Create two new tables:
  - one table named **final_tourism** that contains ***only*** merged data
  - a second table named **nocountryfound** that contains a list of distinct countries from the **cleaned_tourism** table that do not have a match in the **country_info** table

- Create a format for the **Continent** column that assigns continent IDs to the corresponding continent names. Permanently apply the format in the **final_tourism** table.
  - 1 = North America
  - 2 = South America
  - 3 = Europe
  - 4 = Africa
  - 5 = Asia
  - 6 = Oceania
  - 7 = Antarctica

1.2 Advanced Method

This section is an advanced approach for solving the case study. The business requirements and questions are presented without specific code suggestions. You can refer to the corresponding numbered steps in the Intermediate Method section for detailed guidance, or you can refer to the Beginner Method section for suggested code.

For this case study, the steps will be broken down into two sections for you to follow. The first section is preparing and cleaning the **tourism** data, and the second section is merging **cleaned_tourism** table with the **country_info** table. Each section contains steps to help guide you through the problem.

For more information, you can visit the SAS Documentation or post a question in the SAS Training Community.

Prepare the Tourism Data

1. Remove the years 1994 through 2013 from the **tourism** data. Write the resulting table to **cleaned_tourism**.
2. Modify the code in the previous step to create and retain the **Country_Name** and **Tourism_Type** columns and assign appropriate lengths.
3. Modify the code in the previous step to set the **Country_Name** column equal to the country name in the **Country** column. Set the **Tourism_Type** column to either *Inbound tourism* or *Outbound tourism*, according to the value of the **Country** column.

   Note that the **Country** column contains the country name, the type of tourism, and details about the tourism, all in one column. Some rows function as “headers” of the section. The goal is to extract those “header” values and add them to a new column.
4. Modify the code in the previous step to remove the “header” rows with just the country names and tourism types.
5. Modify the code in the previous step to convert values to uppercase for **Series** and also convert its missing values from .. to a missing character value.
6. Modify the code in the previous step to create a new column named **ConversionType** that is equal to the last word in the **Country** column. Hint: It should be either *Mn* or *Thousands*.
7. Modify the code in the previous step to replace the value .. in the **_2014** column with a single period. This will be used later to create a numeric variable.
8. The **ConversionType** column represents whether values in the **_2014** column are being stored as thousands or millions. Use this column to modify the code in the previous step and calculate a new numeric column named **Y2014**. Note that you need to convert the **_2014** column to numeric before multiplying.
9. Modify the code in the previous step to create a new column named **Category,** following the rules described in the “Data Requirements” subsection.
10. Modify the code in the previous step to add a permanent format to the **Y2014** column.
11. Modify the code in the previous step to remove unnecessary variables.

Merge the Data

1. Create a format for the **Continents** column as described in the “Data Requirements” subsection.
2. Sort the **country_info** table by **Country**. Rename the **Country** column as **Country_Name** and write the output to **country_sorted**.
3. Merge the **cleaned_tourism** table with **country_sorted**. Per the guidelines in the “Data Requirements” subsection, create the **final_tourism** and **nocountryfound** tables and permanently apply the CONTINENTS format.

Bonus

In regulated industries, there frequently are automated processes that scan logs for “forbidden” messages and create reports of items requiring justification to regulators.

1. In SAS Enterprise Guide, enable application logging and take note of the log file name and location.
2. Rerun the program that you wrote to prepare the tourism table.
3. Open the **LogScanner.sas** program and run it to test your code.

1.3 Intermediate Method

This section is a step-by-step guide for solving the case study. The numbered steps are the same as in the Advanced Method section, but detailed suggestions are included here. To view possible code to complete each step, refer to the Beginner Method section.

For this case study, the steps will be broken down into two sections for you to follow. The first section is preparing and cleaning the **tourism** data, and the second section is merging **cleaned_tourism** table with the **country_info** table. Each section contains steps to help guide you through the problem.

For more information, you can visit the SAS Documentation or post a question in the SAS Training Community.

Prepare the Tourism Data

1. Remove the years 1994 through 2013 from the **tourism** data. Write the resulting table to **cleaned_tourism**.
   a. Open a new SAS program and write a DATA step that reads the **tourism** table.
   b. Use the DROP= data set option to drop the variables **_1994** through **_2013**.
2. Modify the code in the previous step to create and retain the **Country_Name** and **Tourism_Type** columns and assign appropriate lengths.
   a. Add a LENGTH statement before the SET statement to create **Country_Name** as a character variable with a length of 300 and **Tourism_Type** as a character variable with a length of 20.
   b. Add a RETAIN statement after the LENGTH statement to retain the **Country_Name** and **Tourism_Type** variables. The initial value should be a missing character value.
3. Modify the code in the previous step to set the **Country_Name** column equal to the country name in the **Country** column. Set the **Tourism_Type** column to either *Inbound tourism* or *Outbound tourism*, according to the value of the **Country** column.

   Note that the **Country** column contains the country name, the type of tourism, and details about the tourism, all in one column. Some rows function as “headers” of the section. The goal is to extract those “header” values and add them to a new column.
   a. Add an IF-THEN statement that sets **Country_Name** equal to **Country** when the **A** column is not equal to . (a missing numeric value).
   b. Add an IF-THEN statement that sets **Tourism_Type** equal to *Inbound tourism* when **Country** is equal to *INBOUND TOURISM*. Make sure to account for casing.
   c. Add an ELSE IF-THEN statement that sets **Tourism_Type** equal to *Outbound tourism* when **Country** is equal to *OUTBOUND TOURISM*. Make sure to account for casing.
4. Modify the code in the previous step to remove the “header” rows with just the country names and tourism types.
   a. Add a subsetting IF statement to filter for only rows where **Country_Name** is not equal to **Country** and **Country** is not equal to **Tourism_Type**.

5. Modify the code in the previous step to convert values to uppercase for **Series** and also convert its missing values from *..* to a missing character value.
   a. Add an assignment statement and use the UPCASE function to convert **Series** to uppercase.
   b. Add an IF-THEN statement to set **Series** equal to a missing character value when **Series** equals two periods (..).
6. Modify the code in the previous step to create a new column named **ConversionType** that is equal to the last word in the **Country** column. Hint: It should be either *Mn* or *Thousands*.
   a. Add an assignment statement that creates the variable **ConversionType** and use the SCAN and STRIP functions to extract the final space-separated string from the **Country** column.
7. Modify the code in the previous step to replace the value *..* in the **_2014** column with a single period. This will be used later to create a numeric variable.
   a. Add an IF-THEN statement to set **_2014** equal to a single period when **Series** equals two periods (..).
8. The **ConversionType** column represents whether values in the **_2014** column are being stored as thousands or millions. Use this column to modify the code in the previous step and calculate a new numeric column named **Y2014**. Note that you need to convert the **_2014** column to numeric before multiplying.
   a. Add an IF-THEN/DO statement that evaluates whether **ConversionType** is equal to *Mn*.
   b. Inside the DO block, add an IF-THEN statement that uses the INPUT function and sets **Y2014** equal to the numeric value of **_2014** multiplied by 1,000,000 if the numeric value of **_2014** is not missing.
   c. Add an ELSE statement that sets **Y2014** equal to a missing numeric value if the previous IF condition is not met.
   d. Close the DO block.
   e. Add an ELSE IF-THEN/DO statement that evaluates whether **ConversionType** is equal to *Thousands*.
   f. Inside the second DO block, add an IF-THEN statement that uses the INPUT function and sets **Y2014** equal to the numeric value of **_2014** multiplied by 1,000 if the numeric value of **_2014** is not missing.
   g. Add an ELSE statement that sets **Y2014** equal to a missing numeric value if the previous IF condition is not met.
   h. Close the second DO block.
9. Modify the code in the previous step to create a new column named **Category,** following the rules described in the “Data Requirements” subsection.
   a. Update the first DO block created in Step 8, where **ConversionType** equals *Mn*. Add an assignment statement that sets **Category** equal to the portion of **Country** before the hyphen using the SCAN function, and append - *US$* using the CAT function. Remove leading and trailing blanks by using the R argument in the SCAN function.
   b. Update the second DO block created in Step 8, where **ConversionType** equals *Thousands*. Add an assignment statement that sets **Category** equal to the portion of **Country** before the hyphen using the SCAN function. Remove leading and trailing blanks by using the R argument in the SCAN function.

10. Modify the code in the previous step to add a permanent format to the **Y2014** column.
    a. Add a FORMAT statement that applies the COMMA25. format to the **Y2014** column**.**
11. Modify the code in the previous step to remove unnecessary variables.
    a. Add a DROP statement that removes the **A**, **ConversionType**, **Country**, and **_2014** columns**.**

Merge the Data

1. Create a format for the **Continents** column as described in the “Data Requirements” subsection.
    a. Write a PROC FORMAT step.
    b. Add a VALUE statement that creates a format named **continents**. Assign continent IDs to the corresponding continent names as described in the “Data Requirements” subsection.
2. Sort the **country_info** table by **Country**. Rename the **Country** column as **Country_Name** and write the output to **country_sorted**.
    a. Write a PROC SORT step and use **country_info** as the input table.
    b. Add the RENAME= data set option to the **work.country_info** data set to rename the **Country** column as **Country_Name**.
    c. Add an OUT= option to write the output to a new table named **country_sorted**.
    d. Add a BY statement to sort the table by **Country_Name**.
3. Merge the **cleaned_tourism** table with **country_sorted**. Per the guidelines in the “Data Requirements” subsection, create the **final_tourism** and **nocountryfound** tables and permanently apply the CONTINENTS format.
    a. Write a DATA step that creates the **final_tourism** table and the **nocountryfound** table. Use the KEEP= data set option to keep only the **Country_Name** column in the **nocountryfound** table.
    b. Add a MERGE statement to merge the **cleaned_tourism** and **country_sorted** tables. Add the IN= data set option to both tables to add temporary flags in the PDV.
    c. Add an IF-THEN statement to write only rows that appear in both the **cleaned_tourism** and **country_sorted** tables to the **final_tourism** table.
    d. Add an IF-THEN statement to write rows that appear in the **cleaned_tourism** table but not in the **country_sorted** table and that are the first instance of a particular **Country_Name** value to the **nocountryfound** table.
    e. Add a FORMAT statement to apply the CONTINENTS format to the **Continent** column.

Bonus

In regulated industries, there frequently are automated processes that scan logs for “forbidden” messages and create reports of items requiring justification to regulators.

1. In SAS Enterprise Guide, enable application logging and take note of the log file name and location.
   a. Select **Tools** and then **Options**.
   b. Select **Application Logging**.
   c. Select the **Enable logging** option.
   d. Make note of the **Configuration** file and the **Log** folder.
2. Rerun the program that you wrote to prepare the tourism table.
3. Open the **LogScanner.sas** program and run it to test your code.
   a. Change the value of the macro variable **logpath** to the location where you saved your log file.
   b. Change the value of the macro variable **filename** to the name of log file.

1.4 Beginner Method

This section is a step-by-step guide for solving the case study, including solutions for each task. The numbered steps are the same steps from the “Advanced Method” and “Intermediate Method” sections, but here the solution code for each step is available.

For this case study, the steps are broken down into two sections for you to follow. The first section is preparing and cleaning the **tourism** data, and the second section is merging the **cleaned_tourism** table with the **country_info** table. Each section contains steps to help guide you through the problem.

For more information, you can visit SAS Documentation or post a question in the SAS Training Community.

Prepare the Tourism Data

1. Remove the years 1994 through 2013 from the **tourism** data. Write the resulting table to **cleaned_tourism**.
   a. Open a new SAS program and write a DATA step that reads the **tourism** table.
   b. Use the DROP= data set option to drop the variables **_1994** through **_2013**.

```
data cleaned_tourism;
    set work.tourism (drop=_1995-_2013);
run;
```

2. Modify the code in the previous step to create and retain the **Country_Name** and **Tourism_Type** columns and assign appropriate lengths.
   a. Add a LENGTH statement before the SET statement to create **Country_Name** as a character variable with a length of 300 and **Tourism_Type** as a character variable with a length of 20.
   b. Add a RETAIN statement after the LENGTH statement to retain the **Country_Name** and **Tourism_Type** variables. The initial value should be a missing character value.

```
data cleaned_tourism;
    length Country_Name $300 Tourism_Type $20;
    retain Country_Name "" Tourism_Type "";
    set work.tourism (drop=_1995-_2013);
run;
```

3. Modify the code in the previous step to set the **Country_Name** column equal to the country name in the **Country** column. Set the **Tourism_Type** column to either *Inbound tourism* or *Outbound tourism*, according to the value of the **Country** column.

   Note that the **Country** column contains the country name, the type of tourism, and details about the tourism, all in one column. Some rows function as “headers” of the section. The goal is to extract those “header” values and add them to a new column.
   a. Add an IF-THEN statement that sets **Country_Name** equal to **Country** when the **A** column is not equal to . (a missing numeric value).
   b. Add an IF-THEN statement that sets **Tourism_Type** equal to *Inbound tourism* when **Country** is equal to *INBOUND TOURISM*. Make sure to account for casing.

c. Add an ELSE IF-THEN statement that sets **Tourism_Type** equal to *Outbound tourism* when **Country** is equal to *OUTBOUND TOURISM*. Make sure to account for casing.

```
data cleaned_tourism;
    length Country_Name $300 Tourism_Type $20;
    retain Country_Name "" Tourism_Type "";
    set work.tourism (drop=_1995-_2013);
    if A ne . then Country_Name=Country;
    if lowcase(Country)='inbound tourism' then
        Tourism_Type="Inbound tourism";
    else if lowcase(Country)='outbound tourism' then
        Tourism_Type="Outbound tourism";
run;
```

4. Modify the code in the previous step to remove the “header” rows with just the country names and tourism types.

a. Add a subsetting IF statement to filter for only rows where **Country_Name** is not equal to **Country** and **Country** is not equal to **Tourism_Type**.

```
data cleaned_tourism;
    length Country_Name $300 Tourism_Type $20;
    retain Country_Name "" Tourism_Type "";
    set work.tourism (drop=_1995-_2013);
    if A ne . then Country_Name=Country;
    if lowcase(Country)='inbound tourism' then
        Tourism_Type="Inbound tourism";
    else if lowcase(Country)='outbound tourism' then
        Tourism_Type="Outbound tourism";
    if Country_Name ne Country and Country ne Tourism_Type;
run;
```

5. Modify the code in the previous step to convert values to uppercase for **Series** and also convert its missing values from .. to a missing character value.

a. Add an assignment statement and use the UPCASE function to convert **Series** to uppercase.

b. Add an IF-THEN statement to set **Series** equal to a missing character value when **Series** equals two periods (..).

```
data cleaned_tourism;
    length Country_Name $300 Tourism_Type $20;
    retain Country_Name "" Tourism_Type "";
    set work.tourism (drop=_1995-_2013);
    if A ne . then Country_Name=Country;
    if lowcase(Country)='inbound tourism' then
        Tourism_Type="Inbound tourism";
    else if lowcase(Country)='outbound tourism' then
        Tourism_Type="Outbound tourism";
    if Country_Name ne Country and Country ne Tourism_Type;
    Series=upcase(series);
    if Series=".." then Series="";
run;
```

6. Modify the code in the previous step to create a new column named **ConversionType** that is equal to the last word in the **Country** column. Hint: It should be either *Mn* or *Thousands*.

   a. Add an assignment statement that creates the variable **ConversionType** and use the SCAN and STRIP functions to extract the final space-separated string from the **Country** column.

```
data cleaned_tourism;
    length Country_Name $300 Tourism_Type $20;
    retain Country_Name "" Tourism_Type "";
    set work.tourism (drop=_1995-_2013);
    if A ne . then Country_Name=Country;
    if lowcase(Country)='inbound tourism' then
        Tourism_Type="Inbound tourism";
    else if lowcase(Country)='outbound tourism' then
        Tourism_Type="Outbound tourism";
    if Country_Name ne Country and Country ne Tourism_Type;
    Series=upcase(series);
    if Series=".." then Series="";
    ConversionType=strip(scan(country,-1,' '));
run;
```

7. Modify the code in the previous step to replace the value .. in the **_2014** column with a single period. This will be used later to create a numeric variable.

   a. Add an IF-THEN statement to set **_2014** equal to a single period when **Series** equals two periods (..).

```
data cleaned_tourism;
    length Country_Name $300 Tourism_Type $20;
    retain Country_Name "" Tourism_Type "";
    set work.tourism (drop=_1995-_2013);
    if A ne . then Country_Name=Country;
    if lowcase(Country)='inbound tourism' then
        Tourism_Type="Inbound tourism";
    else if lowcase(Country)='outbound tourism' then
        Tourism_Type="Outbound tourism";
    if Country_Name ne Country and Country ne Tourism_Type;
    Series=upcase(series);
    if Series=".." then Series="";
    ConversionType=strip(scan(country,-1,' '));
    if _2014='..' then _2014='.';
run;
```

8. The **ConversionType** column represents whether values in the **_2014** column are being stored as thousands or millions. Use this column to modify the code in the previous step and calculate a new numeric column named **Y2014**. Note that you need to convert the **_2014** column to numeric before multiplying.

   a. Add an IF-THEN/DO statement that evaluates whether **ConversionType** is equal to *Mn*.

   b. Inside the DO block, add an IF-THEN statement that uses the INPUT function and sets **Y2014** equal to the numeric value of **_2014** multiplied by 1,000,000 if the numeric value of **_2014** is not missing.

c. Add an ELSE statement that sets **Y2014** equal to a missing numeric value if the previous IF condition is not met.

d. Close the DO block.

e. Add an ELSE IF-THEN/DO statement that evaluates whether **ConversionType** is equal to *Thousands*.

f. Inside the second DO block, add an IF-THEN statement that uses the INPUT function and sets **Y2014** equal to the numeric value of **_2014** multiplied by 1,000 if the numeric value of **_2014** is not missing.

g. Add an ELSE statement that sets **Y2014** equal to a missing numeric value if the previous IF condition is not met.

h. Close the second DO block.

```
data cleaned_tourism;
    length Country_Name $300 Tourism_Type $20;
    retain Country_Name "" Tourism_Type "";
    set work.tourism (drop=_1995-_2013);
    if A ne . then Country_Name=Country;
    if lowcase(Country)='inbound tourism' then
       Tourism_Type="Inbound tourism";
    else if lowcase(Country)='outbound tourism' then
       Tourism_Type="Outbound tourism";
    if Country_Name ne Country and Country ne Tourism_Type;
    Series=upcase(series);
    if Series=".." then Series="";
    ConversionType=strip(scan(country,-1,' '));
    if _2014='..' then _2014='.';
    if ConversionType = 'Mn' then
       do;
          if input(_2014,16.) ne . then
             Y2014=input(_2014,16.)*1000000;
             else Y2014=.;
       end;
    else if ConversionType = 'Thousands' then
       do;
          if input(_2014,16.) ne . then
             Y2014=input(_2014,16.)*1000;
             else Y2014=.;
       end;
run;
```

9. Modify the code in the previous step to create a new column named **Category,** following the rules described in the “Data Requirements” subsection.

a. Update the first DO block created in Step 8, where **ConversionType** equals *Mn*. Add an assignment statement that sets **Category** equal to the portion of **Country** before the hyphen using the SCAN function, and append - *US$* using the CAT function. Remove leading and trailing blanks by using the R argument in the SCAN function.

b. Update the second DO block created in Step 8, where **ConversionType** equals *Thousands*. Add an assignment statement that sets **Category** equal to the portion of **Country** before the hyphen using the SCAN function. Remove leading and trailing blanks by using the R argument in the SCAN function.

```
data cleaned_tourism;
    length Country_Name $300 Tourism_Type $20;
    retain Country_Name "" Tourism_Type "";
    set work.tourism (drop=_1995-_2013);
    if A ne . then Country_Name=Country;
    if lowcase(Country)='inbound tourism' then
        Tourism_Type="Inbound tourism";
    else if lowcase(Country)='outbound tourism' then
        Tourism_Type="Outbound tourism";
    if Country_Name ne Country and Country ne Tourism_Type;
    Series=upcase(series);
    if Series=".." then Series="";
    ConversionType=strip(scan(country,-1,' '));
    if _2014='..' then _2014='.';
    if ConversionType = 'Mn' then
       do;
          if input(_2014,16.) ne . then
             Y2014=input(_2014,16.)*1000000;
             else Y2014=.;
          Category=cat(scan(country,1,'-','r')," - US$");
       end;
    else if ConversionType = 'Thousands' then
       do;
          if input(_2014,16.) ne . then
             Y2014=input(_2014,16.)*1000;
             else Y2014=.;
          Category=scan(country,1,'-','r');
       end;
run;
```

10. Modify the code in the previous step to add a permanent format to the **Y2014** column.

a. Add a FORMAT statement that applies the COMMA25. format to the **Y2014** column.

```
data cleaned_tourism;
    length Country_Name $300 Tourism_Type $20;
    retain Country_Name "" Tourism_Type "";
    set work.tourism (drop=_1995-_2013);
    if A ne . then Country_Name=Country;
    if lowcase(Country)='inbound tourism' then
        Tourism_Type="Inbound tourism";
    else if lowcase(Country)='outbound tourism' then
        Tourism_Type="Outbound tourism";
    if Country_Name ne Country and Country ne Tourism_Type;
    Series=upcase(series);
    if Series=".." then Series="";
    ConversionType=strip(scan(country,-1,' '));
```

```
    if _2014='..' then _2014='.';
        if ConversionType = 'Mn' then
       do;
          if input(_2014,16.) ne . then
             Y2014=input(_2014,16.)*1000000;
             else Y2014=.;
          Category=cat(scan(country,1,'-','r')," - US$");
       end;
    else if ConversionType = 'Thousands' then
       do;
          if input(_2014,16.) ne . then
             Y2014=input(_2014,16.)*1000;
             else Y2014=.;
          Category=scan(country,1,'-','r');
       end;
    format Y2014 comma25.;
run;
```

11. Modify the code in the previous step to remove unnecessary variables.

a. Add a DROP statement that removes the **A**, **ConversionType**, **Country**, and **_2014** columns.

```
data cleaned_tourism;
    length Country_Name $300 Tourism_Type $20;
    retain Country_Name "" Tourism_Type "";
    set work.tourism (drop=_1995-_2013);
    if A ne . then Country_Name=Country;
    if lowcase(Country)='inbound tourism' then
        Tourism_Type="Inbound tourism";
    else if lowcase(Country)='outbound tourism' then
        Tourism_Type="Outbound tourism";
    if Country_Name ne Country and Country ne Tourism_Type;
    Series=upcase(series);
    if Series=".." then Series="";
    ConversionType=strip(scan(country,-1,' '));
    if _2014='..' then _2014='.';
    if ConversionType = 'Mn' then
       do;
         if input(_2014,16.) ne . then
            Y2014=input(_2014,16.)*1000000;
            else Y2014=.;
         Category=cat(scan(country,1,'-','r')," - US$");
       end;
    else if ConversionType = 'Thousands' then
       do;
         if input(_2014,16.) ne . then
            Y2014=input(_2014,16.)*1000;
            else Y2014=.;
         Category=scan(country,1,'-','r');
       end;
```

```
    format Y2014 comma25.;
    drop A ConversionType Country _2014;
run;
```

Merge the Data

1. Create a format for the **Continents** column as described in the “Data Requirements” subsection.
   a. Write a PROC FORMAT step.
   b. Add a VALUE statement that creates a format named **continents**. Assign continent IDs to the corresponding continent names as described in the “Data Requirements” subsection.

```
proc format;
   value continents
      1 = "North America"
      2 = "South America"
      3 = "Europe"
      4 = "Africa"
      5 = "Asia"
      6 = "Oceania"
      7 = "Antarctica";
run;
```

2. Sort the **country_info** table by **Country**. Rename the **Country** column as **Country_Name** and write the output to **country_sorted**.
   a. Write a PROC SORT step and use **country_info** as the input table.
   b. Add the RENAME= data set option to the **work.country_info** data set to rename the **Country** column as **Country_Name**.
   c. Add an OUT= option to write the output to a new table named **country_sorted**.
   d. Add a BY statement to sort the table by **Country_Name**.

```
proc sort data=work.country_info(rename=(Country=Country_Name))
          out=Country_Sorted;
    by Country_Name;
run;
```

3. Merge the **cleaned_tourism** table with **country_sorted**. Per the guidelines in the “Data Requirements” subsection, create the **final_tourism** and **nocountryfound** tables and permanently apply the CONTINENTS format.
   a. Write a DATA step that creates the **final_tourism** table and the **nocountryfound** table. Use the KEEP= data set option to keep only the **Country_Name** column in the **nocountryfound** table.
   b. Add a MERGE statement to merge the **cleaned_tourism** and **country_sorted** tables. Add the IN= data set option to both tables to add temporary flags in the PDV.
   c. Add an IF-THEN statement to write only rows that appear in both the **cleaned_tourism** and **country_sorted** tables to the **final_tourism** table.
   d. Add an IF-THEN statement to write rows that appear in the **cleaned_tourism** table but not in the **country_sorted** table and that are the first instance of a particular **Country_Name** value to the **nocountryfound** table.

e. Add a FORMAT statement to apply the CONTINENTS format to the **Continent** column.

```
data final_tourism nocountryfound(keep=Country_Name);
    merge cleaned_tourism(in=t)
          country_sorted(in=c);
    by country_name;
    if t=1 and c=1 then output final_tourism;
    if (t=1 and c=0) and first.country_name then
       output nocountryfound;
    format Continent continents.;
run;
```

Bonus

In regulated industries, there frequently are automated processes that scan logs for “forbidden” messages and create reports of items requiring justification to regulators.

1. In SAS Enterprise Guide, enable application logging and take note of the log file name and location.
   a. Select **Tools** and then **Options**.
   b. Select **Application Logging**.
   c. Select the **Enable logging** option.
   d. Make note of the **Configuration** file and the **Log** folder.
2. Rerun the program that you wrote to prepare the tourism table.
3. Open the **LogScanner.sas** program and run it to test your code.
   a. Change the value of the macro variable **logpath** to the location where you saved your log file.
   b. Change the value of the macro variable **filename** to the name of log file.

```
/*Note: update the location and filename as necessary*/
%let logpath=s:\saswork\logs;
%let filename=test.log
```