Course Guide

Advanced Data Preparation Using IBM SPSS Modeler (v18.1.1)

Course code 0A058 ERC 1.0

IBM Training

January 2018

Contents

Course overview

Overview

This course covers advanced topics to aid in the preparation of data for a successful data science project. You will learn how to use functions, deal with missing values, use advanced field operations, handle sequence data, apply advanced sampling methods, and improve efficiency.

Intended audience

This advanced course is intended for anyone who wants to become familiar with the full range of techniques available in IBM SPSS Modeler for data preparation.

Topics covered

Topics covered in this course include:

- Using functions to cleanse and enrich data
- Using additional field transformations
- Working with sequence data
- Sampling, partitioning and balancing data
- Improving efficiency

Course prerequisites

Participants should have:

- Experience using IBM SPSS Modeler including familiarity with the IBM SPSS Modeler environment, creating streams, reading data files, exploring data, setting the unit of analysis, combining datasets, deriving and reclassifying fields, and basic knowledge of modeling.
- Prior completion of the *IBM SPSS Modeler and Data Science* course is recommended.

Document conventions

Conventions used in this guide follow Microsoft Windows application standards, where applicable. As well, the following conventions are observed:

- **Bold**: Bold style is used in demonstration and exercise step-by-step solutions to indicate a user interface element that is actively selected or text that must be typed by the participant.
- *Italic*: Used to reference book titles.
- CAPITALIZATION: All file names, table names, column names, and folder names appear in this guide exactly as they appear in the application.
  To keep capitalization consistent with this guide, type text exactly as shown.

Exercises

Exercise format

Exercises are designed to allow you to work according to your own pace. Content contained in an exercise is not fully scripted out to provide an additional challenge. Refer back to demonstrations if you need assistance with a particular task. The exercises are structured as follows:

The business question section

This section presents a business-type question followed by a series of tasks. These tasks provide additional information to help guide you through the exercise. Within each task, there may be numbered questions relating to the task. Complete the tasks by using the skills you learned in the unit. If you need more assistance, you can refer to the Task and Results section for more detailed instruction.

The task and results section

This section provides a task based set of instructions that presents the question as a series of numbered tasks to be accomplished. The information in the tasks expands on the business case, providing more details on how to accomplish a task. Screen captures are also provided at the end of some tasks and at the end of the exercise to show the expected results.

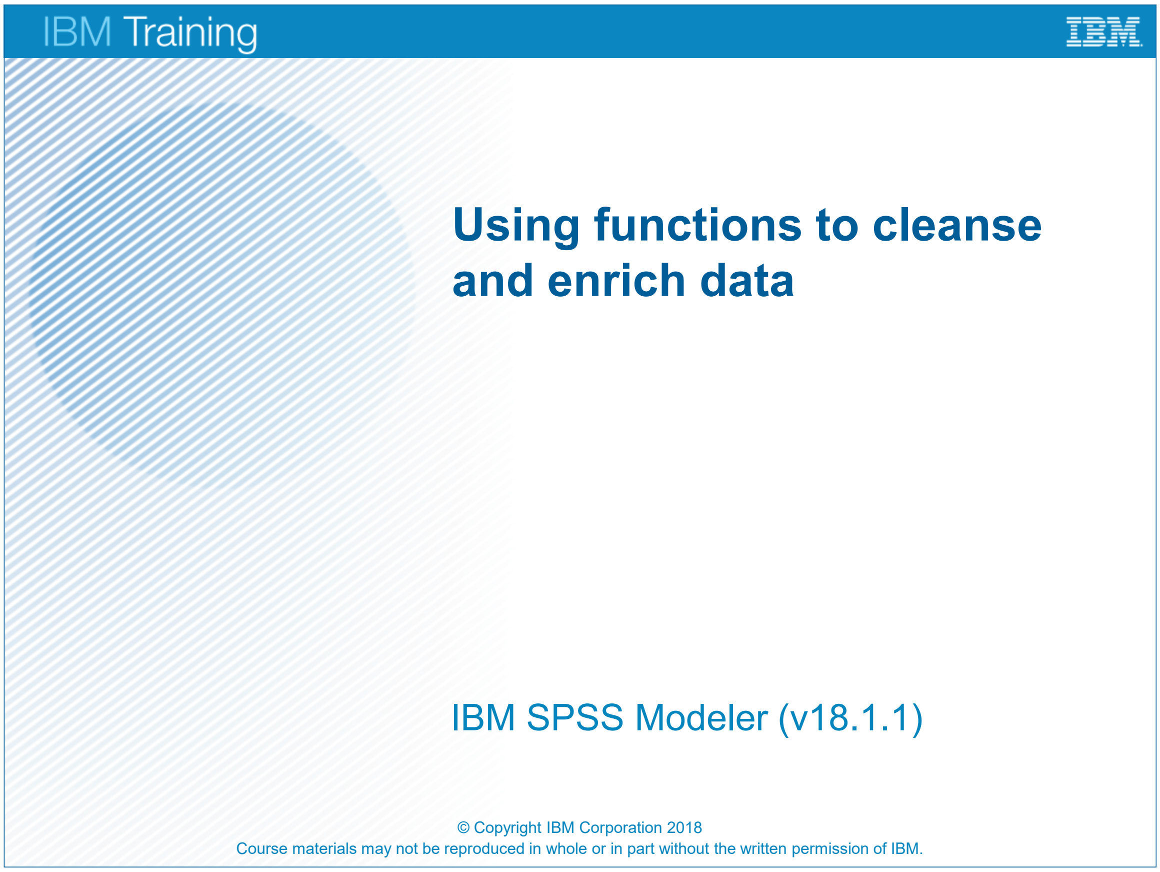
Additional training resources

- Visit IBM Analytics product training and skills validation on the IBM Skills Gateway at http://ibm.com/training/analytics for details on:
  - Instructor-led training in a classroom or online
  - Self-paced training that fits your needs and schedule
  - Comprehensive curricula and training paths that help you identify the courses that are right for you
  - IBM Analytics Certification program and IBM Open Badge program
  - Other resources that will enhance your success with IBM Analytics Software

IBM product help

Help type	When to use	Location
Task-oriented	You are working in the product and you need specific task-oriented help.	*IBM Product* - Help link
Books for Printing (.pdf)	You want to use search engines to find information. You can then print out selected pages, a section, or the whole book. Use Step-by-Step online books (.pdf) if you want to know how to complete a task but prefer to read about it in a book. The Step-by-Step online books contain the same information as the online help, but the method of presentation is different.	Start/Programs/*IBM Product*/Documentation
IBM on the Web	You want to access any of the following:	
	• IBM - Training and Skills Validation	• IBM Skills Gateway at http://ibm.com/training/analytics
	• IBM Analytics Support	• https://www.ibm.com/analytics/resources/support
	• IBM Web site	• http://www.ibm.com

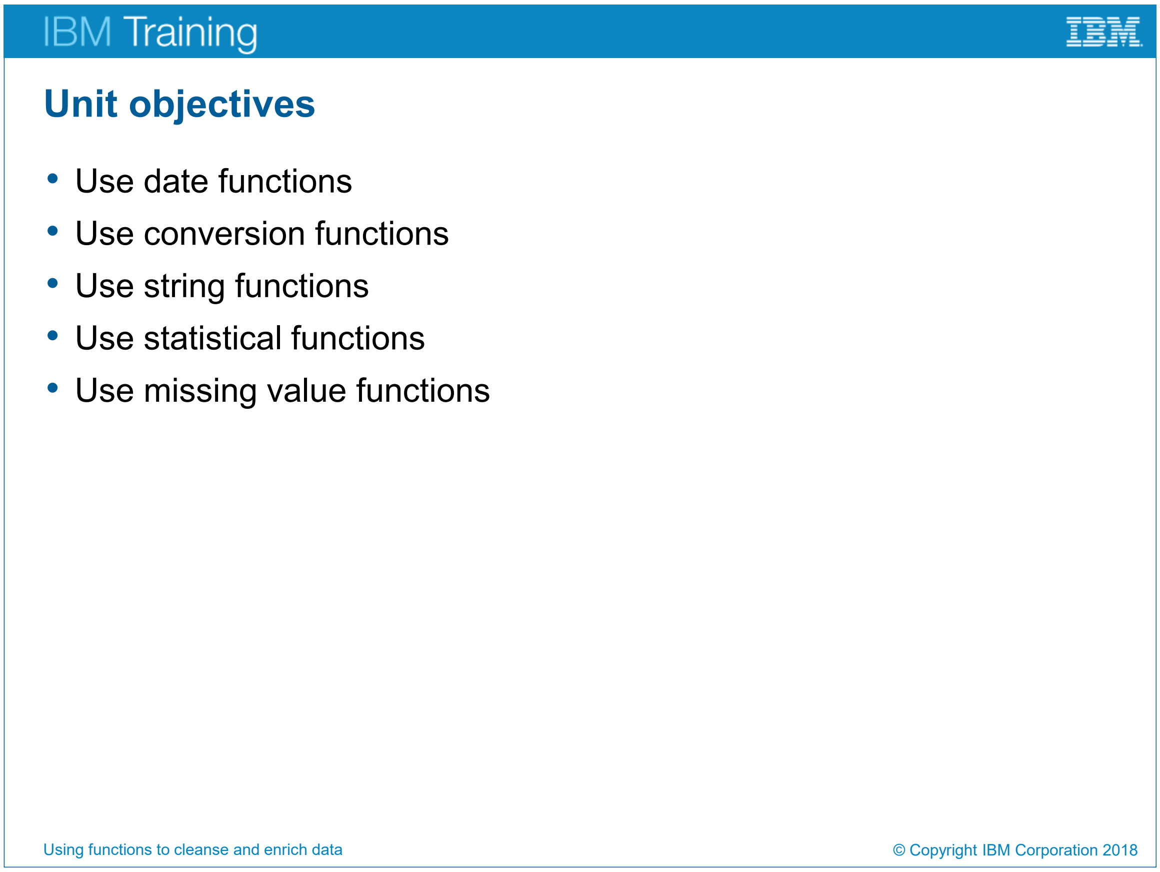
Unit 1 Using functions to cleanse and enrich data



IBM Training

Using functions to cleanse and enrich data

IBM SPSS Modeler (v18.1.1)



IBM Training

Unit objectives

- Use date functions
- Use conversion functions
- Use string functions
- Use statistical functions
- Use missing value functions

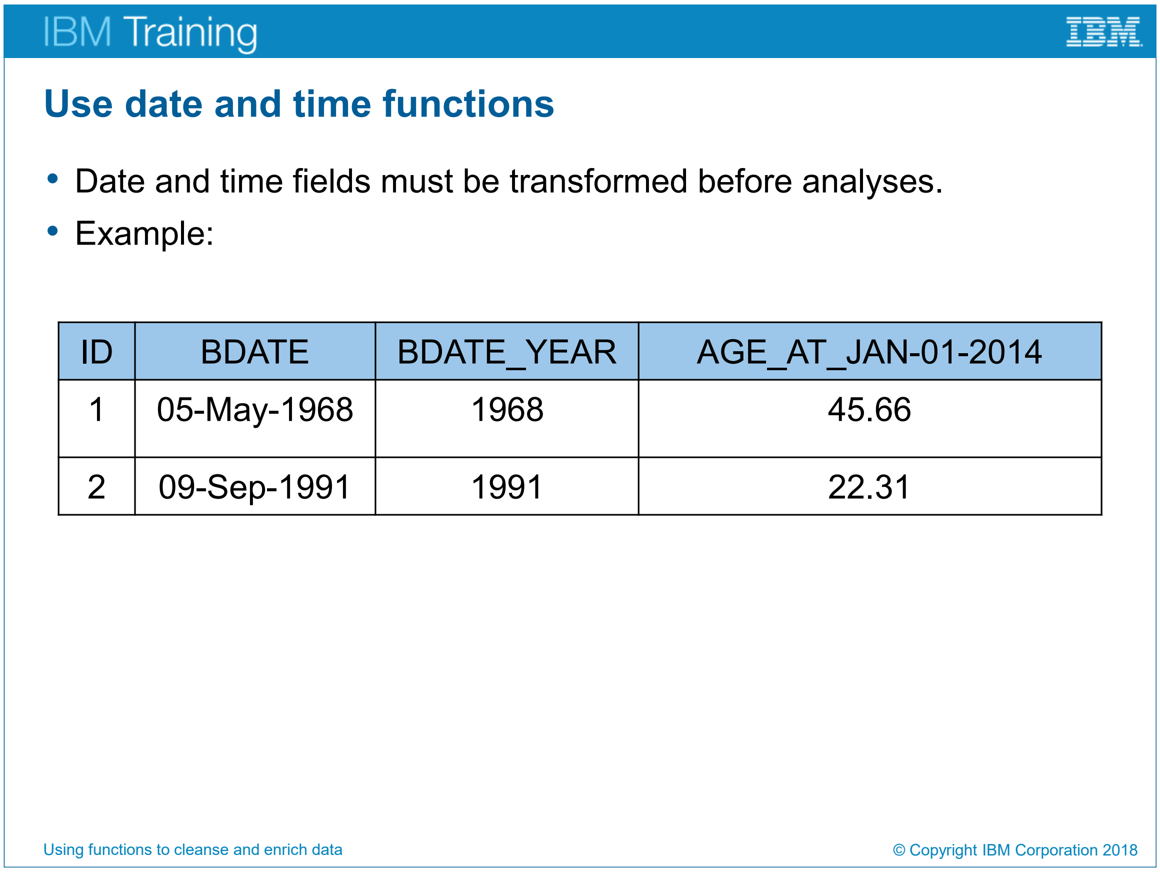
Using functions to cleanse and enrich data

© Copyright IBM Corporation 2018

Unit objectives

Before reviewing this unit you should be familiar with:

- working with IBM SPSS Modeler (streams, nodes, SuperNodes, palettes)
- importing data (Var. File node)
- defining measurement levels and roles, blanks, and instantiating data (Type node)
- selecting fields and records (Filter node, Select node)
- CLEM and the Expression Builder
- deriving fields (Derive node)
- examining the data with graphs (Distribution node, Histogram node) and tables (Table node, Data Audit node, Statistics node, Matrix node, Means node)



IBM Training

Use date and time functions

- Date and time fields must be transformed before analyses.
- Example:

ID	BDATE	BDATE_YEAR	AGE_AT_JAN-01-2014
1	05-May-1968	1968	45.66
2	09-Sep-1991	1991	22.31

Using functions to cleanse and enrich data

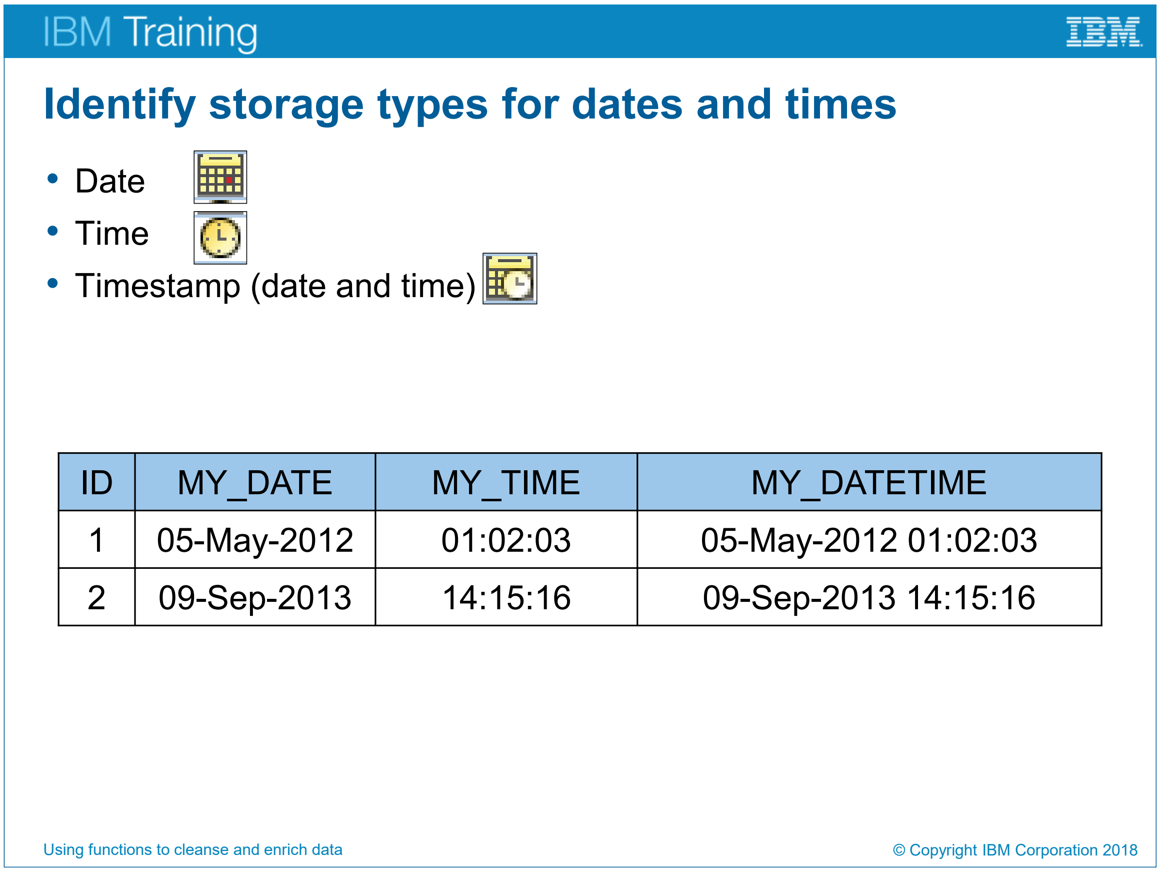
Use date and time functions

Manipulations involving date and time fields are a commonplace task with data.

Date and time functions in IBM SPSS Modeler address issues such as:

- extracting a part from a date or time field such as the year out of a date
- calculating age from date of birth
- calculating length of time that someone has been a customer
- creating a date field from information contained in separate fields

This slide shows an example, with two fields derived from BDATE. The first extracts the year of birth from BDATE, the second age at Jan 1st, 2014.



IBM Training

Identify storage types for dates and times

- Date
- Time
- Timestamp (date and time)

ID	MY_DATE	MY_TIME	MY_DATETIME
1	05-May-2012	01:02:03	05-May-2012 01:02:03
2	09-Sep-2013	14:15:16	09-Sep-2013 14:15:16

Using functions to cleanse and enrich data

Identify storage types for dates and times

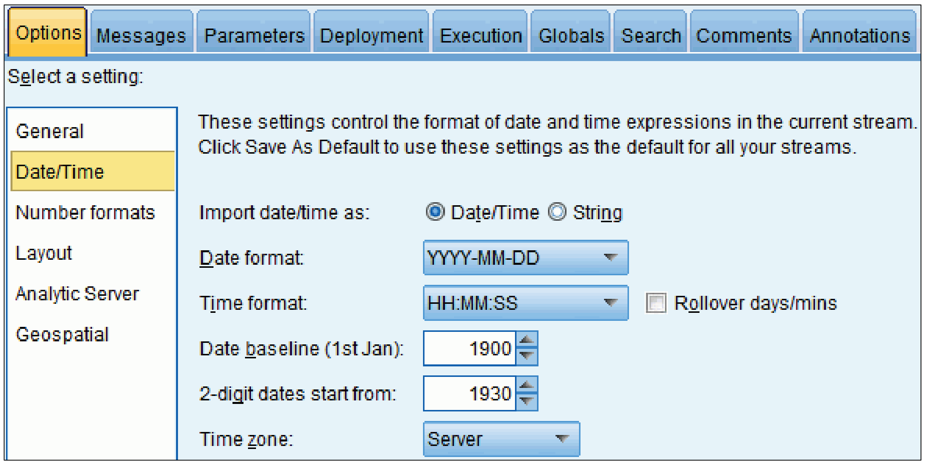
Dates and times fields are numeric fields with special storage types to display the information they contain. IBM SPSS Modeler distinguishes between:

- Date fields: Fields storing a date such as May-05-2016.
- Time fields: Fields storing a time such as 01:02:03.
- Timestamp fields: Fields storing a combination of date and time, such as May-05-2016 01:02:03.

IBM Training

Set options for dates and times

- Defaults for display, defaults for calculations in **Tools\Stream Properties\Options**



Select a setting:

These settings control the format of date and time expressions in the current stream. Click Save As Default to use these settings as the default for all your streams.

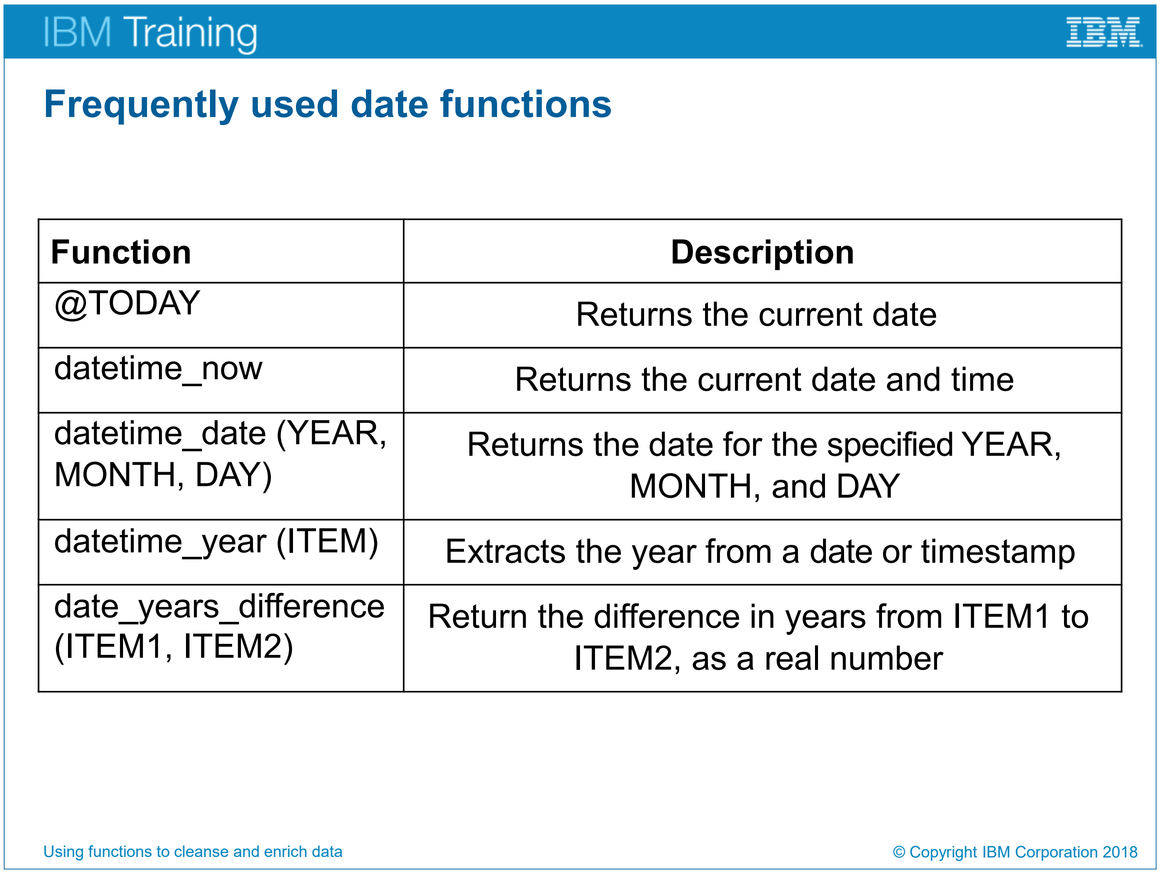
Import date/time as: [✓ Date/Time] String
Date format: YYYY-MM-DD
Time format: HH:MM:SS ☐ Rollover days/mins
Date baseline (1st Jan): 1900
2-digit dates start from: 1930
Time zone: Server

Set options for dates and times

How dates and times are displayed depends on IBM SPSS Modeler's settings. This can be a different format from what it is in the data source. For example, a date field can be stored in a text file in the DD/MM/YYYY format, but when the field is imported in IBM SPSS Modeler it will be displayed according to the specified Date format on the Options tab, Date/Time setting.

For computations involving dates, the Date baseline option sets the baseline year (the baseline month and day is always January 1st). For example, IBM SPSS Modeler's date_in_years function returns the time in years from the baseline date, so date_in_years (DOB) returns 78 for DOB 01-JAN-1978 if the baseline date is 1900.

The 2-digit dates start from option, sets the cutoff year to add century digits for years denoted with two digits. If 1930 is set as the cutoff year, a two-digit year greater than or equal to 30 will be assumed to be in the 20th century. For example, 01-JAN-78 will be set to January 1st of the year 1978. The same setting will use the 21th century for years less than 30; thus, 01-JAN-14 evaluates to January 1st, 2014.



IBM Training

Frequently used date functions

Function	Description
@TODAY	Returns the current date
datetime_now	Returns the current date and time
datetime_date (YEAR, MONTH, DAY)	Returns the date for the specified YEAR, MONTH, and DAY
datetime_year (ITEM)	Extracts the year from a date or timestamp
date_years_difference (ITEM1, ITEM2)	Return the difference in years from ITEM1 to ITEM2, as a real number

Using functions to cleanse and enrich data © Copyright IBM Corporation 2018

Frequently used date functions

This table lists a number of commonly used functions for date and timestamp fields.

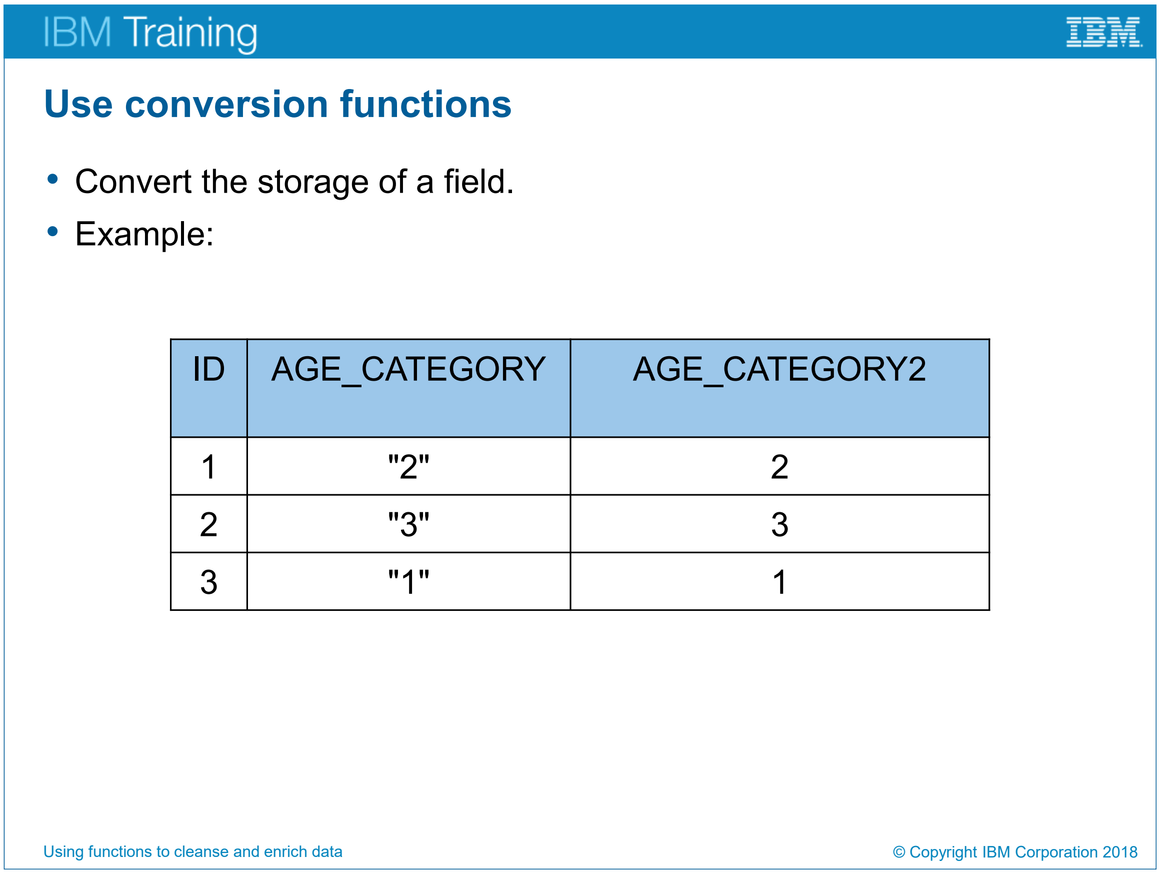
Apart from extracting the year from a date you can extract the month (as a number, long month name, or abbreviated month name) or the day of the week (as a number, long day name, or abbreviated day name).

Analogous to the date_years_difference function you can compute the difference in days, weeks or months.Date functions, like all functions, can be nested and can be used in conjunction with non-date functions. For example, the following expression computes the age, truncated to an integer because of the intof function, at January 1st, 2014:

intof (date_years_difference (DOB, datetime_date (2014, 1, 1)))

Functions analogous to the ones presented on this slide are available for time, such as datetime_hour which extracts the hour from a time or timestamp field, and time_hours_difference which gives the difference in hours between times/timestamps.

Date and time functions are available in the main category Date and Time in the Expression Builder.



IBM Training

Use conversion functions

- Convert the storage of a field.
- Example:

ID	AGE_CATEGORY	AGE_CATEGORY2
1	"2"	2
2	"3"	3
3	"1"	1

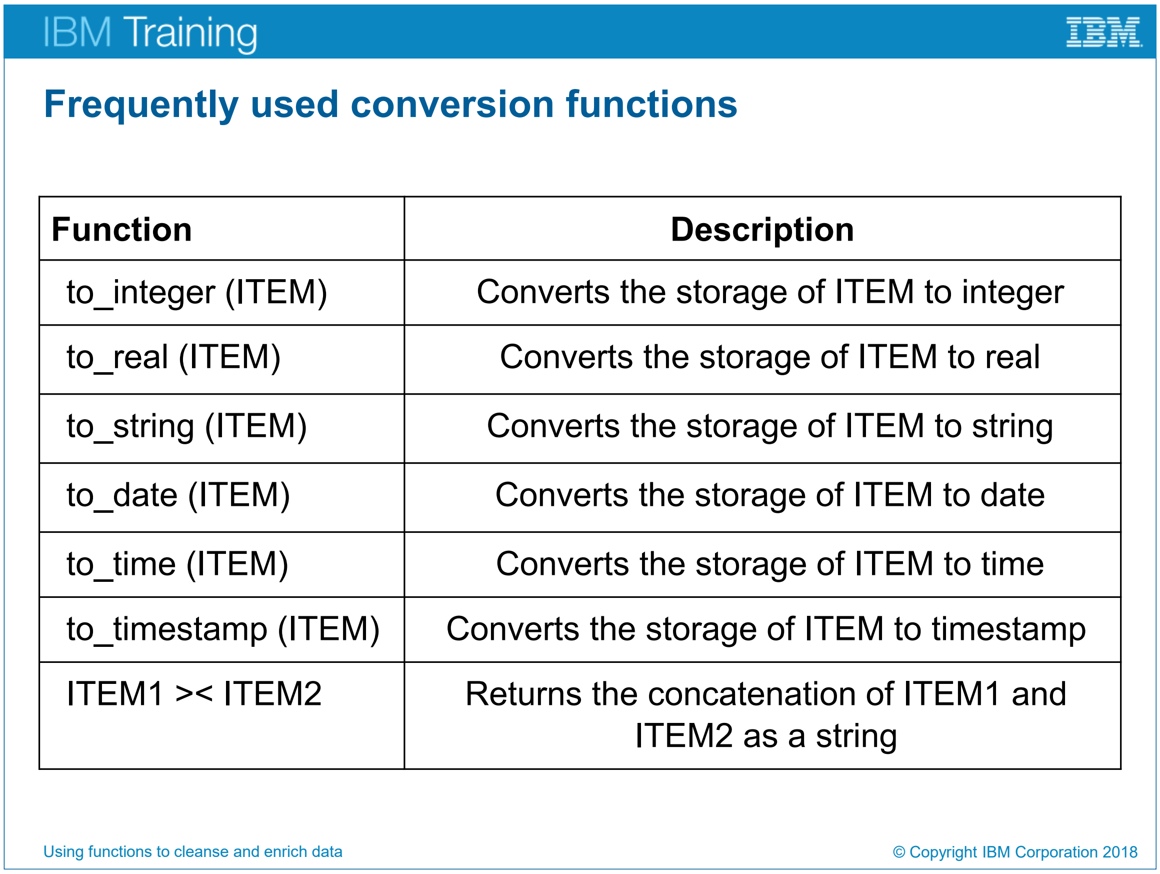
Using functions to cleanse and enrich data © Copyright IBM Corporation 2018

Use conversion functions

A field may have the wrong storage for analyses. This slide shows an example. The AGE_CATEGORY field stores the string values "1", "2", and "3", whereas these values should be integer so that the field can be used as an ordinal field in modeling tasks. The AGE_CATEGORY2 field stores the values as integers.

Other situations where a conversion function is required are:

- convert a real field to an integer field
- convert a timestamp field to a date field



IBM Training

Frequently used conversion functions

Function	Description
to_integer (ITEM)	Converts the storage of ITEM to integer
to_real (ITEM)	Converts the storage of ITEM to real
to_string (ITEM)	Converts the storage of ITEM to string
to_date (ITEM)	Converts the storage of ITEM to date
to_time (ITEM)	Converts the storage of ITEM to time
to_timestamp (ITEM)	Converts the storage of ITEM to timestamp
ITEM1 >< ITEM2	Returns the concatenation of ITEM1 and ITEM2 as a string

Using functions to cleanse and enrich data

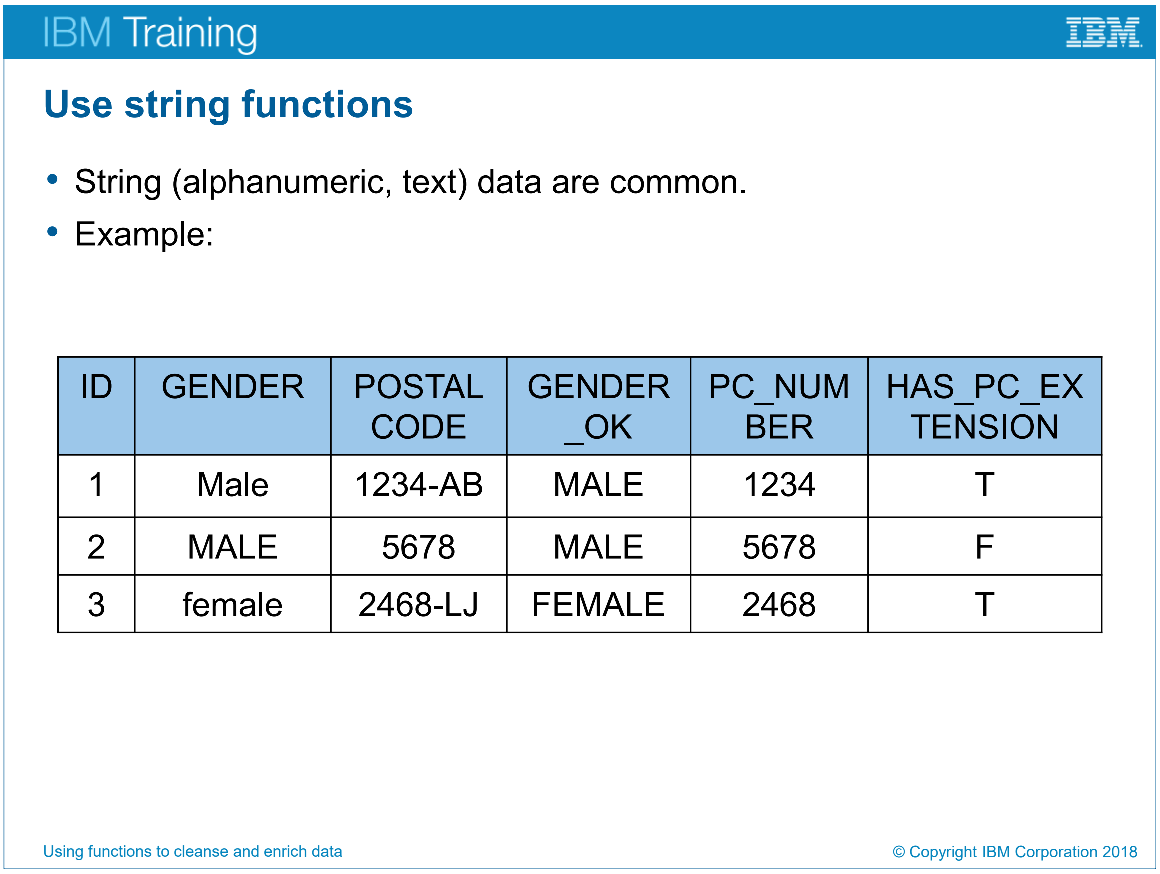
© Copyright IBM Corporation 2018

Frequently used conversion functions

The table on this slide lists frequently used conversion functions. The concatenation function>< is also considered to be a conversion function, because this function will combine fields of any storage into a string field. For example, the expression ZIPCODE_1 >< "-" >< ZIPCODE_2, with ZIPCODE1 and ZIPCODE2 integer fields, returns a string field although none of the source fields are strings.

When you want to change the storage of a field you can either create a new field or overwrite the field. You can use the Derive node or Filler node to create a new field or to overwrite the filed, respectively. The Filler node is presented in the *Using additional field transformations* unit in this course.

Conversion functions are available in the main category Conversion in the Expression Builder.



IBM Training

Use string functions

- String (alphanumeric, text) data are common.
- Example:

ID	GENDER	POSTAL CODE	GENDER _OK	PC_NUM BER	HAS_PC_EX TENSION
1	Male	1234-AB	MALE	1234	T
2	MALE	5678	MALE	5678	F
3	female	2468-LJ	FEMALE	2468	T

Using functions to cleanse and enrich data

© Copyright IBM Corporation 2018

Use string functions

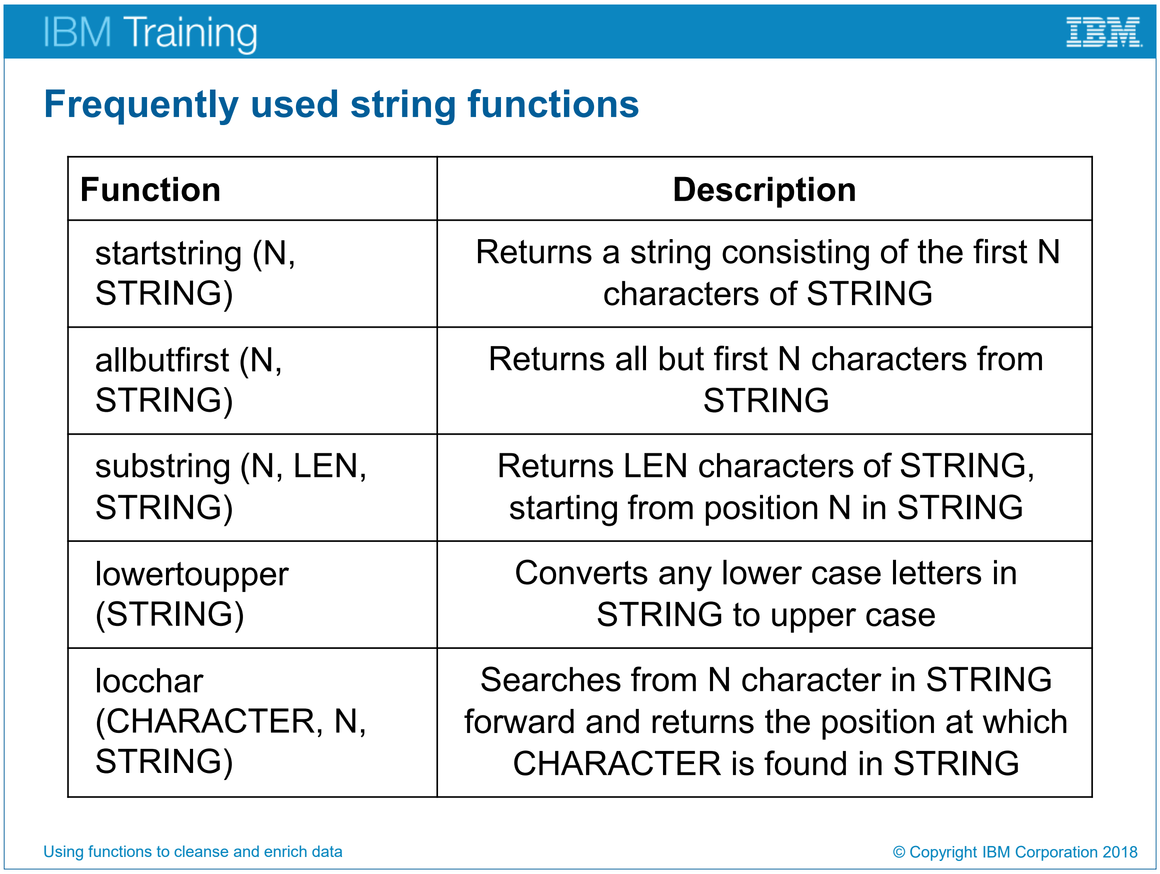
String (or alphanumeric or text) fields, such as names, zip postal codes, and telephone numbers are common in datasets, especially in databases.

IBM SPSS Modeler offers many functions to accommodate working with strings. Tasks that can be completed include:

- correct spelling
- extract a portion of a postal code
- concatenate the first name and the surname into a single field
- extract keywords from a set of text responses

This slide shows an example:

- GENDER_OK replaces the different spellings in GENDER with upper case characters.
- PC_NUMBER extracts the numeric part out of the POSTALCODE field, as an integer.
- HAS_PC_EXTENSION flags if the POSTALCODE field has an extension.



IBM Training

Frequently used string functions

Function	Description
startstring (N, STRING)	Returns a string consisting of the first N characters of STRING
allbutfirst (N, STRING)	Returns all but first N characters from STRING
substring (N, LEN, STRING)	Returns LEN characters of STRING, starting from position N in STRING
lowertoupper (STRING)	Converts any lower case letters in STRING to upper case
locchar (CHARACTER, N, STRING)	Searches from N character in STRING forward and returns the position at which CHARACTER is found in STRING

Using functions to cleanse and enrich data © Copyright IBM Corporation 2018

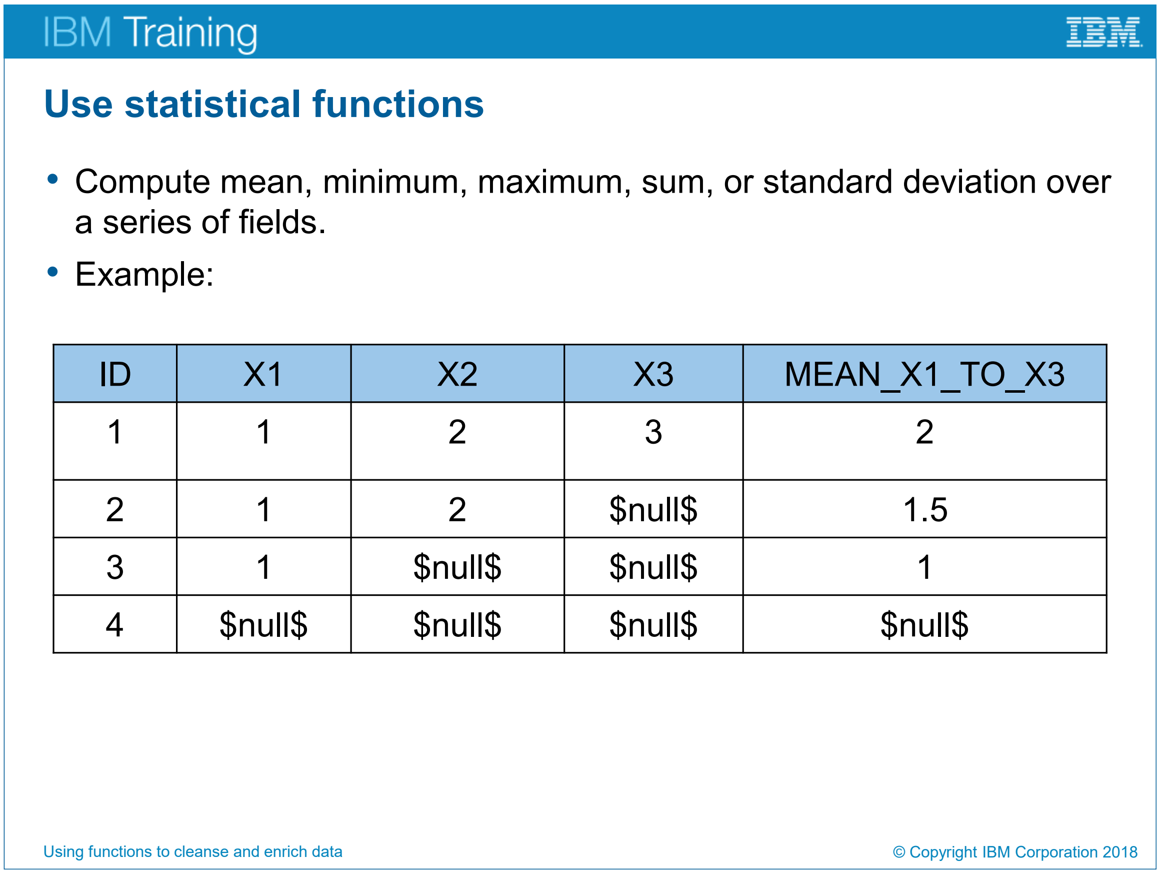
Frequently used string functions

This slide lists only a few of the many string functions that are commonly used. Other frequently used functions are:

- substring_between (N1, N2, STRING): Returns the substring of STRING, which begins at subscript N1 and ends at subscript N2.
- count_substring (STRING, SUBSTRING): Returns the number of instances of SUBSTRING in STRING.
- length (STRING): Returns the number of characters of STRING.
- replace (SUBSTRING1, SUB STRING2, STRING): Replaces SUB STRING1 with SUBSTRING2 in STRING.

When you have text in a function, such as in replace ("Ms", "Miss", NAME), enter the text in double quotes. Important exceptions to this rule are the locchar, locchar_back, stripchar and skipchar functions, where the character must be enclosed within single back quotes. For example, to determine the position of the @ in an e-mail address, specify locchar (`@`, E_MAIL_ADDRESS).

String functions are available in the main category String in the Expression Builder.



IBM Training

Use statistical functions

- Compute mean, minimum, maximum, sum, or standard deviation over a series of fields.
- Example:

ID	X1	X2	X3	MEAN_X1_TO_X3
1	1	2	3	2
2	1	2	$null$	1.5
3	1	$null$	$null$	1
4	$null$	$null$	$null$	$null$

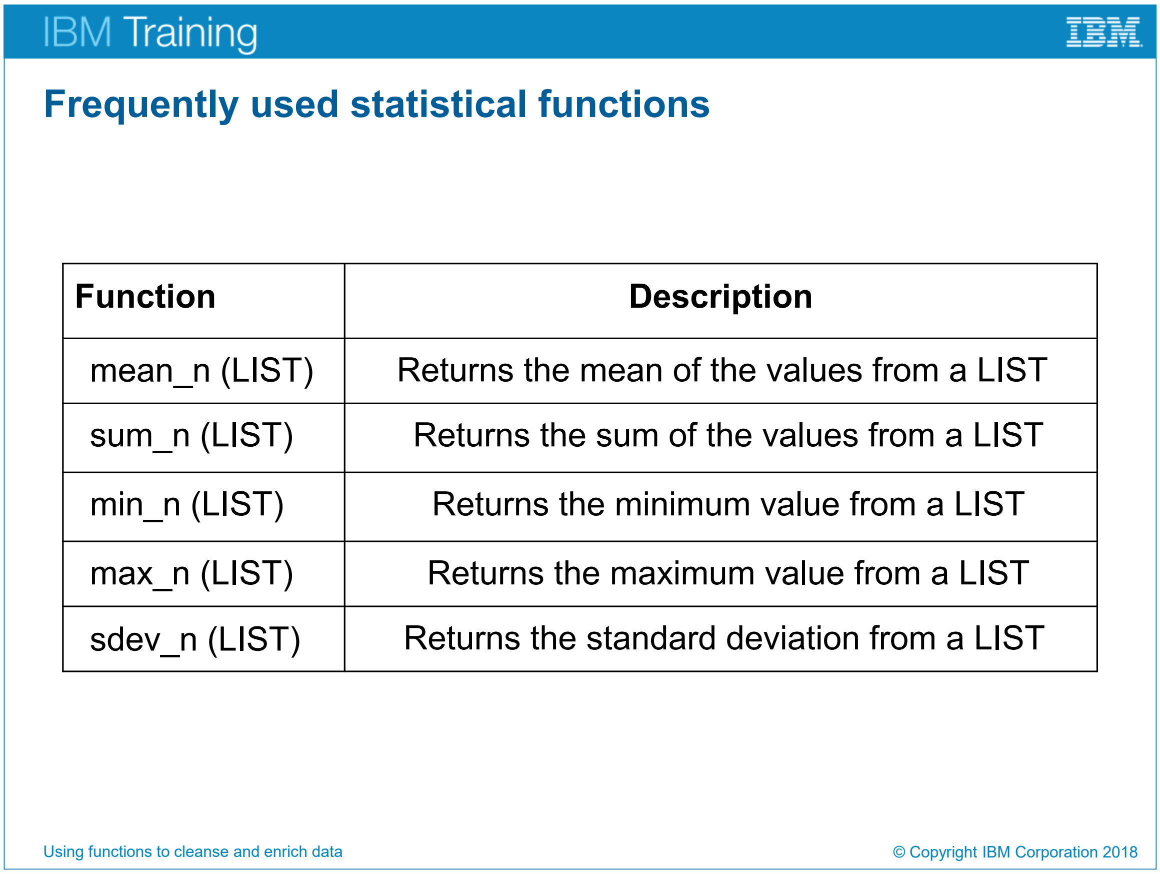
Using functions to cleanse and enrich data

Use statistical functions

Statistical functions are used to calculate a statistic such as the mean or the sum of a series of fields. Notice that these functions compute the result per record, rather than computing the mean or sum over records. For the latter, use the Aggregate node (if you want to have the statistics in your dataset) or the Statistics node (if you only want to know the statistics).

Rather than using a statistical function you can use an arithmetic expression, but the results of using an arithmetic expression can be different from those obtained when a function is used. The calculation of the mean on this slide illustrates the difference. Undefined values are accounted for when you use a statistical function. For example, using the function to compute the mean returns 1.5 for the second record, 1 for the third records, although these records have a $null$ value. When you compute the mean by (X1 + X2 + X3) / 3 the outcome for these records will be undefined ($null$).

This example shows that statistical functions and arithmetic expressions do not return the same result when you have missing values.



IBM Training

Frequently used statistical functions

Function	Description
mean_n (LIST)	Returns the mean of the values from a LIST
sum_n (LIST)	Returns the sum of the values from a LIST
min_n (LIST)	Returns the minimum value from a LIST
max_n (LIST)	Returns the maximum value from a LIST
sdev_n (LIST)	Returns the standard deviation from a LIST

Using functions to cleanse and enrich data

© Copyright IBM Corporation 2018

Frequently used statistical functions

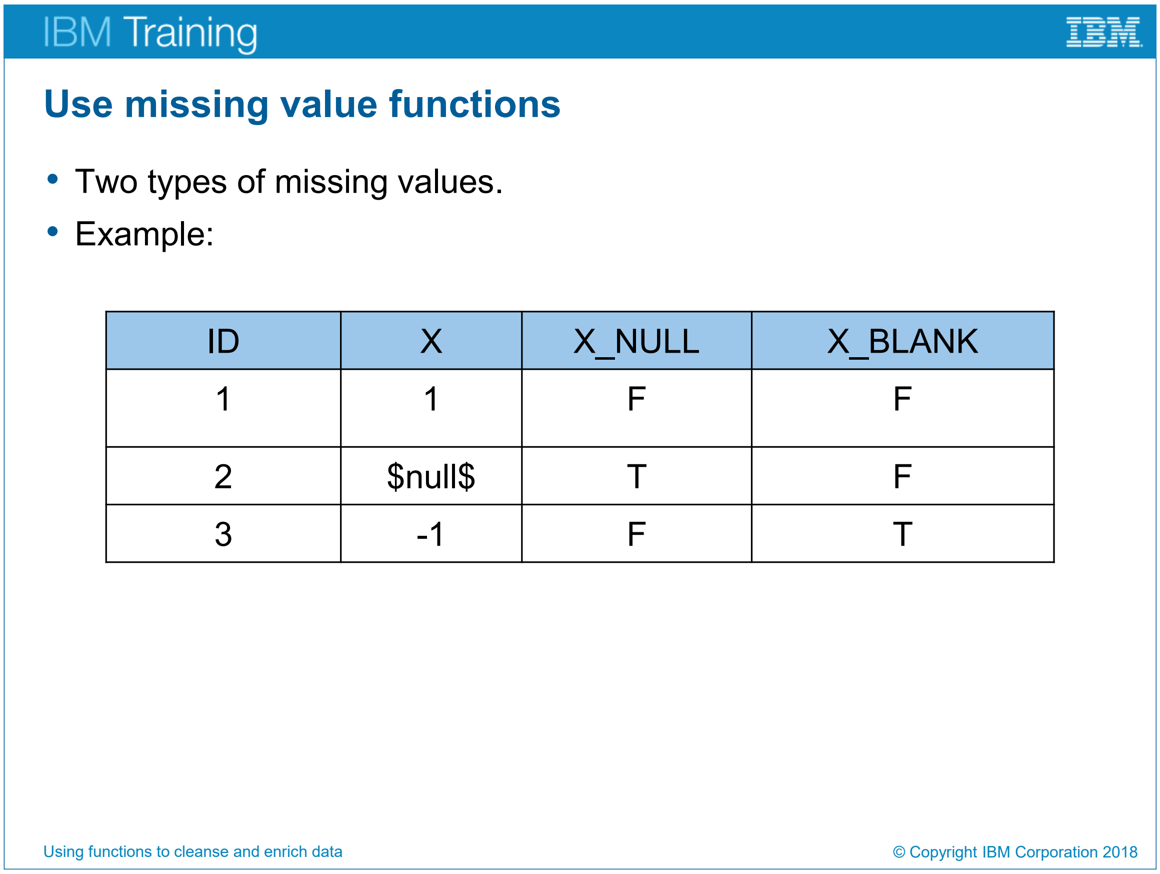
Statistical functions require a list of fields from which to compute the statistic. There are several ways to specify the list:

- Enclose the fields in square brackets.
- Use the @FIELDS_BETWEEN function, and then specify the first field and the last field in the list, separated by a comma.
- Use the @FIELDS_MATCHING function, defining all fields whose name match the supplied pattern, where a question mark (?) in the pattern matches one character and an asterisk (*) matches 0 or more characters.

For example, the following three expressions compute the mean for X1 to X3 (assuming that X1, X2 and X3 are adjacent in the dataset and that there are no other X fields in the dataset):

```
mean_n ([X1 X2 X3])
mean_n (@FIELDS_BETWEEN (X1, X3))
mean_n (@FIELDS_MATCHING ("X?")
```

The mean_n, sum_n, and sdev_n functions are available under Numeric in the Expression Builder. The min_n and max_n functions are listed under Comparison.



IBM Training

Use missing value functions

- Two types of missing values.
- Example:

ID	X	X_NULL	X_BLANK
1	1	F	F
2	$null$	T	F
3	-1	F	T

Using functions to cleanse and enrich data

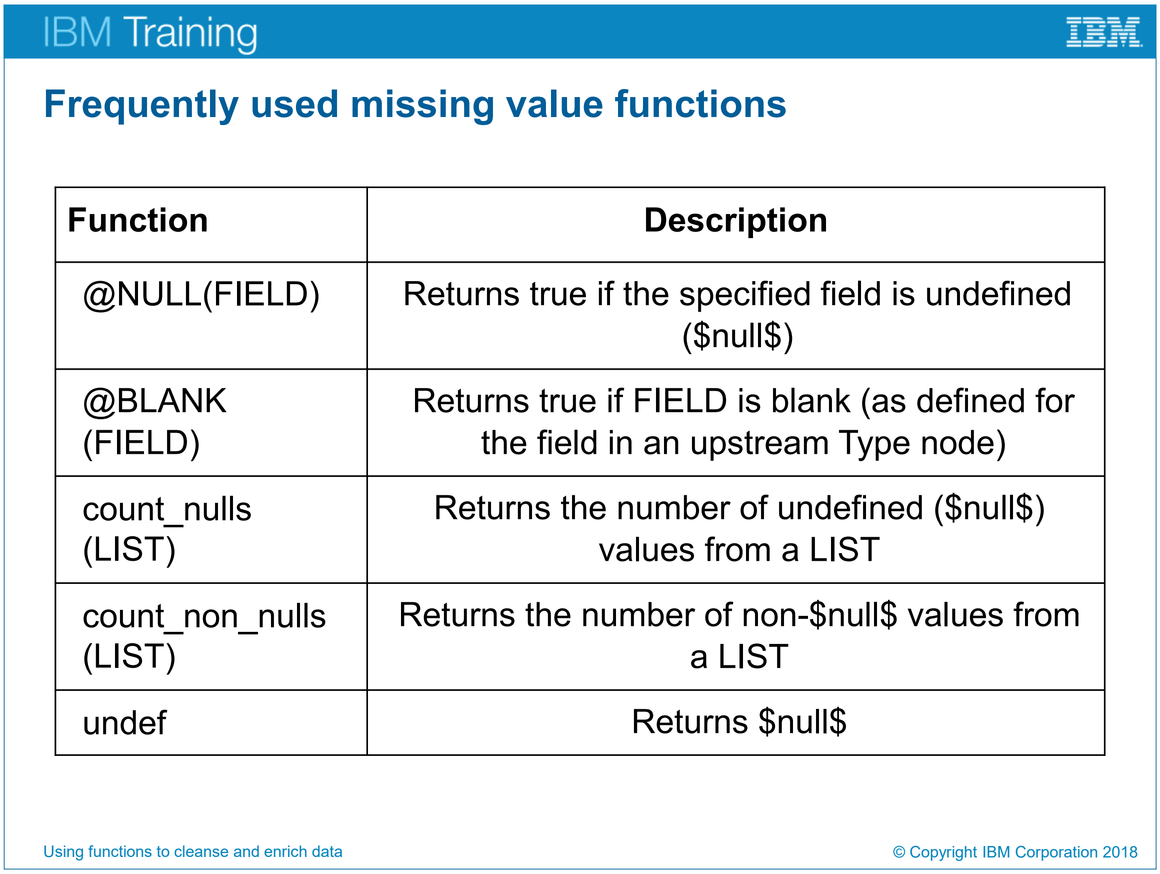
© Copyright IBM Corporation 2018

Use missing value functions

In any project, you will encounter missing values. IBM SPSS Modeler distinguishes between undefined ($null$) values and blank values, also referred to as user-defined missing values, such as a value -1 for AGE.

By default, $null$ values are not regarded as blank values, but you can instruct IBM SPSS Modeler to handle $null$ values as blank values in a Type node. Refer to the *Introduction to IBM SPSS Modeler and Data Science* course for details.

In the example on this slide,-1 was declared as blank for X, but the $null$ value was not declared as blank for this field. X_NULL and X_BLANK flag if X is undefined ($null$) or blank, respectively. X_NULL is only true for the second record. X_BLANK is only true for the third record. When the undefined ($null$) value would have been declared as blank value for X, X_BLANK would also have returned true for the second record.



IBM Training

Frequently used missing value functions

Function	Description
@NULL(FIELD)	Returns true if the specified field is undefined ($null$)
@BLANK (FIELD)	Returns true if FIELD is blank (as defined for the field in an upstream Type node)
count_nulls (LIST)	Returns the number of undefined ($null$) values from a LIST
count_non_nulls (LIST)	Returns the number of non-$null$ values from a LIST
undef	Returns $null$

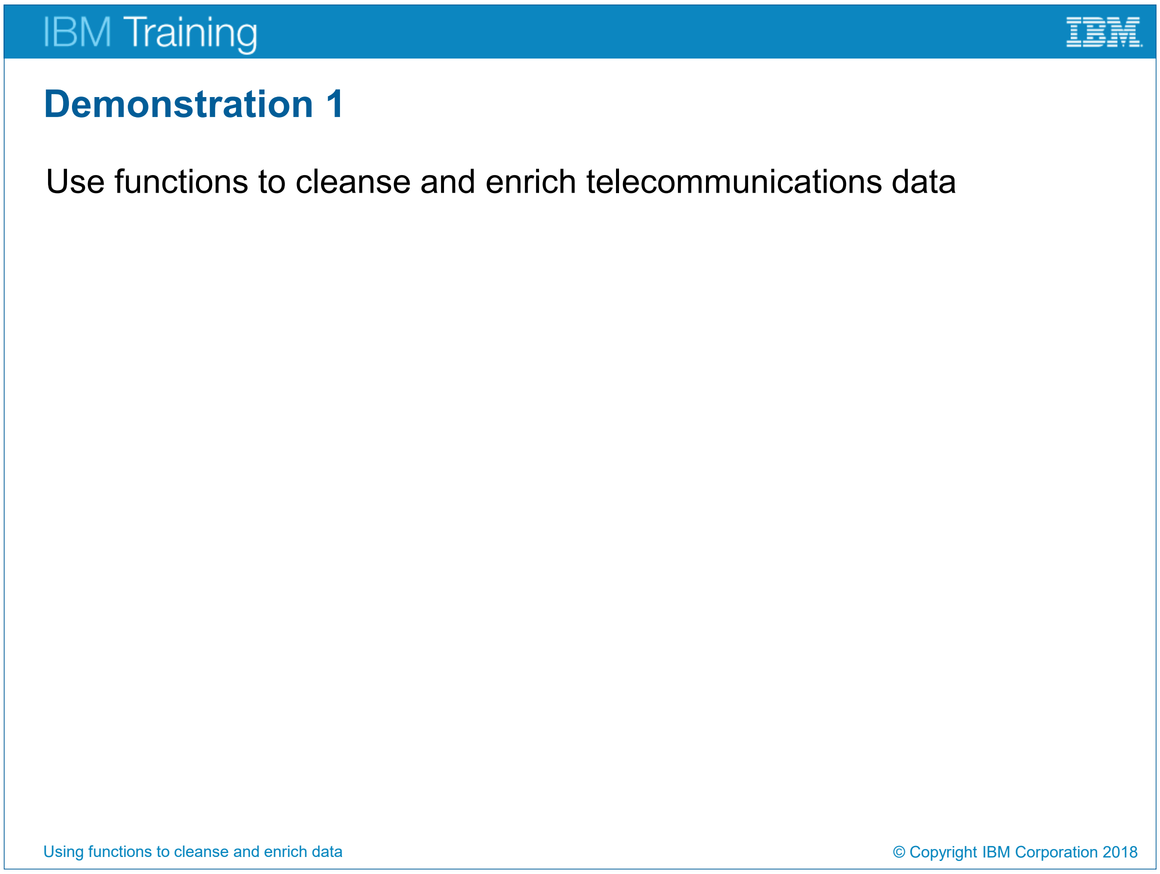
Using functions to cleanse and enrich data © Copyright IBM Corporation 2018

Frequently used missing value functions

The @NULL function and the @BLANK function flag whether the specified field is undefined ($null$) or blank, respectively. As illustrated on the previous slide, the @BLANK function will also return true for undefined ($null$) values, provided that undefined values are declared as blank.

The count_nulls function and count_non_nulls function return the number of $null$ values and the number of non-$null$ values, respectively. There is no function to count the number of blanks on a series of fields.

The @NULL, @BLANK and undef function are listed under Blanks and Null in the Expression Builder. The count_nulls function and the count_non_nulls function are available under Comparison.



IBM Training

Demonstration 1

Use functions to cleanse and enrich telecommunications data

Using functions to cleanse and enrich data

© Copyright IBM Corporation 2018

Demonstration 1: Use functions to cleanse and enrich telecommunications data

Demonstration 1: Use functions to cleanse and enrich telecommunications data

Purpose:
You work for a telecommunications firm where you have to cleanse and enrich a dataset that stores demographic and churn data on the company's customers. Using the transformed dataset, you can build better models later.

Data file:	**telco x subset.csv**
Data folder:	**C:\Training\0A058**

Task 1. Start IBM SPSS Modeler and set the working folder.

The instruction to start IBM SPSS Modeler will depend on the operating system. The following instruction pertains to Microsoft Windows 10. If you have another operating system running, adjust the instruction to that operating system.

1. From the **Start** menu, expand **IBM SPSS Modeler 18.1**, and then click **IBM SPSS Modeler 18.1**. When a splash window displays, click **Cancel**.

   Note: IBM SPSS Modeler 18.1.1 displays "IBM SPSS Modeler 18.1" in the Start menu and program name.

   It is useful to set the working folder when opening and saving files.

2. From the **File** menu, click **Set Directory**, navigate to the **C:\Training\0A058** folder, and then click **Set**.

Task 2. Use date functions to derive fields.

1. From the **Sources** palette, double click the **Var. File** node to add it to the stream canvas.
2. Double-click the **Var. File** node to edit the **Var. File** node.

   Note: From this point forward, the instruction will just be to edit a node.

3. To the right of the **File** field, click the **Browse for file** ... button, navigate to the **C:\Training\0A058** folder if necessary, click the **telco x subset.csv** file, and then click **Open** to import the data. Do not close the **Var. File** dialog box.

   Note: From this point forward, the instruction will just be to import the data file.

4. In the **Var. File** dialog box, click **Preview**, and then scroll to the last fields in the **Preview** output window.

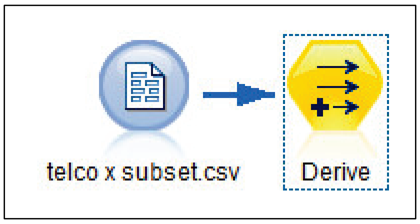
   CONNECT_DATE is the start date of the customer's subscription. END_DATE stores the date that the customer cancelled the subscription; END_DATE is undefined when the customer did not cancel the subscription.
5. Click **OK** to close the **Preview** output window.

   Note: From this point forward, the instruction will just be to close an output window.
6. Click **OK** to close the **Var. File** dialog box.

   Note: From this point forward, the instruction will just be to close a dialog box.

   You will compute the length of stay, in months.
7. From the **Field Ops** palette, double-click the **Derive** node to add it downstream from the **Var. File** node.

   The results appear as follows:

8. Note: Placing node B downstream from node A means that the data flows from A to B. From this point forward, the instruction will just be to add one node downstream from another.
9. Edit the **Derive** node.
10. Under **Derive field**, type **MONTHS_CUSTOMER**.
11. Under **Formula**, enter **date_months_difference (CONNECT_DATE, END_DATE)**.

    Note: Type the expression or use the Expression Builder to construct the expression. In this course "enter" refers to typing or using the Expression Builder, according to your preference. Here, when you use the **Expression Builder**, look for the **date_months_difference** function in the **Date and Time** function group.
12. Click **Preview**, and then scroll to the last fields in the **Preview** output window.

    The new field stores the number of months that elapsed between the two dates, as a real number. If you want the result as an integer, use a function such as round, intof or to_integer.

    Notice that the date_months_difference function returns the undefined ($null$) value when END_DATE is undefined ($null$).
13. Close the **Preview** output window.

14. Close the **Derive** dialog box.

    You will derive a field that stores the month the customer was connected, and a field that stores the month the customer cancelled his subscription. Furthermore, month must be returned as a string, not as a number. For example, the result must be January rather than 1.

    Because the same function can be applied, you can compute the two fields in a single Derive node.

15. From the **Field Ops** palette, add a **Derive** node downstream from the **Derive** node named **MONTHS_CUSTOMER**.

16. Edit the **Derive** node, and then set the **Mode** to **Multiple**.

    The Derive dialog box reflects the change. The Derive field box is replaced by a Derive from box where the source fields are selected.

17. Under **Derive from**, click the **Pick from the set of available fields** button, Ctrl+click **CONNECT_DATE** and **END_DATE**, and then click **OK**.

    Note: From this point forward, the instruction will just be to select the field(s).

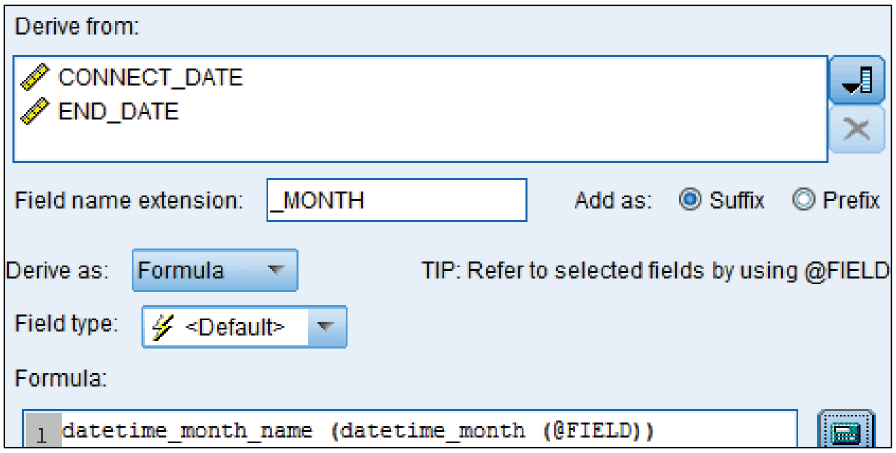
18. Beside **Field name extension**, replace the current extension by **_MONTH**.

    You need the function datetime_month_name (MONTH NUMBER) to get the name of the month. This function cannot be applied directly to a date because this function's argument must be an integer in the range from 1 to 12. Therefore, first extract the month's number from the date using the datetime_month function and then apply the datetime_month_name function.

    Because the formula must be applied to both fields, you will use @FIELD as a substitute for the field names.

19. Under **Formula**, enter **datetime_month_name (datetime_month (@FIELD))**. If you use the **Expression Builder**, locate the **datetime_month** and **datetime_month_name** function in the **Date and Time** function group. Locate the **@FIELD** function in the **@ Functions** function group.

    The results appear as follows:



Derive from:
CONNECT_DATE
END_DATE

Field name extension: _MONTH

Add as: [✓ Suffix] Prefix

Derive as: Formula

TIP: Refer to selected fields by using @FIELD

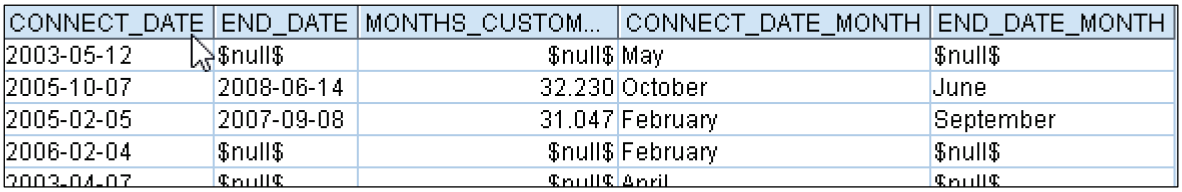
Field type: <Default>

Formula:

```
datetime_month_name (datetime_month (@FIELD))
```

20. Click **Preview**, and then scroll to the last fields in the **Preview** output window.

    The results appear similar to the following:



CONNECT_DATE	END_DATE	MONTHS_CUSTOM...	CONNECT_DATE_MONTH	END_DATE_MONTH
2003-05-12	$null$	$null$	May	$null$
2005-10-07	2008-06-14	32.230	October	June
2005-02-05	2007-09-08	31.047	February	September
2006-02-04	$null$	$null$	February	$null$
2003-04-07	$null$	$null$	April	$null$

    Two fields were created in a single Derive node. The first stores the name of the month in which the customer was connected to the company's network, the second stores the month in which the customer cancelled the subscription (undefined when END_DATE is undefined).

21. Close the **Preview** output window.
22. Close the **Derive** dialog box.

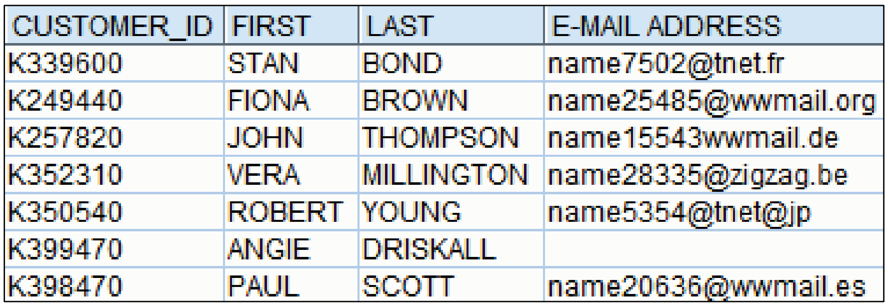
Task 3. Use string functions to derive fields.

You will check the e-mail address for its validity. You will also derive a field that stores the top-level domain name (such as .org).

1. From the **Output** palette, add a **Table** node downstream from the **Derive** node named **_MONTH**.
2. Right-click the **Table** node and then click **Run**.

   Note: From this point forward, the instruction will just be to run the node.

   The results appear similar to the following:



CUSTOMER_ID	FIRST	LAST	E-MAIL ADDRESS
K339600	STAN	BOND	name7502@tnet.fr
K249440	FIONA	BROWN	name25485@wwmail.org
K257820	JOHN	THOMPSON	name15543wwmail.de
K352310	VERA	MILLINGTON	name28335@zigzag.be
K350540	ROBERT	YOUNG	name5354@tnet@jp
K399470	ANGIE	DRISKALL	
K398470	PAUL	SCOTT	name20636@wwmail.es

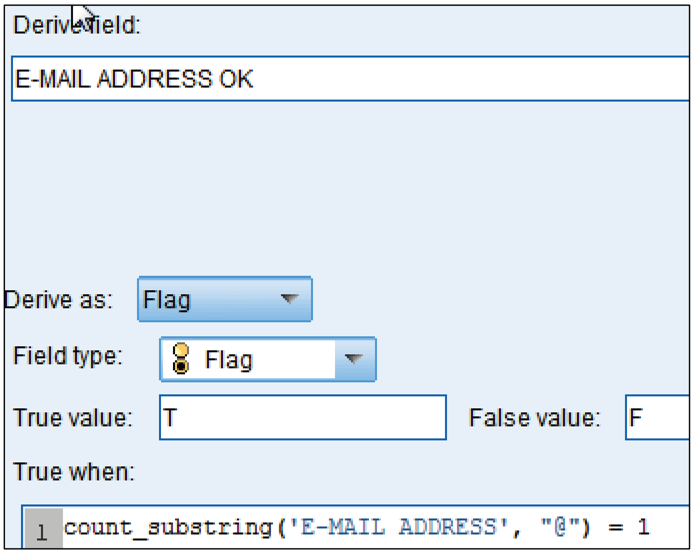
   The values in E-MAIL_ADDRESS do not always represent a valid e-mail address. For example, the third record (CUSTOMER_ID K257820) misses the at sign @ in the address, whereas the fifth record (CUSTOMER_ID K350540) has two. E-MAIL_ADDRESS for the sixth record (CUSTOMER_ID K399470) appears as empty space. A valid e-mail address should have exactly one at sign @.

3. Close the **Table** output window.

   You will derive a field named E-MAIL ADDRESS OK which returns true if the address has exactly one at sign. To derive this field you will need the count_substring (STRING, SUBSTRING) function. This function returns the number of instances of SUBSTRING that are found in STRING.

4. From the **Field Ops** palette, add a **Derive** node downstream from the **Derive** node named _**MONTH** (make room by repositioning the Table node, if preferred).
5. Edit the **Derive** node.
6. Under **Derive field**, type **E-MAIL ADDRESS OK**.
7. Beside **Derive as**, click **Flag** from the list.
8. Under **True when**, enter **count_substring('E-MAIL ADDRESS', "@") = 1**. If you use the **Expression Builder**, locate the **count_substring** function in the **String** function group.

   The results appear as follows:



Derive field: E-MAIL ADDRESS OK

Derive as: Flag

Field type: Flag

True value: T

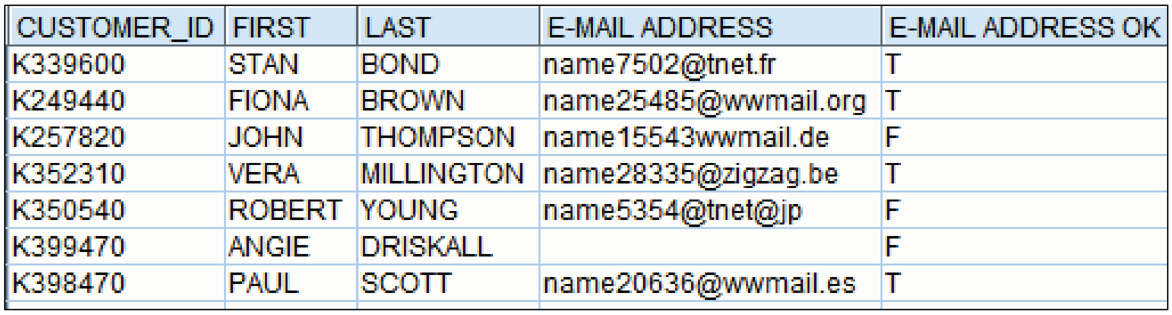
False value: F

True when:

```
count_substring('E-MAIL ADDRESS', "@") = 1
```

9. Click **Preview**, and then move **E-MAIL ADDRESS OK** next to **E-MAIL_ADDRESS** in the **Preview** output window. (Note: move E-MAIL ADDRESS OK by dragging it to the left, until it is just right from E-MAIL ADDRESS.)

   The results appear similar to the following:



CUSTOMER_ID	FIRST	LAST	E-MAIL ADDRESS	E-MAIL ADDRESS OK
K339600	STAN	BOND	name7502@tnet.fr	T
K249440	FIONA	BROWN	name25485@wwmail.org	T
K257820	JOHN	THOMPSON	name15543wwmail.de	F
K352310	VERA	MILLINGTON	name28335@zigzag.be	T
K350540	ROBERT	YOUNG	name5354@tnet@jp	F
K399470	ANGIE	DRISKALL		F
K398470	PAUL	SCOTT	name20636@wwmail.es	T

   Records with exactly one at sign in their e-mail address are true for E-MAIL ADDRESS OK, records with more than one at sign or no at sign at all are false.

10. Close the **Preview** output window.

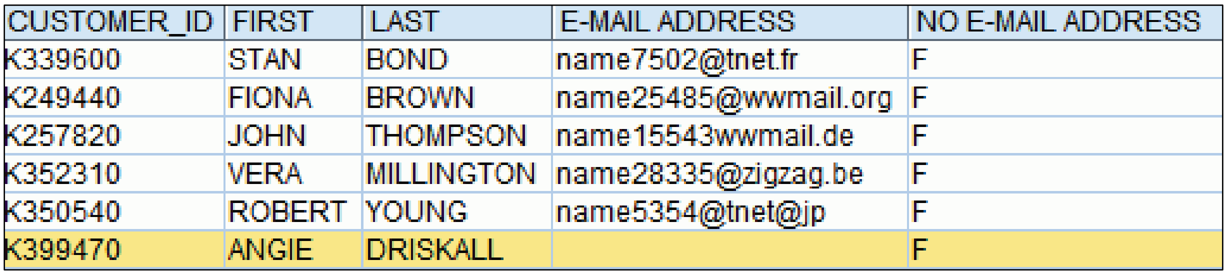
11. Close the **Derive** dialog box.

    Records with an empty e-mail address are of special interest and a field must be derived that flags if there is no e-mail address at all. For example, this field must return true for the sixth record (CUSTOMER_ID K399470).

    There are several ways to accomplish this. You can use the length function to check the length of the e-mail address and to have the function return true if the length is 0. You will explore this option.

12. From the **Field Ops** palette, add a **Derive** node downstream from the **Derive** node named **E-MAIL ADDRESS OK**.
13. Edit the **Derive** node.
14. Under **Derive field**, type **NO E-MAIL ADDRESS**.
15. Beside **Derive as**, click **Flag** from the list.
16. Under **True when**, enter **length ('E-MAIL ADDRESS') = 0**. If you use the **Expression Builder**, locate the **length** function in the **String** function group.
17. Click **Preview**, and then move **NO E-MAIL ADDRESS** next to **E-MAIL ADDRESS**.

    The results appear similar to the following:



CUSTOMER_ID	FIRST	LAST	E-MAIL ADDRESS	NO E-MAIL ADDRESS
K339600	STAN	BOND	name7502@tnet.fr	F
K249440	FIONA	BROWN	name25485@wwmail.org	F
K257820	JOHN	THOMPSON	name15543wwmail.de	F
K352310	VERA	MILLINGTON	name28335@zigzag.be	F
K350540	ROBERT	YOUNG	name5354@tnet@jp	F
K399470	ANGIE	DRISKALL		F

    NO E-MAIL ADDRESS appears to be false for the sixth record. Thus, the length of the e-mail address is greater than 0 for this record. The only explanation can be that the address is comprised of a series of space characters. Thus, the space characters should be removed from the e-mail address first. The trim function is designed for this and can be used in conjunction with the length function to arrive at the required result.

18. Close the **Preview** output window.
19. Under **True when**, enter **length (trim ('E-MAIL ADDRESS')) = 0**. If you use the **Expression Builder**, locate the **length** and **trim** function in the **String** function group.
20. Click **Preview**.

    The sixth record has the true value for the derived field, as required.

21. Close the **Preview** output window.

22. Close the **Derive** dialog box.

    Rather than using the length function combined with the trim function, you could use the iswhitespace function. This function tests whether the specified string is all spaces (zero or more space characters). In the interest of time, this will not be demonstrated. It should be noted that the iswhitespace function is not listed as one of the functions in the Expression Builder, so you would have to type the name of the function.

    As a last example of applying string functions you will derive a field that stores the suffix of the domain name. For example, for an e-mail address such as my_name@tnet.fr it must return fr.

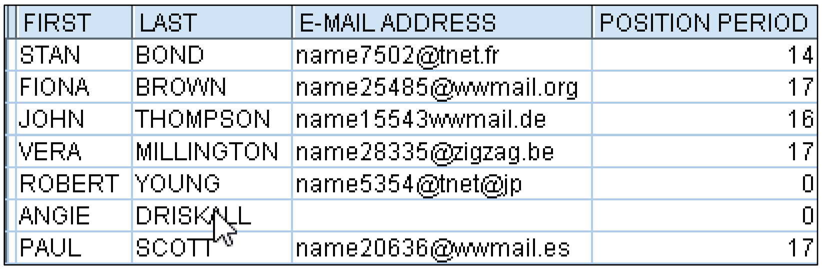
    The approach is based on the idea that it is the part after the last period. Thus, you will first have to locate the period, starting at the end of the e-mail address and then searching backwards. The locchar_back (CHAR, N, STRING) function is designed for this. This function searches backwards from the Nth character of STRING and returns the position at which CHAR is found in STRING. The period is the character to look for and E-MAIL ADDRESS the string to search in. The value of N, the starting position for searching backwards, is not a fixed number but varies with the length of the string. In other words, the value for N must be equal to the length of the e-mail address. The length function does exactly that, as was demonstrated in the previous task.

    Thus, you first will derive a field that gives the position of the last period in the e-mail address.

23. From the **Field Ops** palette, add a **Derive** node downstream from the **Derive** node named **NO E-MAIL ADDRESS**.
24. Edit the **Derive** node.
25. Under **Derive field**, type **POSITION PERIOD**.
26. Under **Formula**, enter **locchar_back (`.`, length ('E-MAIL ADDRESS'), 'E-MAIL ADDRESS')**. If you use the **Expression Builder**, locate the **locchar_back** and **length** function in the **String** function group.

    Note that the period is enclosed within back quotes: `.`.

27. Click **Preview**, and then move **POSITION PERIOD** next to **E-MAIL ADDRESS**.



FIRST	LAST	E-MAIL ADDRESS	POSITION PERIOD
STAN	BOND	name7502@tnet.fr	14
FIONA	BROWN	name25485@wwmail.org	17
JOHN	THOMPSON	name15543wwmail.de	16
VERA	MILLINGTON	name28335@zigzag.be	17
ROBERT	YOUNG	name5354@tnet@jp	0
ANGIE	DRISKALL		0
PAUL	SCOTT	name20636@wwmail.es	17

    POSITION PERIOD stores the position of the last period. When the e-mail address does not contain a period, it returns 0.

28. Close the **Preview** output window.
29. Close the **Derive** dialog box.

    Knowing the position of the last period, there are several ways to proceed. You can use a substring function or a function such as endstring. Here, you will use the substring_between (N1, N2, STRING) function. This function returns the substring of STRING beginning at subscript N1 and ending at subscript N2. In this expression, N1 is the position of the period plus 1, and N2 is the length of the e-mail address.

    The computation will be conditional on whether E-MAIL ADDRESS has a period in it, in which case POSITION PERIOD is greater than 0. Otherwise, the new field must be set to the undefined value ($null$), which is referred to by undef in a CLEM expression.
30. From the **Field Ops** palette, add a **Derive** node downstream from the **Derive** node named **POSITION PERIOD**.
31. Edit the **Derive** node**.**
32. Under **Derive field** type **DOMAIN NAME**.
33. Beside **Derive as**, click **Conditional** from the list.
34. Under **If**, enter **'POSITION PERIOD' > 0**.
35. Under **Then**, enter **substring_between ('POSITION PERIOD' + 1, length ('E-MAIL ADDRESS'), 'E-MAIL ADDRESS')**. If you use the **Expression Builder**, locate the **substring_between** and **length** function in the **String** function group.
36. Under **Else**, type **undef**.
37. Click **Preview**, and then scroll to the last fields in the **Preview** output window.

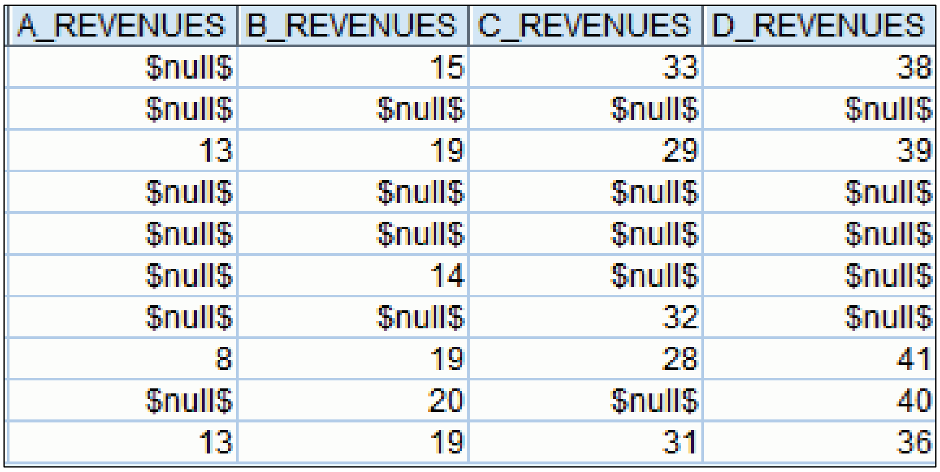
    The new field NAME stores the suffix of the domain name and is undefined when E-MAIL ADDRESS does not have a period in it, as required.
38. Close the **Preview** output window.
39. Close the **Derive** dialog box.

    As noticed, there are more ways to arrive at the same result. You can also use the textsplit function which splits a string at a specified character and takes the 1st part or 2nd part (which is one of the arguments of this function). This, however, assumes that there is only one period in an address which might be a risky assumption.

    Also, it is not necessary to derive a separate field that stores the position of the period in a field, because you can nest functions and create a single expression.

Task 4. Use statistical functions to derive fields.

You will use statistical functions to compute the mean revenues per record and the sum of revenues per record, based on A_REVENUES to D_REVENUES. The table below lists the fields for reference.

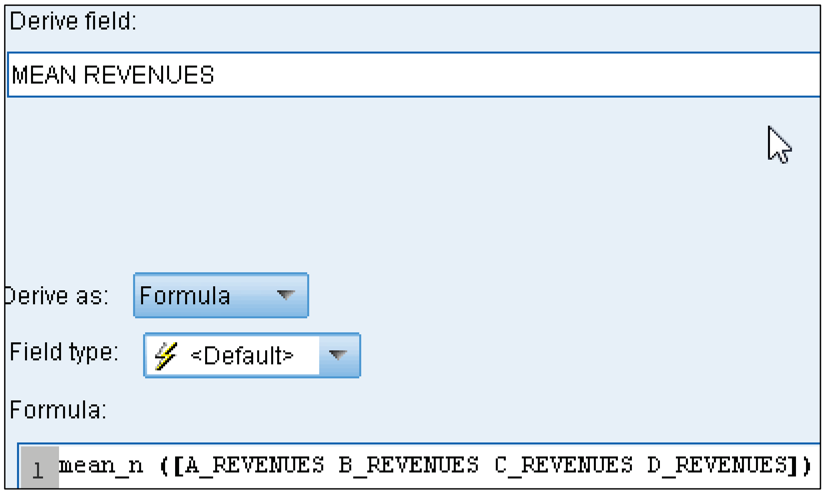


A_REVENUES	B_REVENUES	C_REVENUES	D_REVENUES
$null$	15	33	38
$null$	$null$	$null$	$null$
13	19	29	39
$null$	$null$	$null$	$null$
$null$	$null$	$null$	$null$
$null$	14	$null$	$null$
$null$	$null$	32	$null$
8	19	28	41
$null$	20	$null$	40
13	19	31	36

For example, the third record has mean revenues (13 + 19 + 29 + 39) / 4 = 25.

1. From the **Field Ops** palette, add a **Derive** node downstream from the **Derive** node named **DOMAIN NAME**.
2. Edit the **Derive** node.
3. Under **Derive field**, enter **MEAN REVENUES**.
4. Under **Formula**, enter **mean_n ([A_REVENUES B_REVENUES C_REVENUES D_REVENUES])**. If you use the **Expression Builder**, locate the **mean_n** function in the **Numeric** function group.

   The results appear as follows:



Derive field: MEAN REVENUES

Derive as: Formula

Field type: <Default>

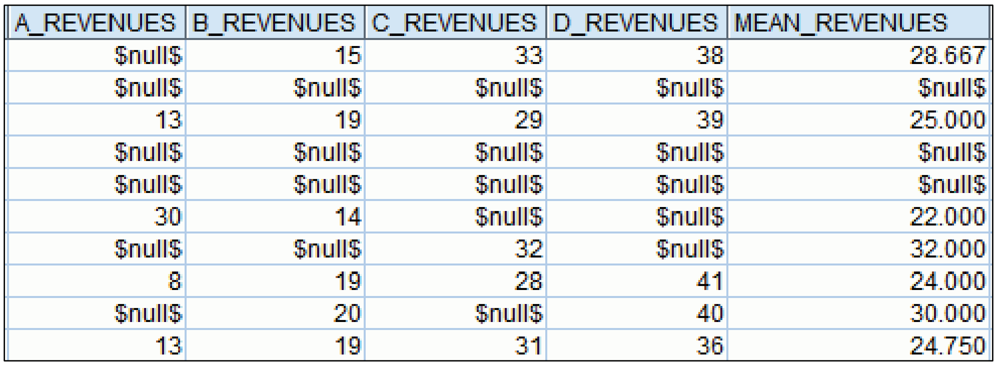
Formula:

```
mean_n ([A_REVENUES B_REVENUES C_REVENUES D_REVENUES])
```

Notice that the list is enclosed within square brackets, and that the fields are separated by a space only.

5. Click **Preview**, and then move **MEAN REVENUES** next to **D_REVENUES**.

   The results appear as follows:



A_REVENUES	B_REVENUES	C_REVENUES	D_REVENUES	MEAN_REVENUES
$null$	15	33	38	28.667
$null$	$null$	$null$	$null$	$null$
13	19	29	39	25.000
$null$	$null$	$null$	$null$	$null$
$null$	$null$	$null$	$null$	$null$
30	14	$null$	$null$	22.000
$null$	$null$	32	$null$	32.000
8	19	28	41	24.000
$null$	20	$null$	40	30.000
13	19	31	36	24.750

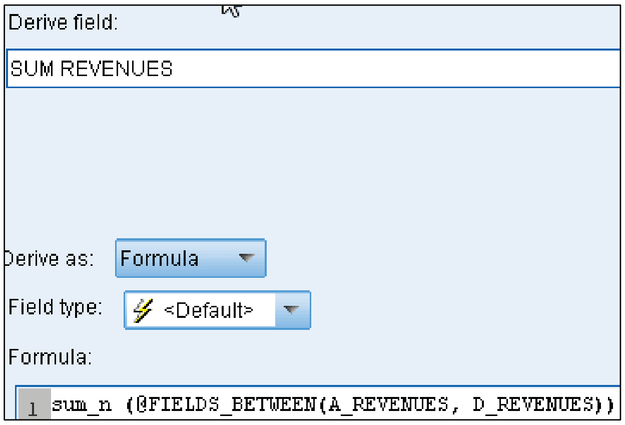
   The mean_n function returned the mean computed over the valid values, and an undefined value when all source fields are undefined.

6. Close the **Preview** output window.
7. Close the **Derive** dialog box.

   Next to computing the mean you will compute the sum. Rather than specifying the source fields by name, you will use @FIELDS_BETWEEN to reference them.

8. From the **Field Ops** palette, add a **Derive** node downstream from the **Derive** node named **MEAN REVENUES**.
9. Edit the **Derive** node.
10. Under **Derive field**, type **SUM REVENUES**.
11. Under **Formula**, enter **sum_n (@FIELDS_BETWEEN(A_REVENUES, D_REVENUES))**. If you use the **Expression Builder**, locate the **sum_n** function in the **Numeric** function group; locate the **@FIELDS_BETWEEN** function in the **@Functions** function group.

    The results appear as follows:



Derive field:
SUM REVENUES

Derive as: Formula

Field type: <Default>

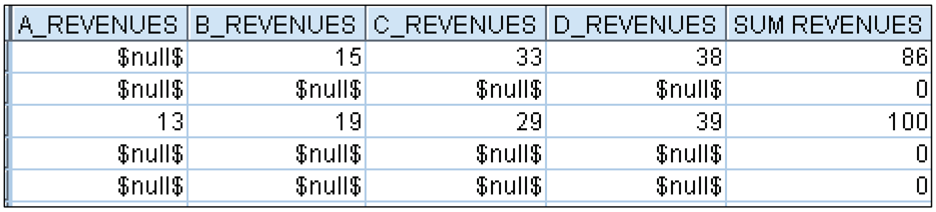
Formula:

```
sum_n (@FIELDS_BETWEEN(A_REVENUES, D_REVENUES))
```

Notice that the fields are not enclosed in square brackets when you use the @FIELDS_BETWEEN function.

12. Click **Preview**, and then move **SUM REVENUES** next to **D_REVENUES**.

    The results appear as follows:



A_REVENUES	B_REVENUES	C_REVENUES	D_REVENUES	SUM REVENUES
$null$	15	33	38	86
$null$	$null$	$null$	$null$	0
13	19	29	39	100
$null$	$null$	$null$	$null$	0
$null$	$null$	$null$	$null$	0

    The sum_n function returns 0 when all source fields are undefined, whereas the mean_n function returned the undefined value. In the next task you will compute the sum conditionally and assign the undefined value to the sum when all source fields are undefined.

13. Close the **Preview** output window.
14. Close the **Derive** dialog box.

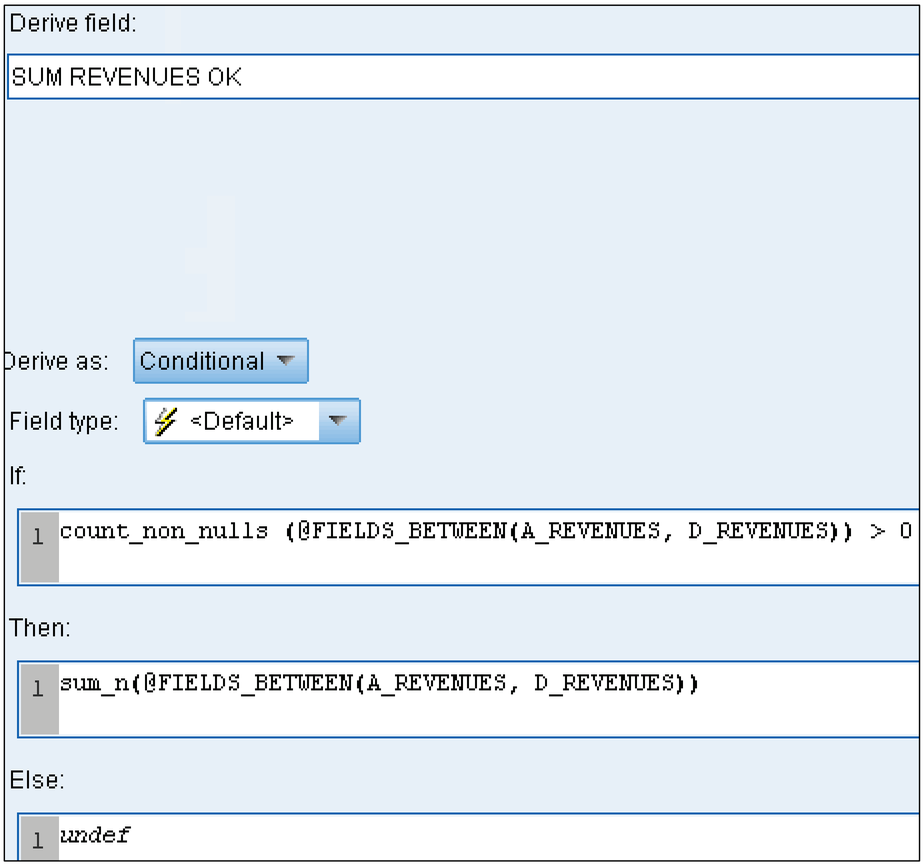
Task 5. Use missing values functions.

You will compute the sum of A_REVENUES to D_REVENUES so that the result is undefined when all source fields are undefined. This means that the sum only has to be computed when the number of non-null values is greater than 0, for which the count_non_nulls function can be used. If the number of non-null values is 0 you will assign the undefined ($null$) value to the new field.

1. From the **Field Ops** palette, add a **Derive** node downstream from the **Derive** node named **SUM REVENUES**.
2. Edit the **Derive** node.
3. Under **Derive field**, enter **SUM REVENUES OK**.
4. Beside **Derive As**, select **Conditional**.
5. Under **If**, enter **count_non_nulls (@FIELDS_BETWEEN(A_REVENUES, D_REVENUES)) > 0**. If you use the **Expression Builder**, locate the **count_non_nulls** function in the **Comparison** function group; locate the **@FIELDS_BETWEEN** function in the **@Functions** function group.
6. Under **Then**, enter **sum_n(@FIELDS_BETWEEN(A_REVENUES, D_REVENUES))**.

7. Under **Else**, enter **undef**.

   The results appear as follows:



Derive field:

SUM REVENUES OK

Derive as: Conditional

Field type: <Default>

If:

```
count_non_nulls (@FIELDS_BETWEEN(A_REVENUES, D_REVENUES)) > 0
```

Then:

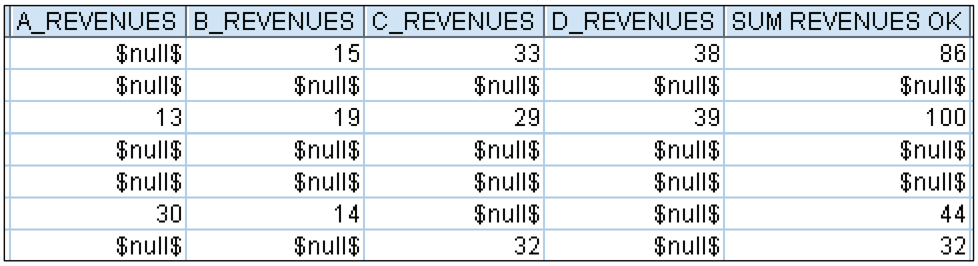
```
sum_n(@FIELDS_BETWEEN(A_REVENUES, D_REVENUES))
```

Else:

```
undef
```

8. Click **Preview**, and move **SUM REVENUES OK** next to **D_REVENUES**.

   The results appear as follows:



A_REVENUES	B_REVENUES	C_REVENUES	D_REVENUES	SUM REVENUES OK
$null$	15	33	38	86
$null$	$null$	$null$	$null$	$null$
13	19	29	39	100
$null$	$null$	$null$	$null$	$null$
$null$	$null$	$null$	$null$	$null$
30	14	$null$	$null$	44
$null$	$null$	32	$null$	32

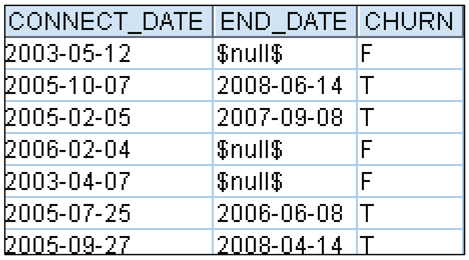
   The new field is undefined when all source fields are undefined, as required.

9. Close the **Preview** output window.
10. Close the **Derive** dialog box.

    As a last example of using missing value functions, derive a field CHURN that flags whether the customer has cancelled the subscription. Notice that END_DATE is not undefined when the customer cancelled.
11. From the **Field Ops** palette, add a **Derive** node downstream from the **Derive** node named **SUM REVENUES OK**.
12. Edit the **Derive** node.

13. Under **Derive field**, enter **CHURN**.
14. Beside **Derive As**, click **Flag** from the list.
15. Under **True when**, enter **not (@NULL (END_DATE))**. If you use the **Expression Builder**, locate the **@NULL** function in the **@ Functions** function group.
16. Click **Preview**, and then move **CHURN** next to **END_DATE**.

    The results appear as follows:

CONNECT_DATE	END_DATE	CHURN
2003-05-12	$null$	F
2005-10-07	2008-06-14	T
2005-02-05	2007-09-08	T
2006-02-04	$null$	F
2003-04-07	$null$	F
2005-07-25	2006-06-08	T
2005-09-27	2008-04-14	T

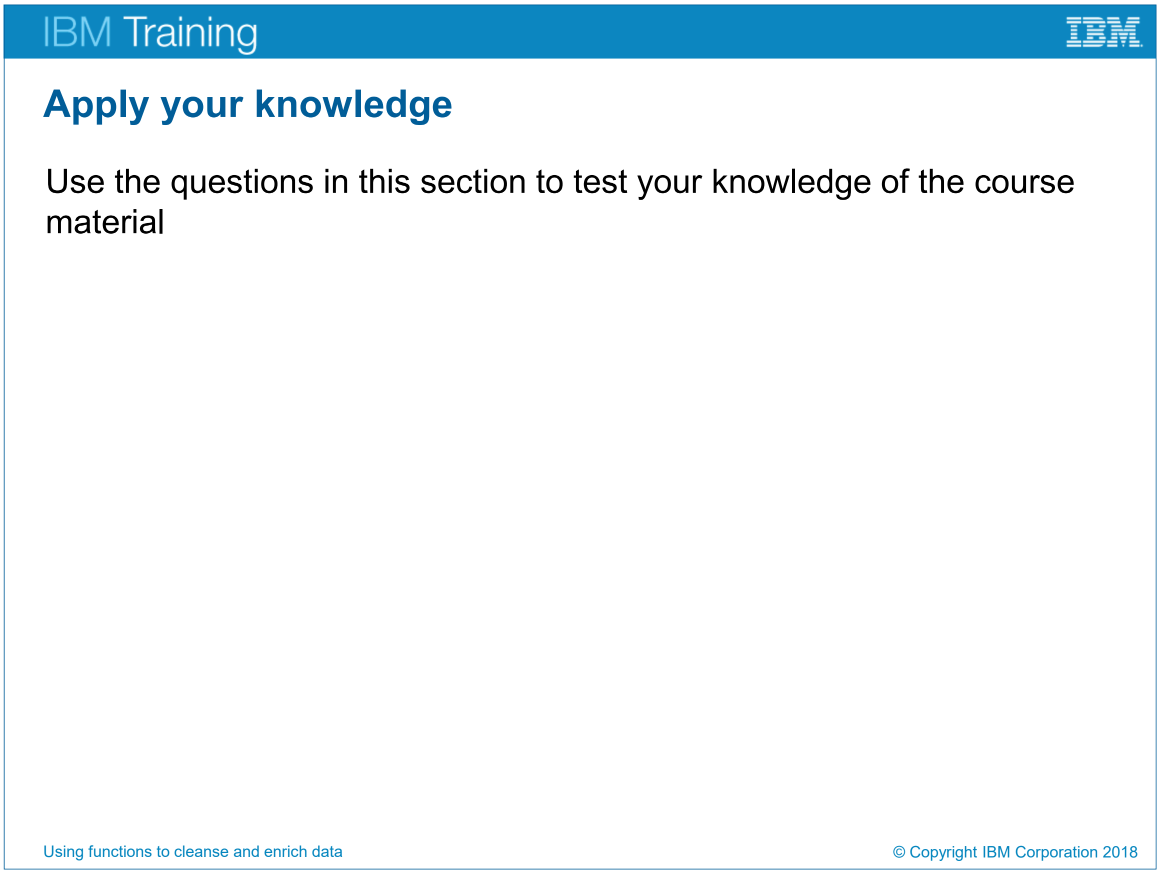
    CHURN is true for those with a non-null end date, else false, as required.
17. Close the **Preview** output window.
18. Close the **Derive** dialog box.

    This completes the demonstration. You will create a clean state for the exercises.
19. From the **File** menu, click **Close Stream**. Click **No** when asked to save the stream.
20. From the **File** menu, click **New Stream**.

    Leave IBM SPSS Modeler open for the exercises.

Results:
You have used functions to cleanse and enrich the data, so that better models can be built later.

You will find the completed stream in the **C:\Training\0A058\Solutions** folder.


IBM Training

IBM

Apply your knowledge

Use the questions in this section to test your knowledge of the course material

Using functions to cleanse and enrich data

© Copyright IBM Corporation 2018

Apply your knowledge

Question 1: What does the expression **date_years_difference (DOB, datetime_date (1, 1, 2014))** compute? (DOB stores date of birth.)

A. Age at JAN-01-2014.

B. Age at today's date.

C. It produces undefined ($null$) values because the order of arguments in the datetime_date function is incorrect.

D. It produces undefined ($null$) values because the order of two functions must be reversed.

Note: The following functions are used:

datetime_date (YEAR, MONTH, DAY). Returns the date value for the given YEAR, MONTH, and DAY.

date_years_difference (DATE1, DATE2). Returns the difference in years from DATE1 to DATE2.

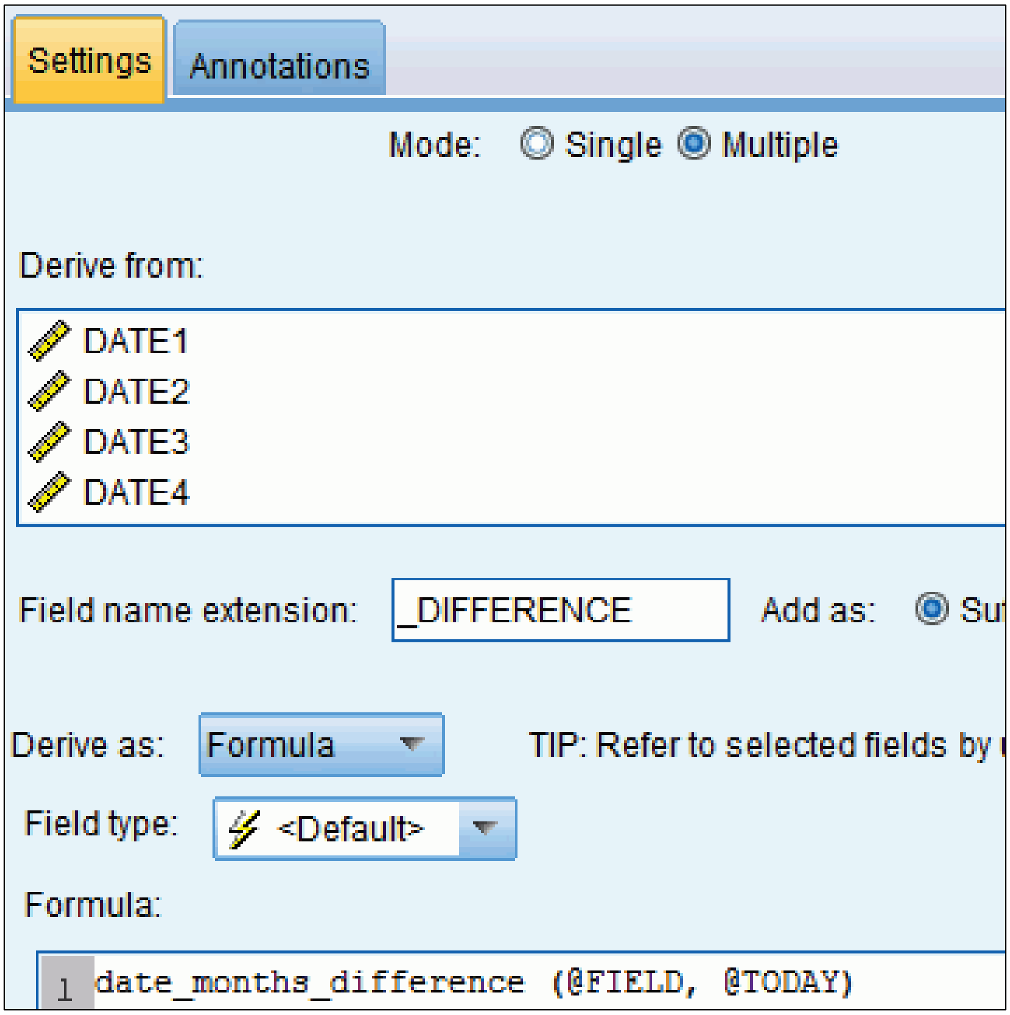
Question 2: Refer to the figure below. Which of the following statements are correct? (Select all that apply.)

A. Four fields are derived, each being the difference in months between DATE1 and today's date, DATE2 and today's date, DATE3 and today's date, and DATE4 and today's date.

B. The term @FIELD in the expression references all fields listed in the Derive from: area.

C. The new fields will be named DATE1_DIFFERENCE, DATE2_DIFFERENCE, DATE3_DIFFERENCE, and DATE4_DIFFERENCE.

D. The expression will issue an error message, because the arguments of the function @TODAY and @FIELD and must be in lower case.



Mode: Single [✓ Multiple]

Derive from:

DATE1
DATE2
DATE3
DATE4

Field name extension: _DIFFERENCE

Add as: [✓ Su

Derive as: Formula

TIP: Refer to selected fields by

Field type: <Default>

Formula:

```
date_months_difference (@FIELD, @TODAY)
```

Question 3: A dataset includes a field named E_MAIL_ADDRESS. Which of the following expressions derive a flag field which returns true if E_MAIL_ADDRESS contains at least one at sign @? (Select all that apply.)

A. locchar(`@`, 1, E_MAIL_ADDRESS) > 0

B. count_substring (E_MAIL_ADDRESS, "@") > 0

C. issubstring("@", E_MAIL_ADDRESS) > 0

D. E_MAIL_ADDRESS matches "*@*"

Note: The following functions are used:

locchar(CHAR,N, STRING). Returns the subscript at which CHAR is found in STRING. Searches from Nth character of STRING forward.

count_substring (STRING, SUBSTRING). Returns the number of instances of SUBSTRING that can be found in STRING.

issubstring (SUBSTRING, STRING). Searches STRING for SUBSTRING. If found, returns the starting subscript.

STRING1 matches STRING2. Returns true if STRING1 matches the pattern defined by STRING2. "?" matches 1 character, "*" matches 0 or more characters.

Question 4: A field named BIRTHDATE has storage string, and is formatted as MMYYYYDD (for example 01201104 represents January 4th, 2011). Which expression converts this string field to a date field?

A. datetime_date (substring (to_integer (BIRTHDATE)))

B. datetime_date (substring (5, 4, BIRTHDATE), substring (1, 2, BIRTHDATE), substring (7, 2, BIRTHDATE))

C. datetime_date(to_integer (substring (3, 4, BIRTHDATE)), to_integer (substring (1, 2, BIRTHDATE)), to_integer (substring (7, 2, BIRTHDATE)))

Note: The following functions are used:

datetime_date (YEAR, MONTH, DAY). Returns the date value for the given YEAR, MONTH, and DAY. The arguments must be integers.

substring (N, LENGTH, STRING). Returns a string, which consists of LEN characters of STRING, starting from the character at N.

to_integer (ITEM). Converts ITEM to an integer. ITEM must be a string, or a number.

Question 5: True or false: The functions @BLANK and @NULL will always return the same outcome.

A. True

B. False

Question 6: True or false: The mean_n function will return 0 when it is applied to a list of fields X1 to X4, when all these fields are undefined.

A. True

B. False

Question 7: Given is an integer field X and a real field Y. What is the storage of the field Z, which is derived as X >< Y?

A. Integer

B. Real

C. String

D. Date

Answers to questions

Answer 1: C. The expression will yield undefined ($null$) values. The datetime_date function needs arguments year, month, and day, in that order. Thus, the correct specification is datetime_date (2014, 1, 1)) rather than datetime_date (1, 1, 2014)).

Answer 2: A, B, C. The dialog box shows a multiple derive where four fields are derived. The formula computes the difference in months between today and the respective source fields, which are referenced by @FIELD. The new field name will be the concatenation of the original field name and _DIFFERENCE, as suffix. The functions @TODAY and @FIELD must be in upper case.

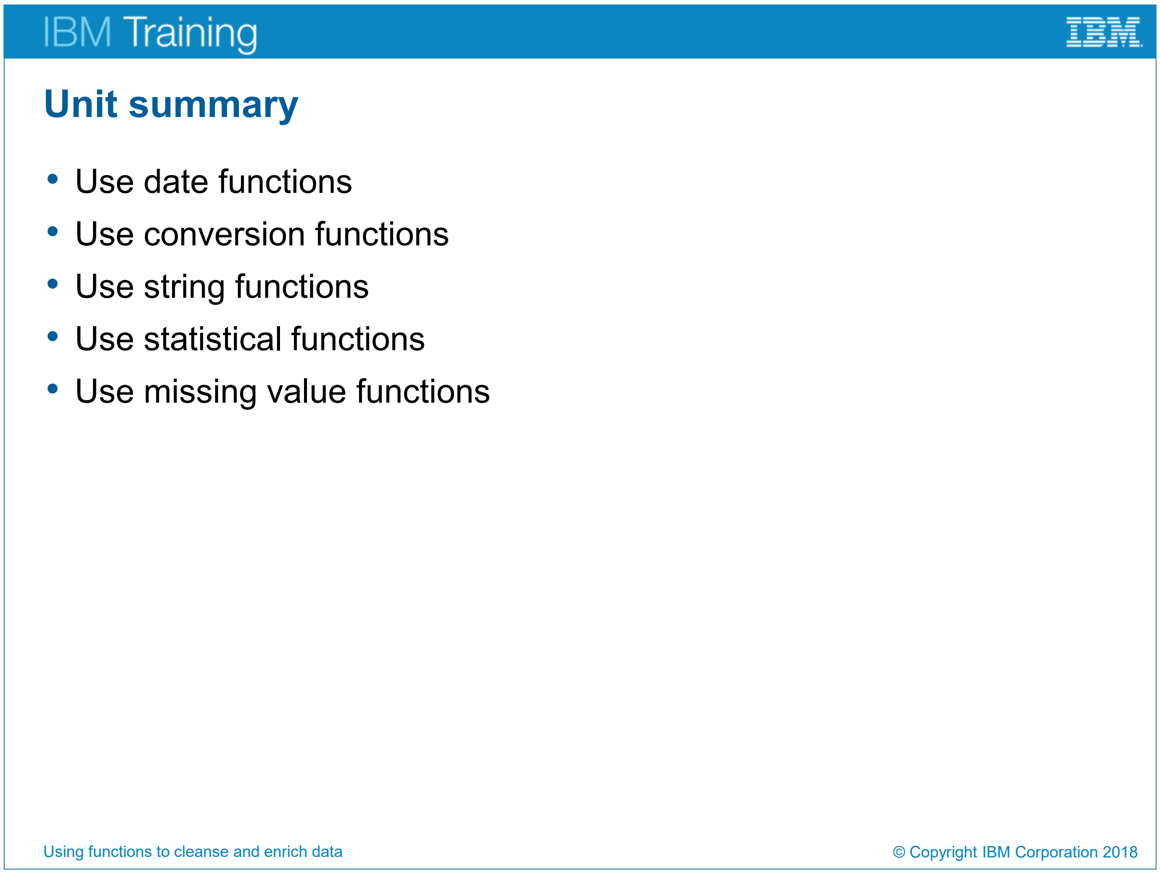
Answer 3: A, B, C, D. All specifications are correct. Notice that the locchar function requires back quotes, whereas the other functions require double quotes.

Answer 4: C. Year, month and day have to be extracted as integers. For example, to_integer(substring (1, 2, BIRTHDATE) extracts the month, as an integer. The datetime_date function is then used to convert the separate parts to a date format.

Answer 5: B. False. When undefined values are not declared as blanks, @BLANK will return false for an undefined ($null$) value, whereas @NULL will return true.

Answer 6: B. False. The mean_n function will return the undefined value when all the source fields are undefined.

Answer 7: C. The >< function will always return a string field, regardless of the storage of the source fields.



IBM Training

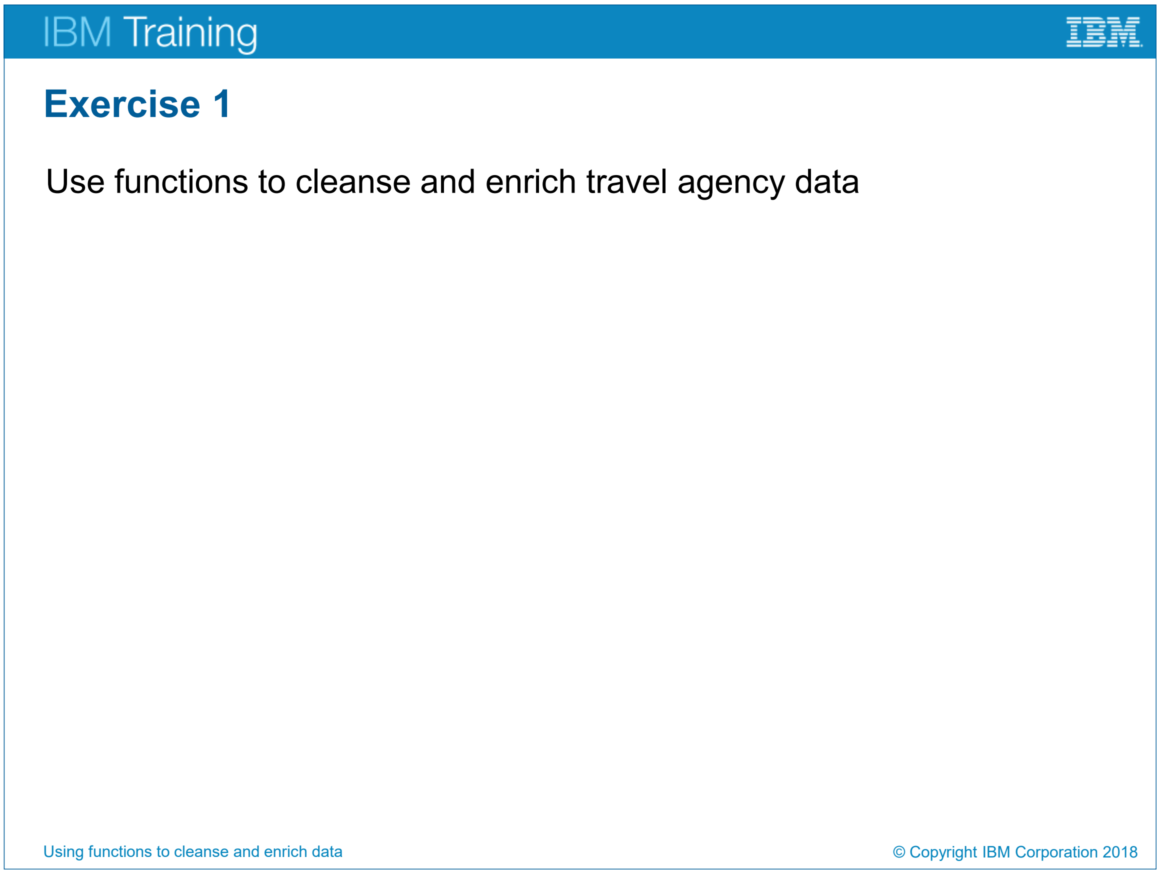
Unit summary

- Use date functions
- Use conversion functions
- Use string functions
- Use statistical functions
- Use missing value functions

Using functions to cleanse and enrich data

© Copyright IBM Corporation 2018

Unit summary



IBM Training

Exercise 1

Use functions to cleanse and enrich travel agency data

Using functions to cleanse and enrich data

© Copyright IBM Corporation 2018

Exercise 1: Use functions to cleanse and enrich travel agency data

Exercise 1: Use functions to cleanse and enrich travel agency data

Data file:	**customers_and_holidays.csv**
Data folder:	**C:\Training\0A058**

In this exercise you will work with data about customers and their holiday destinations. You will derive new fields to answer questions such as "What is the mean age of the customers?", "What was the most popular month to travel?", "What was the most popular destination?", and "What was the mean amount of money spent?"

- Use a **Var. File** node (**Sources** palette) to import the data from **customers_and_holidays.csv** (a comma separated text file), located in the **C:\Training\0A058** folder.

  Add a **Type** node (**Field Ops** palette) downstream from the **Var. File** node, edit the **Type** node, click the cell in the **DISTANCE_TO_BEACH** row and **Missing** column, and then declare **-999** as blank value for **DISTANCE_TO_BEACH**.

  Click **Read Values** to instantiate the data.

- Compute a new field, named **AGE**, the customer’s age (in years, as a real number) on the date that he traveled.

  Note: DOB stores the date of birth, TRAVDATE the date that the customer travelled.

  Use a **Statistics** node (**Output** palette) to examine the mean **AGE**.

- Create two fields: **YEAR_DOB** and **YEAR_TRAVDATE**, the year of birth and the year of travel, respectively. Use only a single Derive node to derive the two fields.

- Create **COUNTRY OK**, which equals **COUNTRY** but starts with an upper case letter, followed by lower case letters (for example: "SPAIN" should become "Spain").

  Check your results with a **Matrix** node (**Output** palette), cross tabulating **COUNTRY** by **COUNTRY OK**.

  Use a **Distribution** node (**Graphs** palette) to identify the most popular destination.

- Create a new field **CODE HOLIDAY** which returns the integer from the **HOLCODE** field. For example, the result for HOLCODE "CAF3105" should be CODE HOLIDAY 3105 (as an integer).

  Note: You may assume that HOLCODE is comprised of three letters, followed by the code.

  Check your results with a **Table** node (**Output** palette).
- Compute a new field named **SPENT TOTAL**, which is the total amount spent on **SPENT_TRAVEL**, **SPENT_HOTEL** and **SPENT_LEISURE**. However, the total should only be computed when there are at least two non-null values on these three fields.

  Use a **Statistics** node (**Output** palette) to see what the highest total amount spent is.
- DISTANCE_TO_BEACH stores the distance to the beach in miles. Create a new field named **DISTANCE KM** that returns the distance to the beach in kilometers (1 mile = 1.61 kilometers).

  Note: When the distance to the beach is unknown, the value for DISTANCE_TO_BEACH equals -999 (which was declared as blank in the first task).

  Use a **Statistics** node (**Output** palette) to see what the mean distance in kilometers is.
- Exit IBM SPSS Modeler without saving anything.

For more information about where to work and the exercise results, refer to the Task and Results section that follows. If you need more information to complete a task, refer to earlier demonstration for detailed steps.

Exercise 1: Tasks and Results

Task 1. Import data, define blanks, and instantiate the data.

- From the **Sources** palette, add a **Var. File** node to the stream canvas.
- Edit the **Var. File** node, and import the data from **customers_and_holidays.csv**, located in the **C:\Training\0A058** folder.
- From the **Field Ops** palette, add a **Type** node downstream from the **Var. File** node.
- Edit the **Type** node:
  - On the **DISTANCE_TO_BEACH** row, click the cell in the **Missing** column, and then click **Specify**.
  - Enable the **Define blanks** option, and then declare **-999** as blank by typing it into the **Missing values** text box.
  - Return to the main dialog box.
- Click **Read Values** to instantiate the data.

Task 2. Compute AGE.

- From the **Field Ops** palette, add a **Derive** node downstream from the **Type** node.
- Edit the **Derive** node:
  - Under **Derive field**, enter **AGE**.
  - Under **Formula**, enter **date_years_difference(DOB, TRAVDATE)**.
- From the **Output** palette, add a **Statistics** node downstream from the **Derive** node.
- Edit the **Statistics** node:
  - Beside **Examine**, select **AGE**.
  - Run the **Statistics** node.

  Mean age is 41.665.

Task 3. Create fields that hold the year for date fields.

- From the **Field Ops** palette, add a **Derive** node downstream from the **Derive** node named **AGE**.
- Edit the **Derive** node:
  - Beside **Mode**, select **Multiple**.
  - **Derive from**: Select **DOB** and **TRAVDATE**.
  - **Field name extension**: Type **YEAR_**.
  - **Add as**: Select **Prefix**.
  - **Formula**: Enter **datetime_year(@FIELD)**.
- From the **Output** palette, run a **Table** node downstream from the **Derive** node to check your results. Rearrange the fields to better examine the results, if preferred.

  YEAR_DOB stores the year one was born, YEAR_TRAVDATE the year of travel, as required.

Task 4. Cleanse the values for COUNTRY.

- From the **Field Ops** palette, add a **Derive** node downstream from the **Derive** node named **YEAR_**.
- Edit the **Derive** node:
  - Under **Derive field**, enter **COUNTRY OK**.
  - **Formula**: Enter **lowertoupper (startstring (1, COUNTRY)) >< uppertolower (allbutfirst (1, COUNTRY))**.
- From the **Output** palette, add a **Matrix** node downstream from the **Derive** node named **COUNTRY OK**.
- Edit the **Matrix** node:
  - Beside **Rows** select **COUNTRY**.
  - Beside **Columns** select **COUNTRY OK**.
  - Run the **Matrix** node.

  The country names are as required.
- From the **Graphs** palette, add a **Distribution** node downstream from the **Derive** node.

- Edit the **Distribution** node:
  - Beside **Field**, select **COUNTRY OK**.
  - Run the **Distribution** node.

  Spain appears to be the most popular holiday destination.

Task 5. Extract a number out of the string HOLCODE.

- From the **Field Ops** palette, add a **Derive** node downstream from the **Derive** node named **COUNTRY OK**.
- Edit the **Derive** node:
  - Under **Derive field**, enter **CODE HOLIDAY**.
  - Under **Formula,** enter **to_integer (allbutfirst (3, HOLCODE))**.
- From the **Output** palette, add a **Table** node downstream from the **Derive** node named **CODE HOLIDAY**, and then run the **Table** node.

  The results are as required.

  Note: You cannot use a Matrix node to check the results, because CODE HOLIDAY is typed as a continuous field. Alternatively, you could have derived CODE HOLIDAY as a string field by omitting the to_integer function.

Task 6. Conditionally compute the sum across fields.

- From the **Field Ops** palette, add a **Derive** node downstream from the **Derive** node named **CODE HOLIDAY**.
- Edit the **Derive** node:
  - Under **Derive field**, enter **SPENT TOTAL**.
  - Under **Derive as**, select **Conditional**.
  - Under **If**, enter **count_non_nulls ([SPENT_TRAVEL SPENT_HOTEL SPENT_LEISURE])>= 2**.
  - Under **Then**, enter **sum_n ([SPENT_TRAVEL SPENT_HOTEL SPENT_LEISURE])**.
  - Under **Else**, enter **undef**.
- From the **Output** palette, add a **Statistics** node downstream from the **Derive** node.

- Edit the **Statistics** node:
    - Select **SPENT TOTAL**.
    - Run the **Statistics** node.

The highest amount spent is 4109.

Task 7. Derive a field taking blank values into account.

- From the **Field Ops** palette, add a **Derive** node downstream from the **Derive** node named **SPENT TOTAL**.
- Edit the **Derive** node:
    - Under **Derive field**, enter **DISTANCE KM**.
    - Under **Derive a**s, select **Conditional**.
    - Under **If**, enter **not (@BLANK (DISTANCE_TO_BEACH))**.
    - Under **Then**, enter **DISTANCE_TO_BEACH * 1.61**.
    - Under **Else**, enter **undef**.
- From the **Output** palette, add a **Statistics** node downstream from the **Derive** node.
- Edit the **Statistics** node:
    - Select **DISTANCE KM**.
    - Run the **Statistics** node.

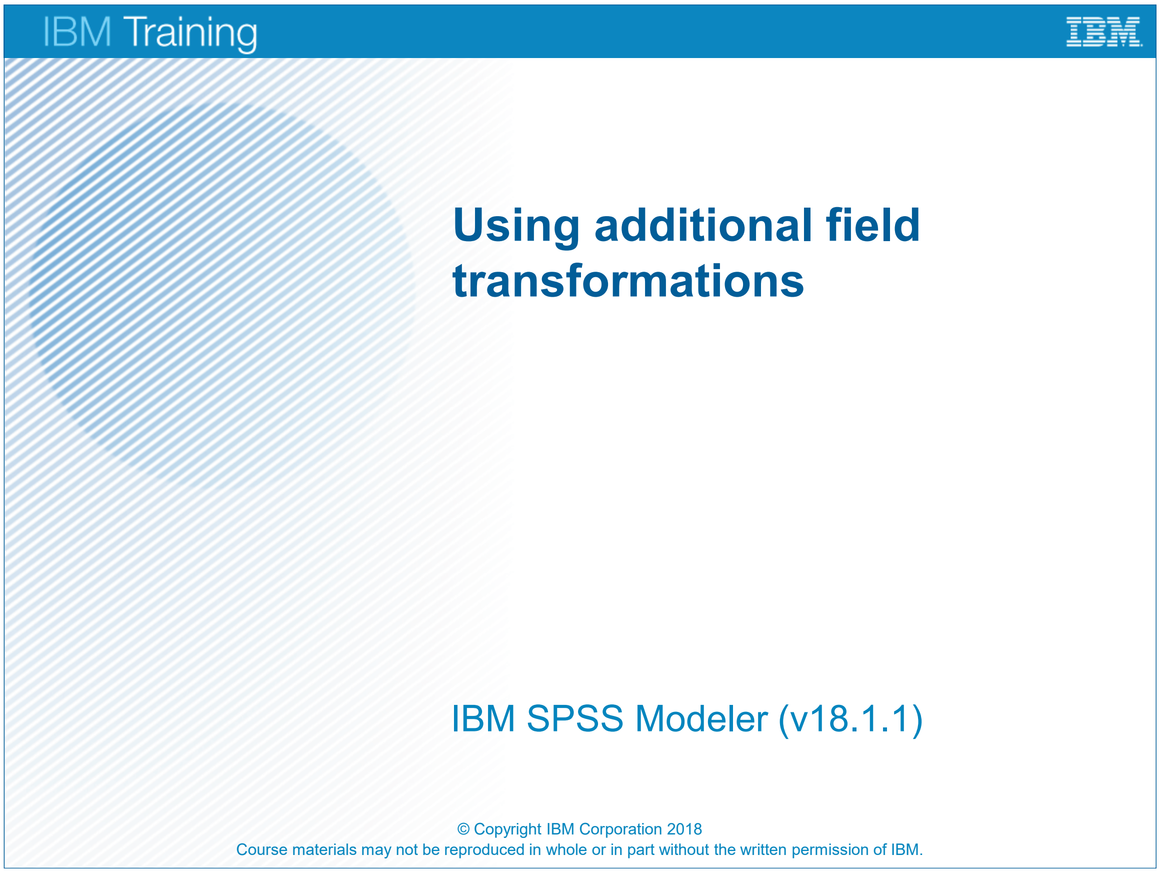
  Mean distance is 6.317 KM. Notice that there are 407 records in computing the statistics, whereas the total number of records in the dataset is 411. Four records were assigned the undefined ($null$) value because they had -999, which was declared as blank value on the source field DISTANCE_TO_BEACH.

Task 8. Exit IBM SPSS Modeler

- From the **File** menu, click **Exit**, and exit **IBM SPSS Modeler** without saving anything.

You will find the solution results in the **C:\Training\0A058\Solutions** folder.

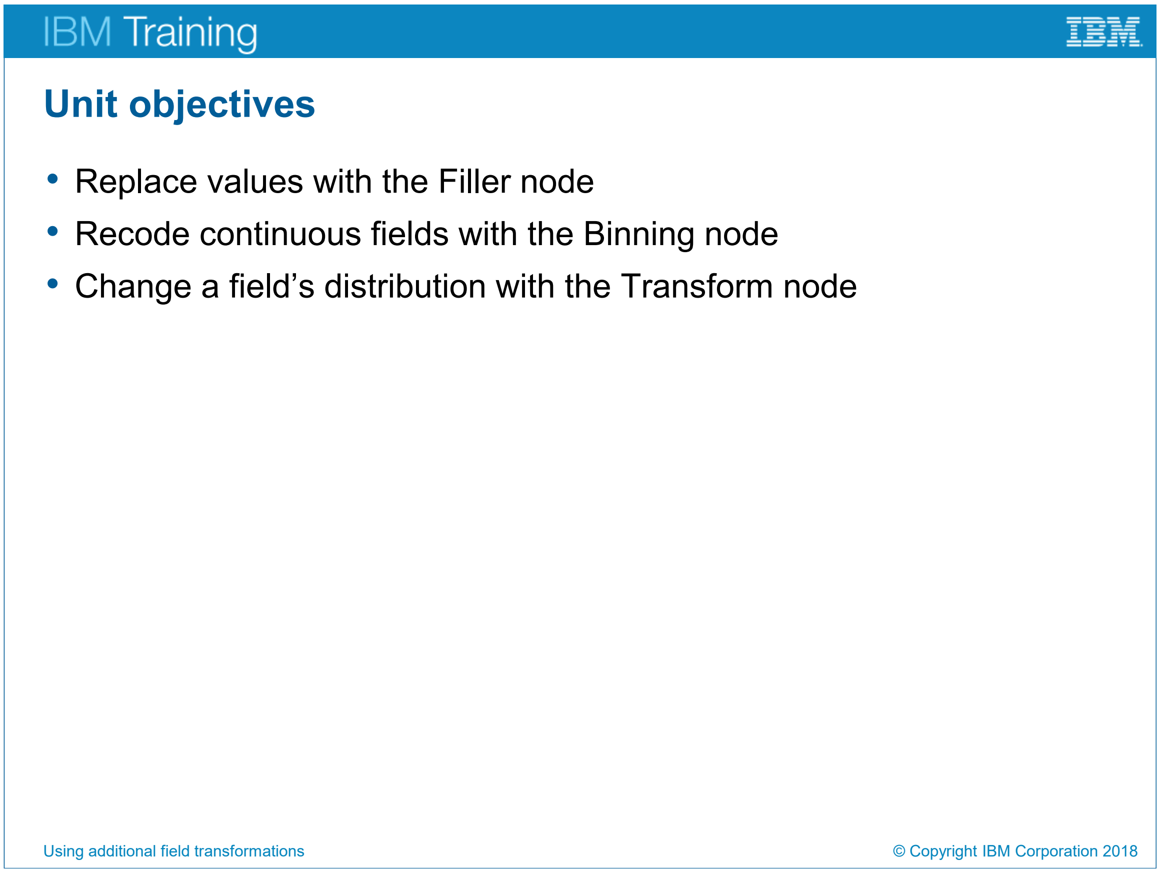
Unit 2 Using additional field transformations



IBM Training

Using additional field transformations

IBM SPSS Modeler (v18.1.1)


IBM Training

Unit objectives

- Replace values with the Filler node
- Recode continuous fields with the Binning node
- Change a field’s distribution with the Transform node

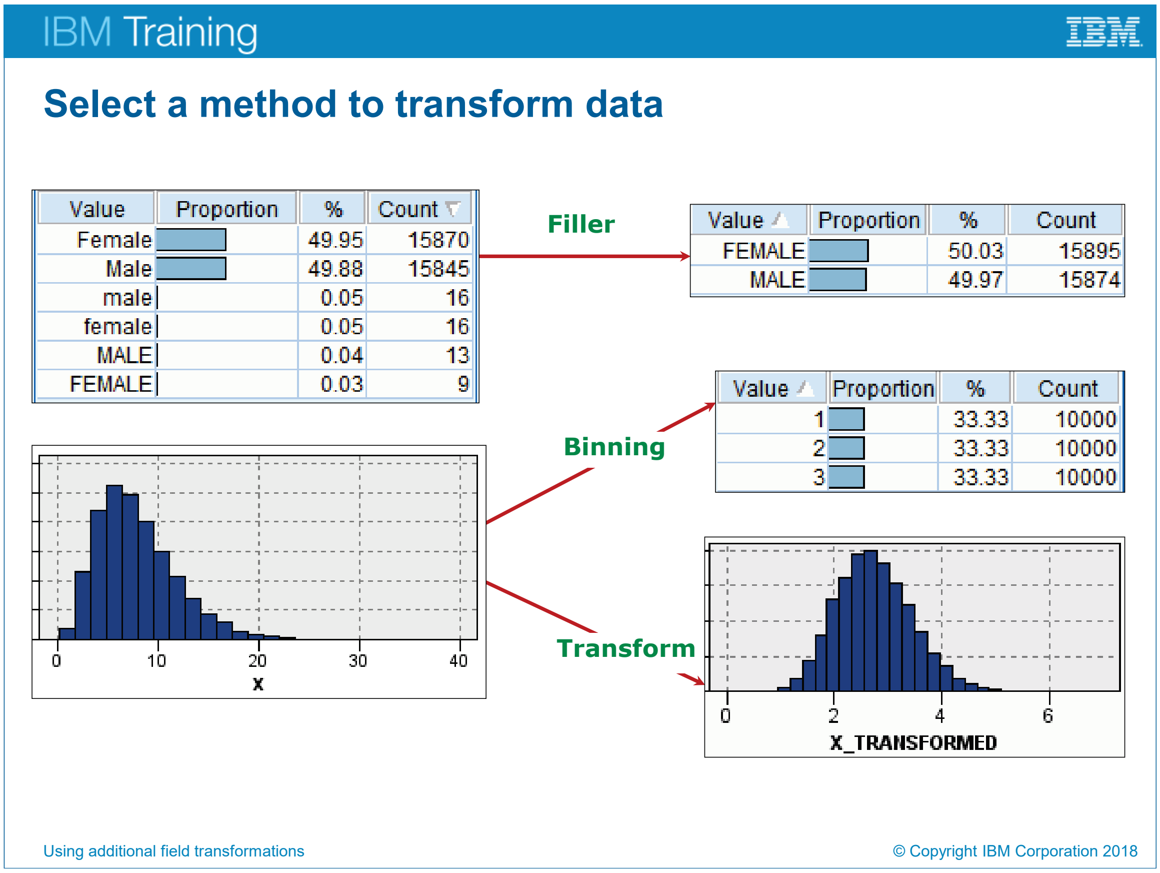
Using additional field transformations

© Copyright IBM Corporation 2018

Unit objectives

Before reviewing this unit you should be familiar with:

- working with IBM SPSS Modeler (streams, nodes, SuperNodes, palettes)
- importing data (Var. File node)
- defining measurement levels, roles, blanks, and instantiating data (Type node)
- selecting fields and records (Filter node, Select node)
- CLEM and the Expression Builder
- deriving fields (Derive node)
- date and time-, conversion-, string-, statistical- and missing value functions
- examining the data with graphs (Distribution node, Histogram node) and tables (Table node, Data Audit node, Statistics node, Matrix node, Means node)


IBM Training
IBM
Filler
Binning
Transform
0
10
20
30
40
X
0
2
4
6
X_TRANSFORMED
Using additional field transformations
© Copyright IBM Corporation 2018

Select a method to transform data

Value	Proportion	%	Count
Female	🟦	49.95	15870
Male	🟦	49.88	15845
male		0.05	16
female		0.05	16
MALE		0.04	13
FEMALE		0.03	9

Filler

Value	Proportion	%	Count
FEMALE	🟦	50.03	15895
MALE	🟦	49.97	15874

Binning

Value	Proportion	%	Count
1	🟦	33.33	10000
2	🟦	33.33	10000
3	🟦	33.33	10000

Transform

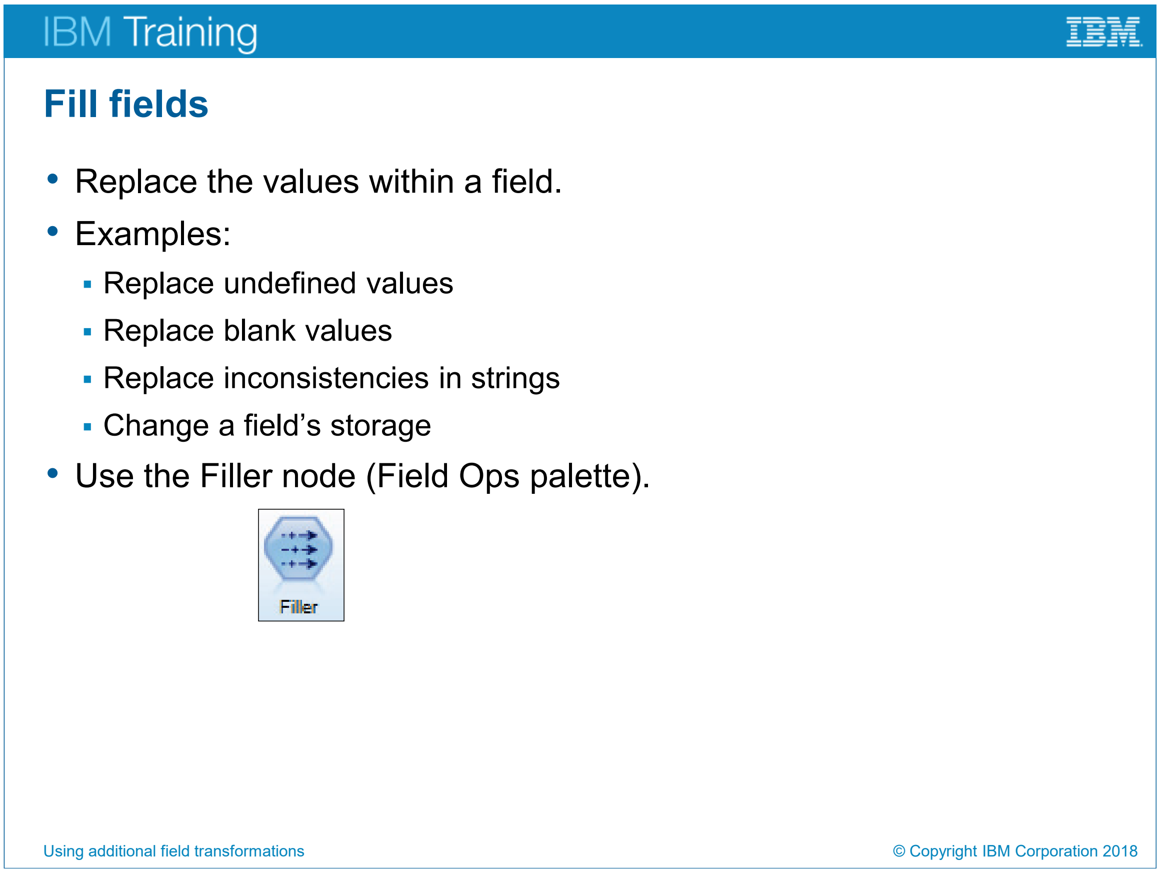
Select a method to transform data

Preparing data for modeling often involves cleaning existing fields, or deriving new ones. Two field operations were presented in the *Introduction to IBM SPSS Modeler and Data Science* course, Derive and Reclassify. In this unit additional nodes are introduced to modify fields, giving you more options to prepare the data for modeling.

In the first place, whereas Derive always creates a new field, it is more efficient to replace the values in the field itself. The Filler node offers this capability.

Secondly, although both Derive and Reclassify can be used to recode a field into a number of categories, there is often the need to do this more efficiently. In particular, a data driven recode is required where the thresholds are based on the data itself. The Binning node provides this functionality.

Thirdly, for modeling purposes, it may be needed to transform a field so that its distribution conforms to the assumptions of the modeling technique. A field's distribution can be explored and, if desired, modified with the Transform node.



IBM Training

Fill fields

- Replace the values within a field.
- Examples:
  - Replace undefined values
  - Replace blank values
  - Replace inconsistencies in strings
  - Change a field's storage
- Use the Filler node (Field Ops palette).

Using additional field transformations

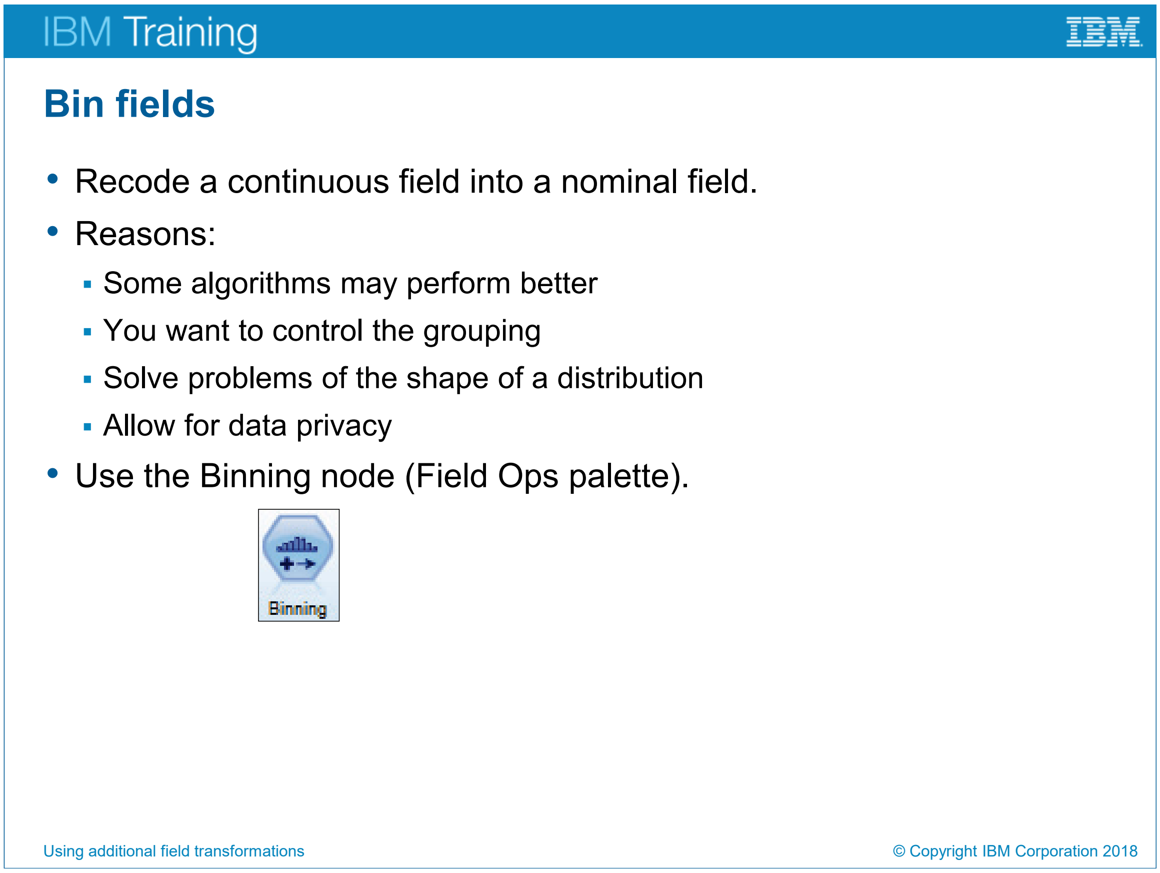
© Copyright IBM Corporation 2018

Fill fields

Sometimes there is a need to replace a field's values. Applications include:

- Replace the undefined ($null$) value by a valid value. For example, set the undefined ($null$) value for a flag field to F.
- Replace user-defined blanks by the $null$ value. For example, to prevent that user-defined blanks are included in the calculation of aggregate statistics you can replace the user-defined blanks by the undefined ($null$) value.
- Remove inconsistencies in string values by replacing them by lower case or upper case letters. For example, if gender stores values FEMALE, female, Female, MALE, Male, male, replace the values by FEMALE and MALE.
- Change the storage of a field. For example, when a field age has values 21.00, 22.00, and so forth, convert the field to integer values 21, 22, and so forth.

The Filler node, located in the Field Ops palette, is used to replace field values and to change storage. The Filler node uses the CLEM language for the specifications.



IBM Training

IBM

Bin fields

- Recode a continuous field into a nominal field.
- Reasons:
  - Some algorithms may perform better
  - You want to control the grouping
  - Solve problems of the shape of a distribution
  - Allow for data privacy
- Use the Binning node (Field Ops palette).

Using additional field transformations

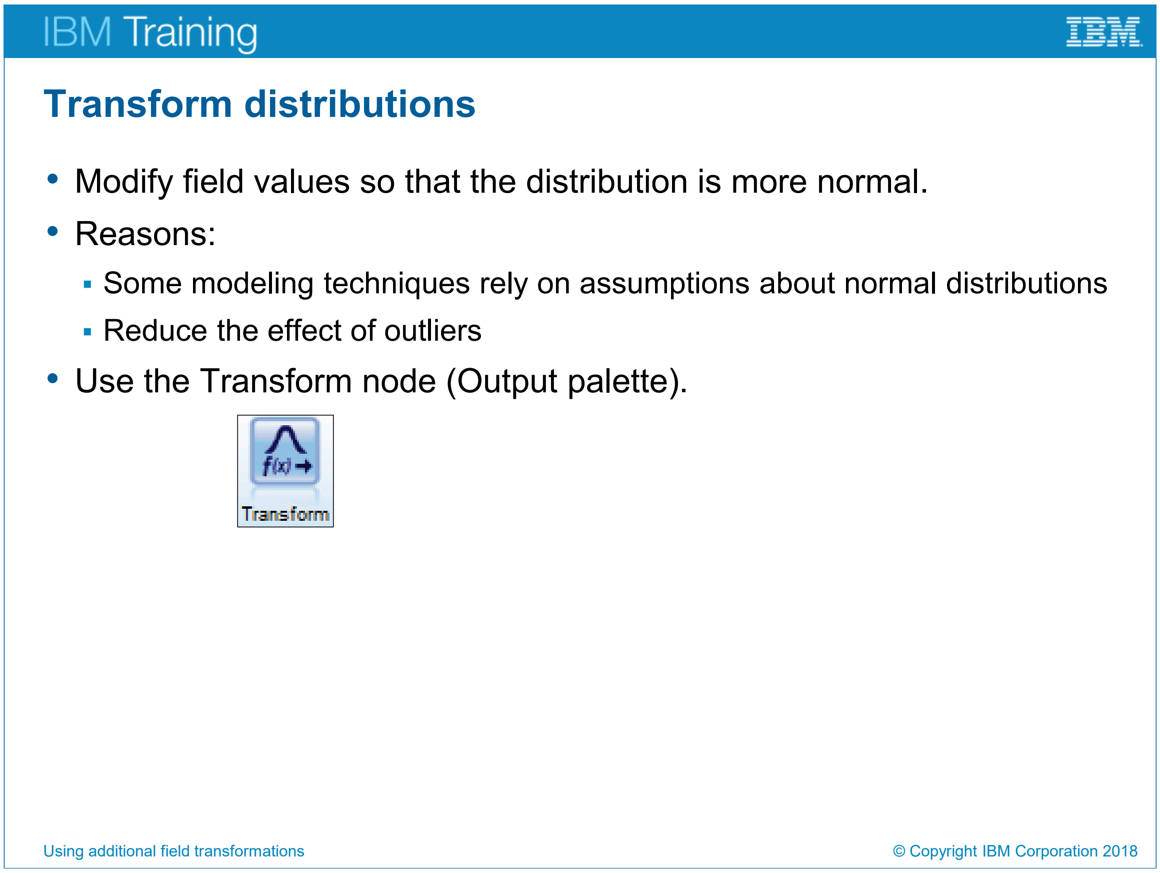
© Copyright IBM Corporation 2018

Bin fields

Binning will recode a continuous field into a nominal field. There are several reasons why you might want to bin a continuous field:Some algorithms such as decision trees may perform better if a predictor has fewer categories.

- Even when an algorithm groups values of a continuous field, you may wish to control the grouping beforehand to create meaningful categories rather than rely on the grouping method of the algorithm.
- The effect of outliers can be reduced by binning, so that all the outliers and extreme cases are placed in one bin.
- Binning solves problems of the shape of a distribution since the continuous field is turned into a categorical field.
- Binning can allow for data privacy by reporting such things as salaries or bonuses in ranges rather than the actual values.

The Binning node is located in the Field Ops palette.



IBM Training

Transform distributions

- Modify field values so that the distribution is more normal.
- Reasons:
  - Some modeling techniques rely on assumptions about normal distributions
  - Reduce the effect of outliers
- Use the Transform node (Output palette).

Transform

Using additional field transformations

© Copyright IBM Corporation 2018

Transform distributions

When the distribution of a continuous field is skewed rather than normal (bell shaped), it can cause problems when you build models:

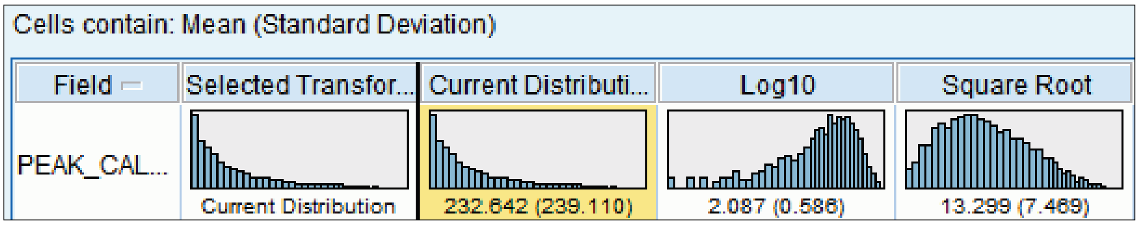
Several of the modeling techniques in IBM SPSS Modeler that are based on traditional statistical theory function best with normal data, including regression, logistic regression, and discriminant analysis. These techniques rely on assumptions about normal distributions of data that may not often be true for real world data.

Also, skewed fields will typically have outliers, because the skewness, in part, is caused by the outlying values. Reducing the effect of outliers may lead to better models.

One approach to handle skewed distributions is to apply a transformation that modifies a field's values so that the overall distribution conforms more to the normal distribution. The Transform node (located in the Output palette, not in the Field Ops palette) accomplishes this.

IBM Training

Explore Transform output



Cells contain: Mean (Standard Deviation)

Field	Selected Transfor...	Current Distributi...	Log10	Square Root
PEAK_CAL...	Current Distribution	232.642 (239.110)	2.087 (0.586)	13.299 (7.469)

Using additional field transformations

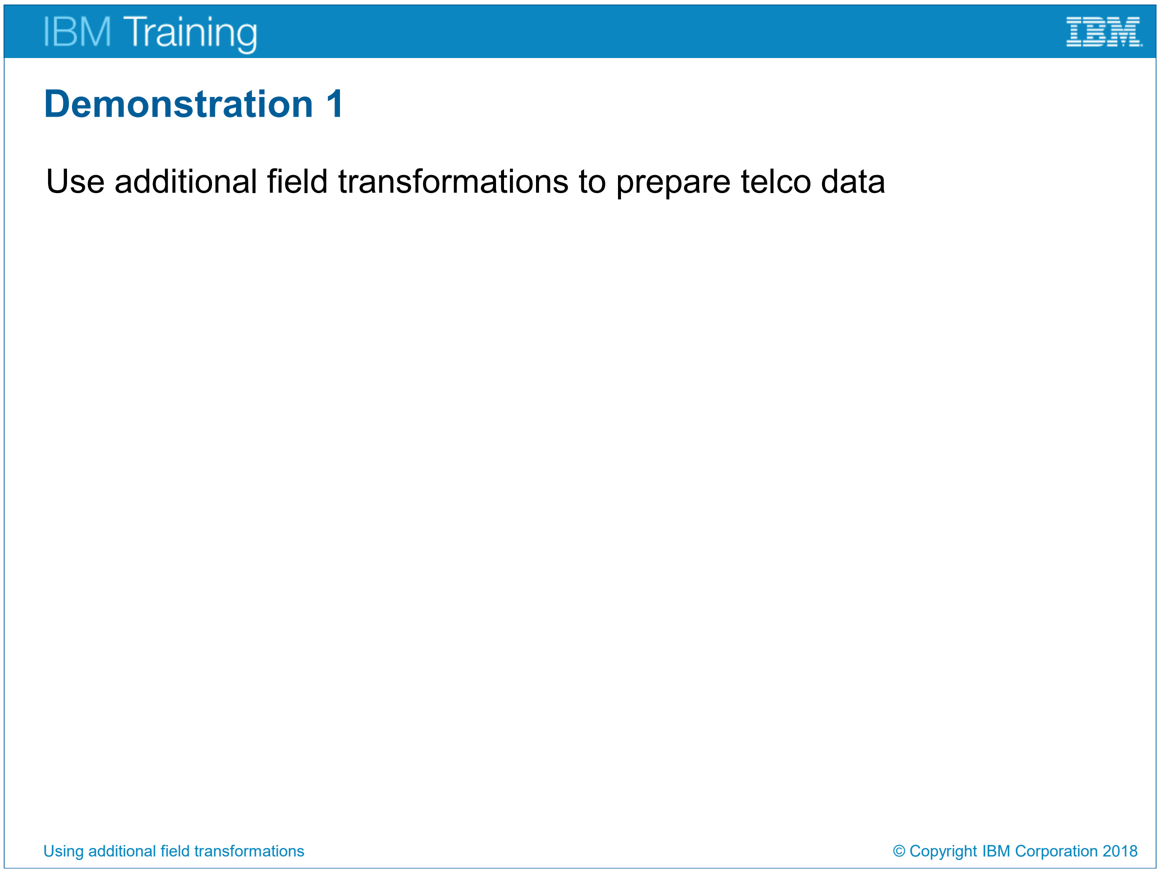
© Copyright IBM Corporation 2018

Explore Transform output

The Transform output window shows thumbnails with the histograms of the transformed fields. Double-clicking a thumbnail will show the full histogram, overlaid with the best fitting normal distribution.

If you want to create a field with the transformed scores you can select either Generate\Derive Node or Generate\Filler Node to generate a new field or to overwrite the field, respectively. In both cases a SuperNode will be generated which comprises the transformations.

As a further option, the transformed scores can be standardized. Standardized scores express how many standard deviations a score is above or below the mean. Refer to the *Improving efficiency* unit in this course for information about standardized scores.



IBM Training

Demonstration 1

Use additional field transformations to prepare telco data

Using additional field transformations

© Copyright IBM Corporation 2018

Demonstration 1: Use additional field transformations to prepare telco data

Demonstration 1: Use additional field transformations to prepare telco data

Purpose:
In order to build better models later you will cleanse data, bin fields, and transform fields so that their distribution is similar to the normal distribution. You will use the Filler node, Binning node and Transform node to accomplish this.

Data file:	**telco x churn data.csv**
Data folder:	**C:\Training\0A058**

Task 1. Start IBM SPSS Modeler and set the working folder.

1. From the **Start** menu, expand **IBM SPSS Modeler 18.1**, and then click **IBM SPSS Modeler 18.1**. When a splash window displays, click **Cancel**.

   If you have already configured IBM SPSS Modeler in a previous demonstration or exercise, you can skip to Task 2.
2. From the **File** menu, click **Set Directory**.
3. Beside **Look in**, navigate to **C:\Training\0A058**, and then click **Set**.

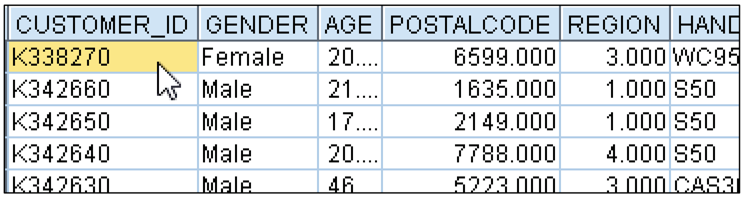
Task 2. Import and instantiate the data.

1. From the **Sources** palette, double-click the **Var. File** node to add it to the stream canvas.
2. Edit the **Var. file** node.
3. To the right of the **File** field, click **Browse** (the Browse window should automatically open to the **C:\Training\0A058** folder, per specifications in Task 1).
4. Select **telco x churn data.csv**, and then click **Open**.
5. Close the **Var. File** dialog box.
6. From the **Field Ops** palette, add a **Type** node downstream from the **Var. File** node.
7. Edit the **Type** node.
8. Click **Read Values** to instantiate the data.

   Notice the different spellings for GENDER.

9. Click **Preview**.

   The results appear similar to the following:



CUSTOMER_ID	GENDER	AGE	POSTALCODE	REGION	HAND
K338270	Female	20....	6599.000	3.000	WC95
K342660	Male	21....	1635.000	1.000	S50
K342650	Male	17....	2149.000	1.000	S50
K342640	Male	20....	7788.000	4.000	S50
K342630	Male	46	5223.000	3.000	CAS3

   AGE, POSTALCODE, and REGION, amongst others, are stored as real values. You will change their storage to integer.

10. Close the **Preview** output window.
11. Close the **Type** dialog box.

Task 3. Use the Filler node to change storage.

You will use the Filler node to change the storage of a number of fields.

1. From the **Field Ops** palette, add a **Filler** node downstream from the **Type** node.

   Note: Be careful to select the Filler node, not the Filter node.

2. Edit the **Filler** node.
3. Under **Fill in fields**, select **AGE**, **POSTALCODE**, **REGION**, and **DROPPED_CALLS**.

   Values will be replaced, depending on the condition that you specify.

4. Beside **Replace**, click **Based on condition**, so that you can view the options.

   You can specify a condition, choose that values must always be replaced, or replace values based on their missing values status (undefined, user-defined blank, or both).

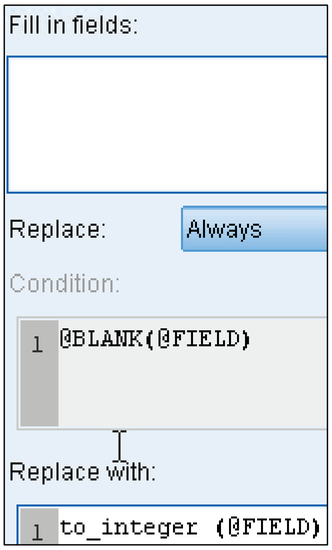
   In this case, real values must always be replaced by integer values.

5. Beside **Replace**, click **Always** from the list.

   In the Replace with area, specify the expression for the new value. Use @FIELD to refer to the selected fields. If preferred, use the Expression builder to enter the expression.

6. Under **Replace with**, enter **to_integer (@FIELD)**. If you use the **Expression Builder**, locate the **to_integer** function in the **Conversion** function group.

   The results appear as follows:



Fill in fields:

Replace: Always

Condition:

```
@BLANK(@FIELD)
```

Replace with:

```
to_integer (@FIELD)
```

7. Click **Preview**.

   The values of the specified fields are integers, as required.
8. Close the **Preview** output window.
9. Close the **Filler** dialog box.

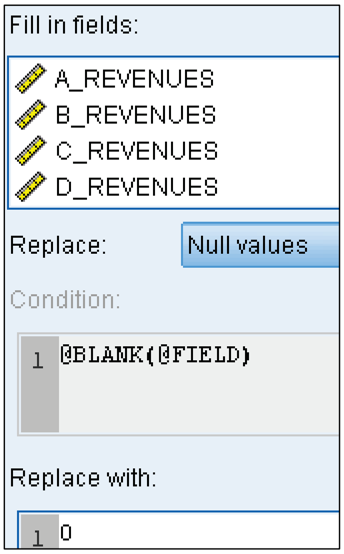
Task 4. Use the Filler node to replace null values.

A_REVENUES to L_REVENUES store the revenues for various products. These fields are undefined ($null$) when one did not purchase the product. These undefined values should be replaced by 0, because missing revenues for a product means that revenues for that product is actually zero.

1. From the **Field Ops** palette, add a **Filler** node downstream from to the **first Filler** node.
2. Edit the **Filler** node.
3. Under **Fill in fields**, select **A_REVENUES** to **L_ REVENUES**.
4. Beside **Replace**, click **Null values** from the list.

5. Under **Replace with**, ensure that the value is **0**.

   The results appear as follows:



Fill in fields:
A_REVENUES
B_REVENUES
C_REVENUES
D_REVENUES

Replace: Null values

Condition:

```
@BLANK(@FIELD)
```

Replace with:

```
0
```

6. Click **Preview**, and then scroll to **A_REVENUES** in the output window.

   $null$ values have been replaced by 0, as required.
7. Close the **Preview** output window.
8. Close the **Filler** dialog box.

Task 5. Use the Filler node to replace strings.

The values of GENDER are not spelled consistently. The different spellings must be replaced by upper case letters.

1. Add a **Filler** node downstream from the **second Filler** node.
2. Edit the **Filler** node.
3. Under **Fill in fields**, select **GENDER**.
4. Beside **Replace**, click **Always** from the list.
5. Under **Replace with**, enter **lowertoupper (@FIELD)** (alternatively, use lowertoupper (GENDER); using @FIELD rather than the field name is more general and will work when the transformation has to be applied to multiple fields).
6. Click **Preview**.

   The values are in upper case, as required.
7. Close the **Preview** output window.
8. Close the **Filler** dialog box.

Task 6. Bin with equal counts.

You will bin BILL_TOTAL into three categories, so that each category has the same numbers of records in it. The first category stores the 33.3% records with the smallest values for BILL_TOTAL, and the third category stores the 33.3% records with largest values for BILL_TOTAL. You could label the categories as bronze customers, silver customers, and gold customers.

1. From the **Field Ops** palette, add a **Binning** node downstream from the **third Filler** node.
2. Edit the **Binning** node.

   Select the field(s) to bin on the Settings tab.
3. Click the **Settings** tab, if necessary.
4. Beside **Bin fields**, select **BILL_TOTAL** (the last field in the list).
5. Beside **Binning method**, click **Fixed-width** to view the options.

   The Fixed-width binning option group the values of the original field into equal ranges, such as AGE in groups of 20-29, 30-39, 40-49 and so forth.

   The Tiles options creates groups based on percentiles. For example, the choice of quartiles will create four groups of equal numbers of records. Another option within this method is to create groups so that the sum of values in each group is approximately the same.

   The Ranks option transforms a field into ranks, from 1 to N, where N is the number of distinct values in the original field.

   The Mean/standard deviation option groups the values based on the number of standard deviations below and above the mean.

   The Optimal option bins a field in such a way that it has the strongest relationship with a specified categorical field, referred to as supervising field.

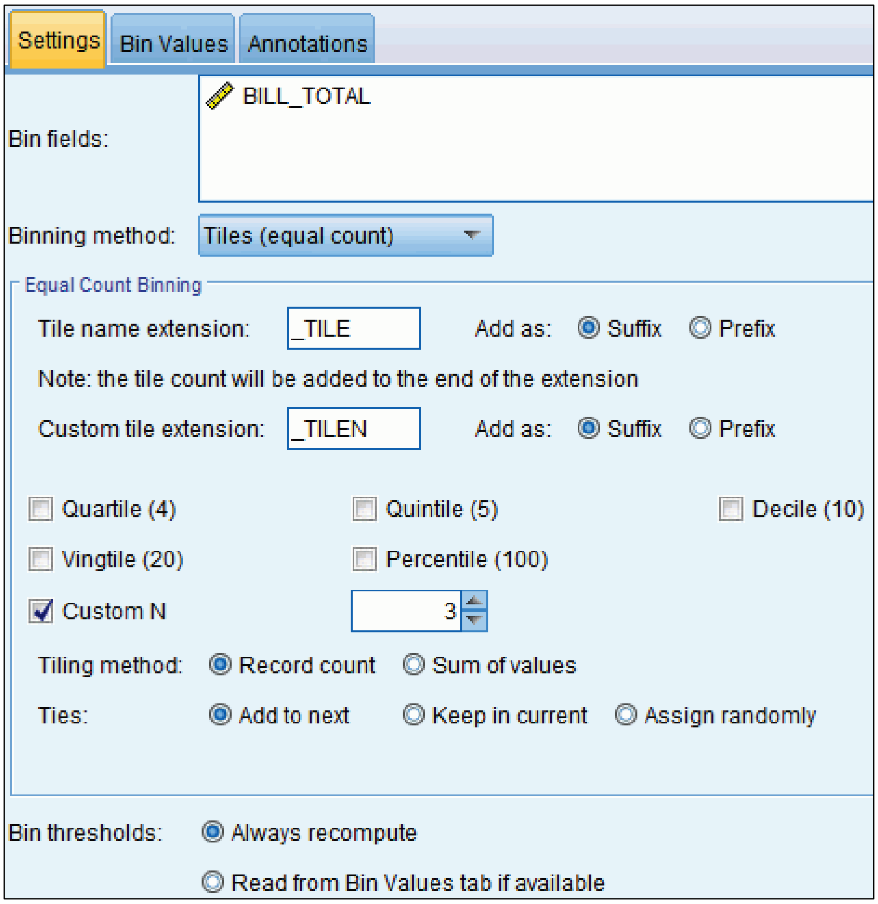
   Which binning method you choose is a business decision. You can use more than one method by adding multiple Binning nodes to the stream, run the model(s) with the binned fields, and determine the most important field(s).

   In this example, three categories with the same number of records in it will be created.
6. In the **Binning method** drop-down list, select **Tiles (equal count)**.

   The lower part of the dialog box will reflect the selected method.
7. Disable the **Decile (10)** option.

8. Enable the **Custom N** option, and then replace the current value by **3**.

   The results appear as follows:



Settings | Bin Values | Annotations

Bin fields: BILL_TOTAL

Binning method: Tiles (equal count)

Equal Count Binning

Tile name extension: _TILE Add as: [✓ Suffix] Prefix

Note: the tile count will be added to the end of the extension

Custom tile extension: _TILEN Add as: [✓ Suffix] Prefix

☐ Quartile (4) ☐ Quintile (5) ☐ Decile (10)

☐ Vingtile (20) ☐ Percentile (100)

☑ Custom N 3

Tiling method: [✓ Record count] Sum of values

Ties: [✓ Add to next] Keep in current Assign randomly

Bin thresholds: [✓ Always recompute]

Read from Bin Values tab if available

   The name for the new field will be BILL_TOTAL_TILEN, the source field name extended with the custom tile extension, _TILEN.

   There are two options for the thresholds that are used for binning. You can use the thresholds values that are already computed in a previous run, or you can re-compute the threshold values when data pass through the Binning node. The Bin Values tab shows these thresholds and enables you to re-compute the thresholds by enforcing a data pass.

   You will examine the thresholds that will be used for binning the field.

9. Click the **Bin Values** tab, and then click the **Read Values** button.

   The first threshold is 115.24. If a record has a BILL_TOTAL less than 115.240602, it will be assigned to the first category of the new field. Likewise, a customer with BILL_TOTAL greater than or equal to 178.166885 will be assigned to the third category.

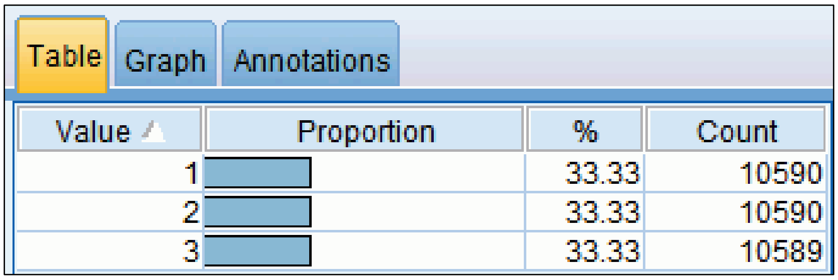
10. Close the **Binning** dialog box.

11. From the **Graphs** palette, add a **Distribution** node downstream from the **Binning** node.

12. Edit the **Distribution** node.

13. Beside **Field**, select **BILL_TOTAL_TILEN**.
14. Click **Run**.

    The results appear as follows:



Value	Proportion	%	Count
1	▬	33.33	10590
2	▬	33.33	10590
3	▬	33.33	10589

    A nominal field has been created with three categories, containing, approximately, the same number of records.

15. Close the **Distribution** output window.

Task 7. Bin a field optimally with respect to a target.

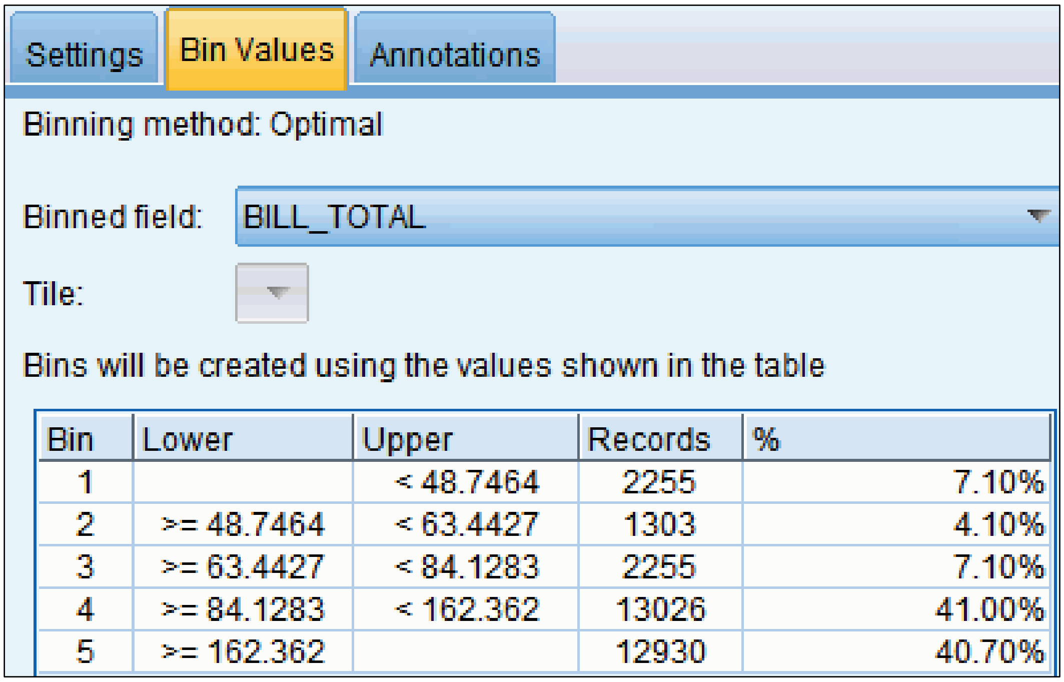
You will bin BILL_TOTAL in such a way that its relationship with CHURN (flagging whether the customer has cancelled the subscription) is maximized.

1. From the **Field Ops** palette, add a **Binning** node downstream from the **first Binning** node.
2. Edit the **second Binning** node.
3. Next to **Bin fields**, select **BILL_TOTAL**.
4. Beside **Binning method**, click **Optimal** from the list.
5. Beside **Supervisor field**, select **CHURN**.

    Notice that the new field name is the source field name, extended with _OPTIMAL.

    You will examine the thresholds used for binning.

6. Click the **Bin Values** tab, and then click **Read Values**.
   The results appear as follows:



Binning method: Optimal

Binned field: BILL_TOTAL

Tile:

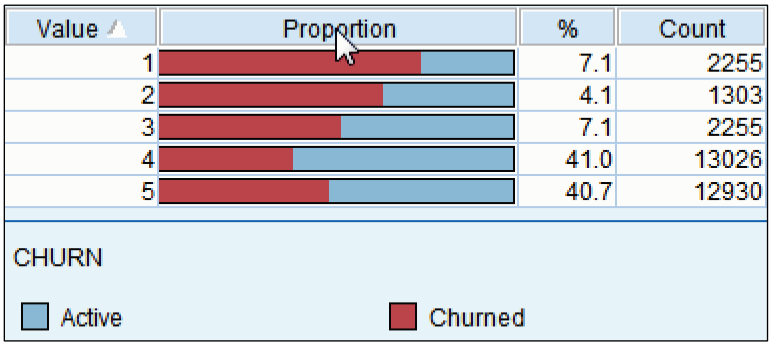
Bins will be created using the values shown in the table

Bin	Lower	Upper	Records	%
1		< 48.7464	2255	7.10%
2	>= 48.7464	< 63.4427	1303	4.10%
3	>= 63.4427	< 84.1283	2255	7.10%
4	>= 84.1283	< 162.362	13026	41.00%
5	>= 162.362		12930	40.70%

   Optimal binning will create a categorical field with 5 categories; the first threshold is 48.7464.
7. Close the **Binning** dialog box.
8. From the **Graphs** palette, add a **Distribution** node downstream from the **second Binning** node.
9. Edit the **Distribution** node.
10. Beside **Field**, select **BILL_TOTAL_OPTIMAL**.
11. Beside **Color**, select **CHURN**.
12. Enable the **Normalize by color** option.

13. Click **Run**

    The results appear similar to the following:



Value	Proportion	%	Count
1		7.1	2255
2		4.1	1303
3		7.1	2255
4		41.0	13026
5		40.7	12930

CHURN

🟦 Active 🟥 Churned

The first group has the lowest values for BILL_TOTAL values, and the highest churn rate. Category 5 also shows a high churn rate, which may be a point of concern to the company.

14. Close the **Distribution** output window.

Task 8. Transform a field so that its distribution is less skewed.

You will examine the distribution of BILL_TOTAL and apply a transformation so that its distribution is less skewed (conforms better to the normal distribution).

1. From the **Output** palette, add a **Transform** node downstream from the **second Binning** node.
2. Edit the **Transform** node.

   Fields are selected on the Fields tab.
3. Click the **Fields** tab, if necessary.
4. Beside **Fields**, select **BILL_TOTAL**.
5. Click the **Options** tab.

   By default, all transformations are applied.

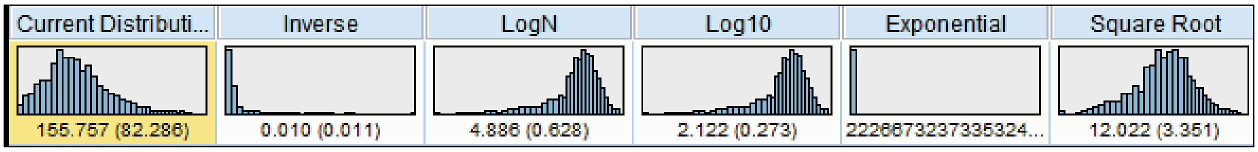
   The Inverse function is not defined for 0 and the two Log functions are not defined for values less than or equal to 0. When a field's value is 0, the Inverse function and the two Log functions return the undefined value ($null$). Therefore, an offset value can be specified for these three functions. The offset value will be added to the original scores. The offset value can be quite small, although when more than one field is specified at the same time, the offset will apply to all.

   Note: The log n transformation takes the natural log, which is the log function with base e (the number 2.718…). The log 10 transformation is the log function with base 10.

   You will keep the default to apply all transformations.

6. Click **Run**.

   The results appear as follows:

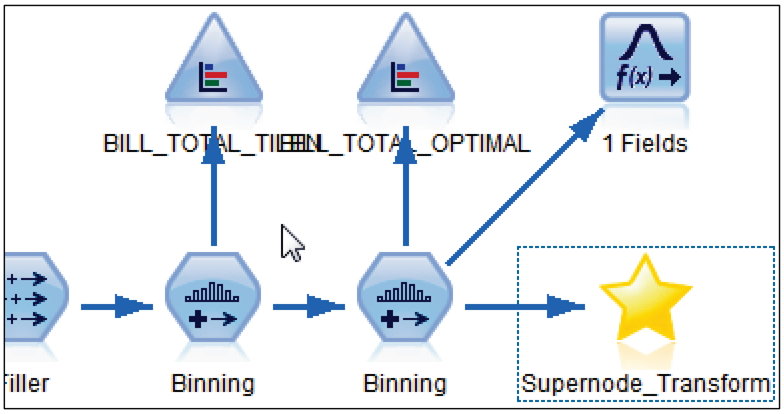


Current Distributi...	Inverse	LogN	Log10	Exponential	Square Root
155.757 (82.286)	0.010 (0.011)	4.886 (0.628)	2.122 (0.273)	2226673237335324...	12.022 (3.351)

   The square root transformation appears to be the best candidate to make the distribution less skewed. You will derive a new field that stores the transformed scores.

7. Click the **Square Root thumbnail** so that it is selected.
8. From the **Generate** menu, click **Derive Node**, and then click **OK** in the dialog box that displays (you will compute non-standardized scores).
9. Close the **Transform** output window.
10. Add the **Supernode_Transform** node, located in the upper left corner on the stream canvas, downstream from the **second Binning** node (make room by moving the Transform node, if preferred).

    The results appear similar to the following:



11. Edit the **Supernode_Transform** node, click **Zoom in**, and then edit the node named **BILL_TOTAL_Square Root**.

    A new field is derived, named BILL_TOTAL_Square Root, by applying the sqrt function to BILL_TOTAL.

12. Close the **Derive** dialog box, click the **Zoom out of SuperNode** button, and then close the **Supernode_Transform** dialog box.
13. From the **Output** palette, add a **Data Audit** node downstream from the **Supernode_Transform** node.
14. Edit the **Data Audit** node.
15. Click the **Settings** tab, and then click **Use custom fields**.
16. Beside **Fields**, select **BILL_TOTAL** and **BILL_TOTAL_Square Root.**

17. Click **Run**.

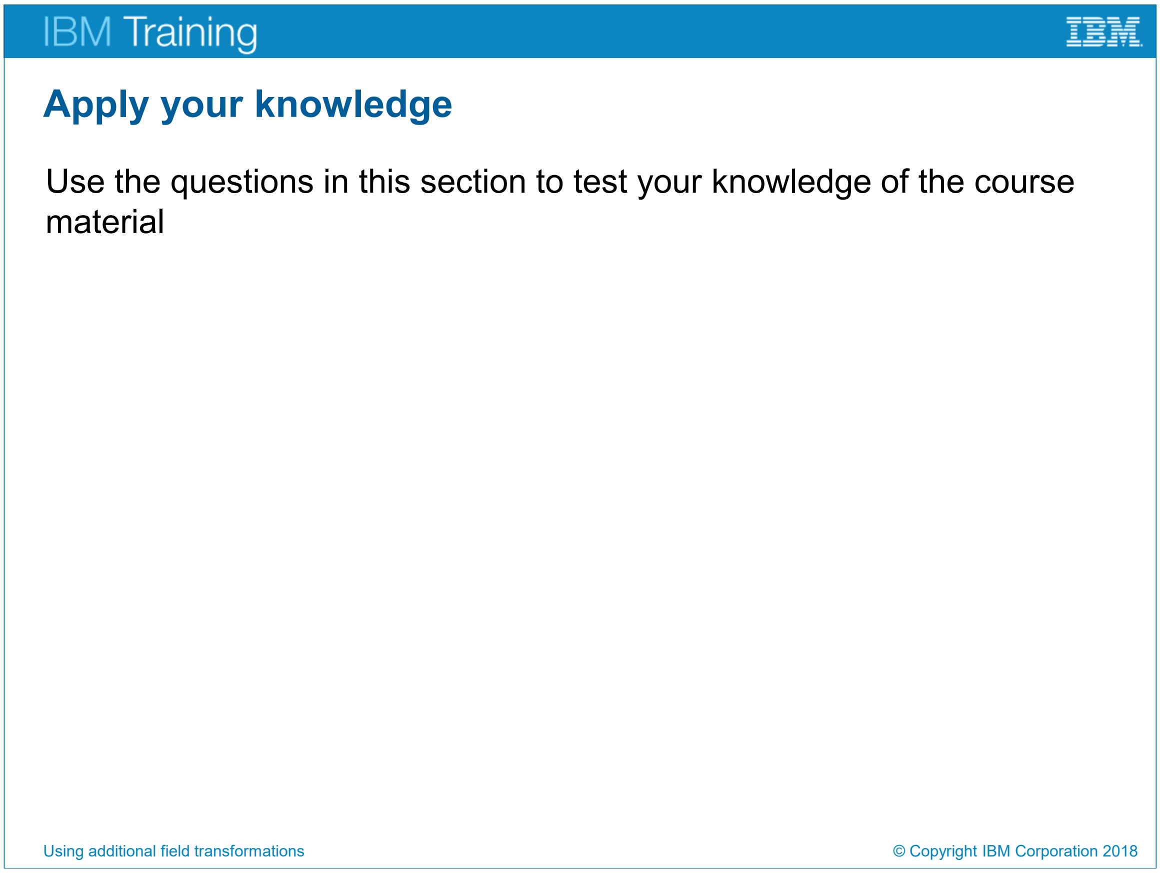
    The skewness of the original field is 0.917, indicating a substantial departure from normality. The transformed field is very close to a symmetric distribution given its skewness of -0.003. This can be enough motivation to use the transformed field in modeling next to or instead of the original field.
18. Close the **Data Audit** output window.

    This completes the demonstration. You will create a clean state for the exercises.
19. From the **File** menu, click **Close Stream**. Click **No** when asked to save the stream.
20. From the **File** menu, click **New Stream**.

    Leave IBM SPSS Modeler open for the exercise.

Results:
You have cleansed data using the Filler node and added new fields with the Binning node and the Transform node. Now that you have added these fields to the dataset you will be able to build better models later.

You will find the completed stream in the **C:\Training\0A058\Solutions** folder.



IBM Training

Apply your knowledge

Use the questions in this section to test your knowledge of the course material

Using additional field transformations

© Copyright IBM Corporation 2018

Apply your knowledge

Questions

Question 1: True or false: The Filler node will create a new field.

A. True

B. False

Question 2: A string field stores the values "1", "2", etc. For modeling purposes you need to convert the string values to integers 1, 2, etc. The node to use is:

A. Filler

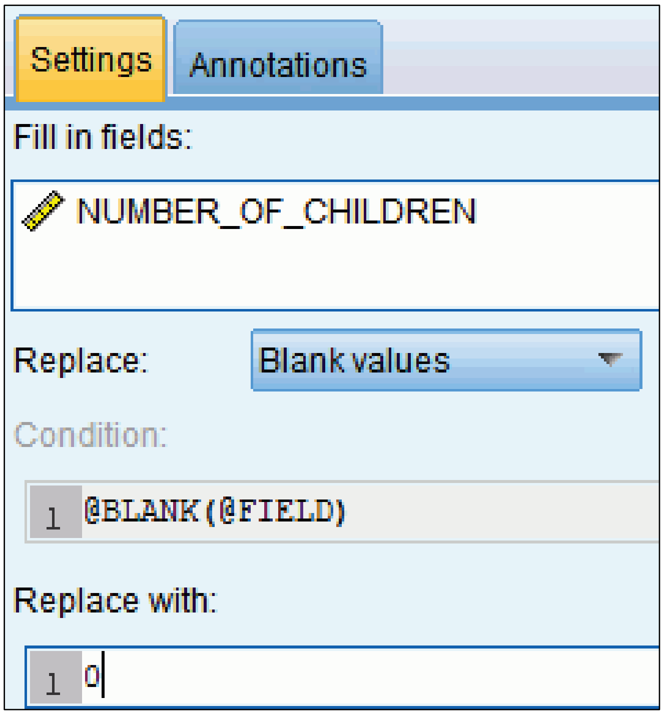
B. Binning

C. Transform

Question 3: A dataset comprises a field named NUMBER_OF_CHILDREN, for which the undefined value ($null$) is declared as a blank value. True or false: The specifications shown in the figure below will replace the $null$ value by 0 for NUMBER_OF_CHILDREN.

A. True

B. False



Fill in fields:

NUMBER_OF_CHILDREN

Replace: Blank values

Condition:

```
@BLANK(@FIELD)
```

Replace with:

```
0
```

Question 4: There are a number of reasons why one should or should not use the Binning node. Which of the following statements regarding the rationale behind binning are correct? (Select all apply.)

A. The effect of outliers in modeling can be reduced by binning because the outliers will be placed into the same bin.

B. Binning has the advantage of enhancing the amount of information about the field.

C. You may want to create custom groups rather than rely on the way values are grouped by the modeling algorithm.

D. Binning reduces the number of undefined ($null$) values on a field.

Question 5: Refer to the figures that follow. What were the settings to create the new field?

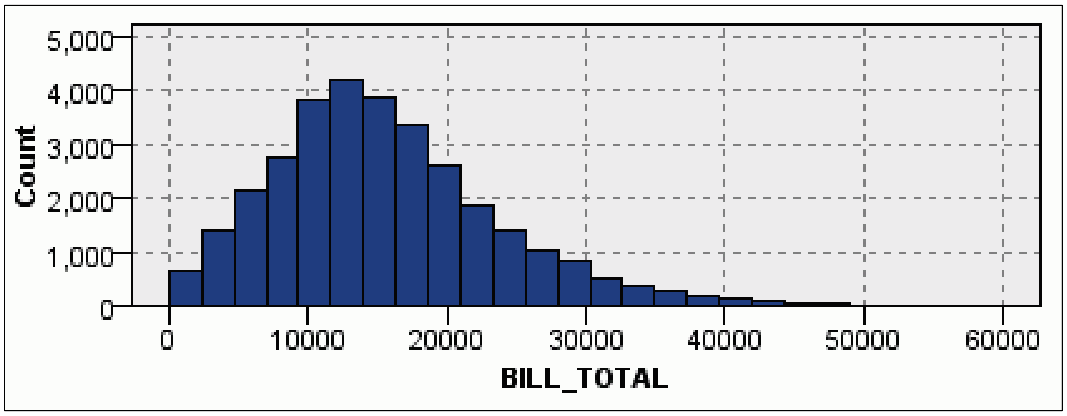
A. Binning method: Tiles; Tiling method: Tiles: Record count.

B. Binning method: Tiles; Tiling method: Tiles: Sum of values.

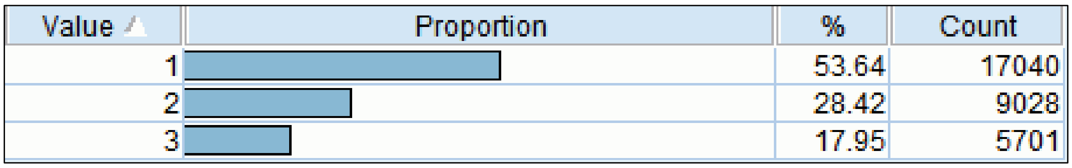
C. Binning method: Fixed Width; Bin width: 500

D. Binning method: Fixed Width; Bin width: 5000

Original field:



New field using the Binning node:



Value	Proportion	%	Count
1		53.64	17040
2		28.42	9028
3		17.95	5701

Question 6: True or false: Optimal binning always requires a supervisor field.

A. True

B. False

Question 7: True or false: The Transform node is located in the Field Ops palette.

A. True

B. False

Question 8: Suppose that a field has values between -100 and -1 (including boundary values -100 and -1), and that you want to explore how a Log10 transformation will change the field's distribution. What is the best value for the offset?

A. -100

B. 1

C. 100

D. 101

Answers to questions

Answer 1: B. False. The Filler node replaces values in an existing field.

Answer 2: A. Only the Filler node enables you to change the storage of a field.

Answer 3: A. True. Because the undefined ($null$) value is declared as blanks, replacing blank values will also replace the undefined ($null$) values.

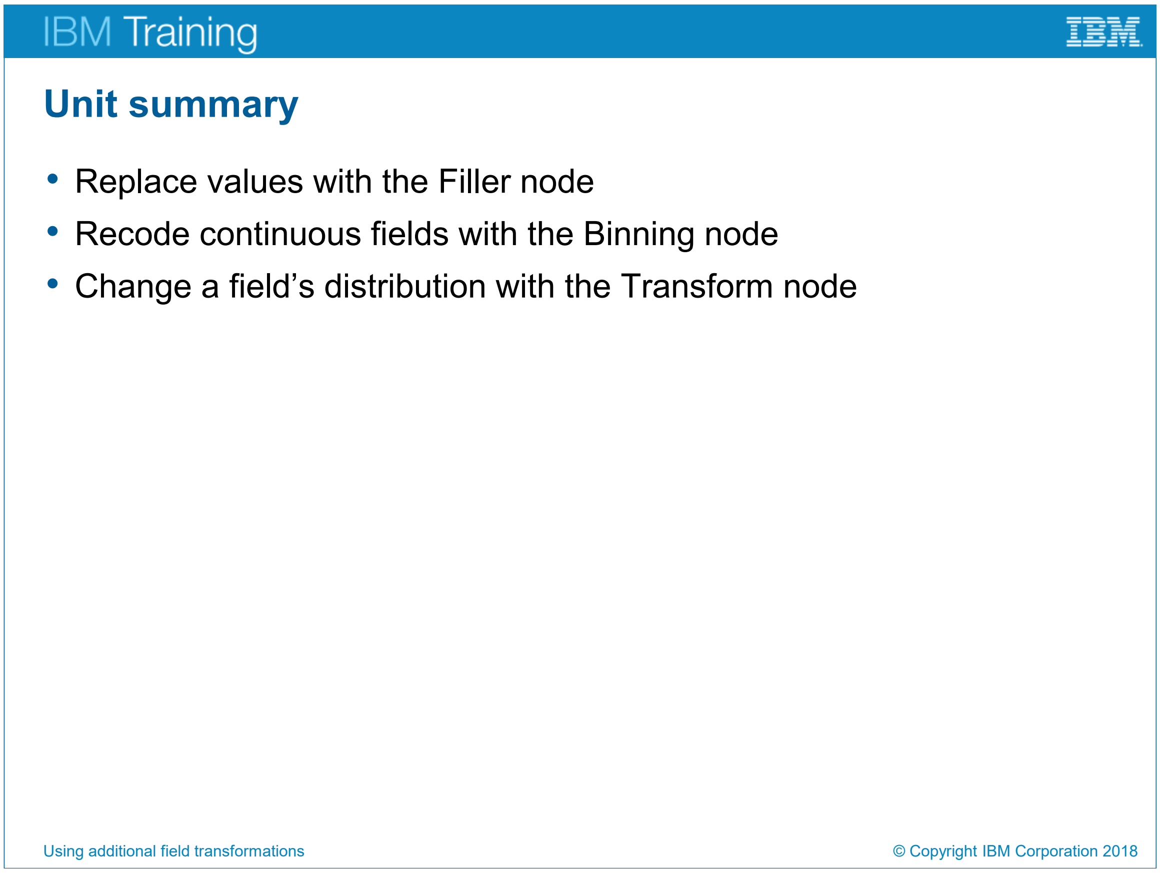
Answer 4: A, C. Outliers will be binned into the same category, thus reducing their effect. Also, binning enables you to create your own categories rather than being dependent on how a modeling algorithm bins values. Binning will reduce the amount of information, not enhance it. The number of undefined ($null$) values will not change, because $null$ values on the source field will be $null$ on the binned field as well.

Answer 5: B. The binning method did not create intervals of width 500 or 5000, because if so there would have been more than three categories. Also, the number of records is not the same in the categories, so that dismisses the tiling method of an equal record count and leaves the tiling binning method that is based on the sum of values.

Answer 6: A. True. Optimal binning requires a supervisor field by definition.

Answer 7: B. The Transform node is located in the Output palette, not in the Field Ops palette.

Answer 8: D. The Log 10 transformation is defined for values greater than 0; therefore, the best offset value is 101.



IBM Training

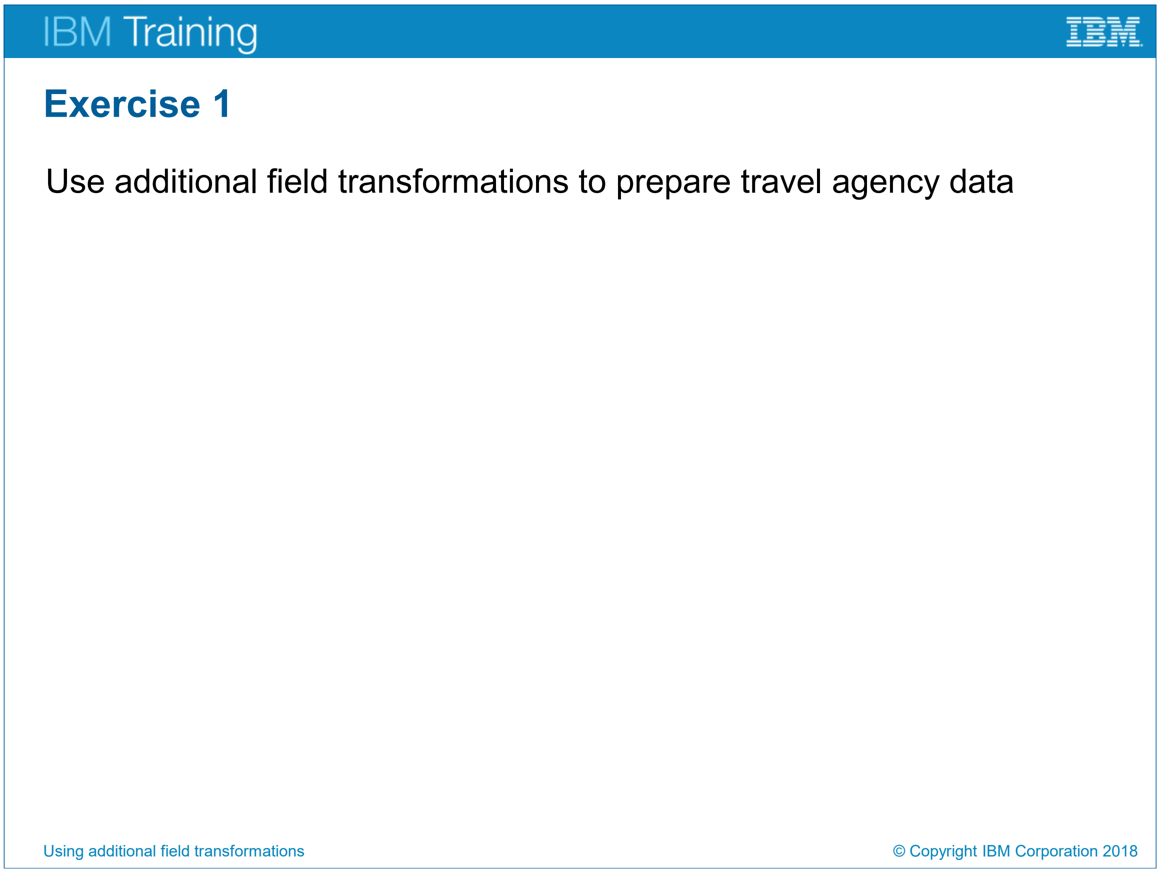
Unit summary

- Replace values with the Filler node
- Recode continuous fields with the Binning node
- Change a field’s distribution with the Transform node

Using additional field transformations

© Copyright IBM Corporation 2018

Unit summary



IBM Training

Exercise 1

Use additional field transformations to prepare travel agency data

Using additional field transformations

© Copyright IBM Corporation 2018

Exercise 1: Use additional field transformations to prepare travel agency data

Exercise 1:
Use additional field transformations to prepare travel agency data

Data file:	**customers_and_holidays.csv**
Data folder:	**C:\Training\0A058**

You will clean and enrich travel agency data.

- Use a **Var. File** node (**Sources** palette), to import data from the comma separated text file **customers_and_holidays.csv**, located in the **C:\Training\0A058** folder.

  In a **Type** node (**Field Ops** palette), declare **-999** as blank value for **DISTANCE_TO_BEACH**, and then instantiate the data by clicking the **Read Values** button.
- Replace the values for **REGION** and **COUNTRY** so that region and country name are always in upper case (for example: "East Anglia" should become "EAST ANGLIA" and "spain" should become "SPAIN").
- Replace blank values in **DISTANCE_TO_BEACH** by undefined ($null$) values.
- Recode **SPENT_LEISURE** into a new field named **SPENT_LEISURE_CATEGORY** that has three categories, with the same sum of **SPENT_LEISURE** in each of the categories. (Hint: for **Binning method**, choose **Tiles (equal count)**, in conjunction with **Tiling method Sum of values**.)

  Use a **Distribution** node (**Graphs** palette), to identify the category with the fewest number of records in it (apart from the $null$ category).
- Bin **SPENT_LEISURE** in such a way that its relationship with **SATISFIED** is optimal.

  Use a **Matrix** node (**Output** palette), to identify the category with the highest percentage of satisfied customers.
- Transform **SPENT_HOTEL** into a new field so that the distribution of this new field is less skewed than that of the source field, **SPENT_HOTEL**.

  Use a **Data Audit** node (**Output** palette) to compare the skewness of the transformed field to the skewness of the original field.
- Exit IBM SPSS Modeler without saving anything.

For more information about where to work and the exercise results, refer to the Task and Results section that follows. If you need more information to complete a task, refer to earlier demonstrations for detailed steps.

Exercise 1: Tasks and Results

Task 1. Import and instantiate the data.

- From the **Sources** palette, add a **Var. File** node to import the data from **customers_and_holidays.csv**, located in the **C:\Training\0A058** folder.
- From the **Field Ops** palette, add a **Type** node downstream from the **Var. file** node.
- Edit the **Type** node:
  - On the **DISTANCE_TO_BEACH** row, click the cell in the **Missing** column, and then click **Specify**.
  - In the **DISTANCE_TO_BEACH Values** dialog box, enable the **Define blanks** option.
  - Declare **-999** as blank by typing it into the **Missing values** text box.
  - Click **OK** to return to the main dialog box.
  - Click **Read Values** to instantiate the data.

Task 2. Correct the spelling.

- From the **Field Ops** palette, add a **Filler** node downstream from the **Type** node.
- Edit the **Filler** node:
  - On the **Settings** tab, under **Fill in fields**, select **REGION** and **COUNTRY**.
  - Set the **Replace** field to **Always**.
  - Under **Replace with**, enter **lowertoupper (@FIELD)**.

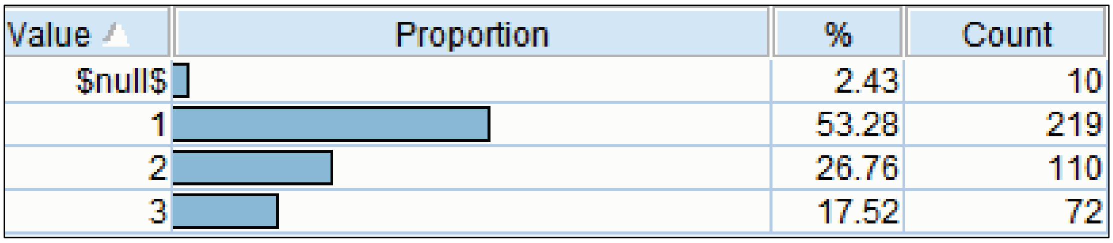
Task 3. Replace blanks by undefined values.

- From the **Field Ops** palette, add a **Filler** node downstream from the first **Filler** node.
- Edit the **Filler** node:
  - Under **Fill in fields**, select **DISTANCE_TO_BEACH**.
  - Beside **Replace**, click **Blank values** from the list.
  - Under **Replace with**, type **undef**.

Task 4. Recode a field into three categories, based on the sum of values.

- From the **Field Ops** palette, add a **Binning** node downstream from the second **Filler** node.
- Edit the **Binning** node:
  - Beside **Bin fields**, select **SPENT_LEISURE**.
  - Beside **Binning method**, click **Tiles (equal count)** from the list.
  - Beside **Custom tile extension**, type **_CATEGORY**.
  - Disable the **Decile (10)** option.
  - Enable the **Custom N** option and then set its value to **3**.
  - Beside **Tiling method**, click **Sum of values**.
- From the **Graphs** palette, add a **Distribution** node downstream from the **Binning** node.
- Edit the **Distribution** node:
  - Beside **Field**, select **SPENT_LEISURE_CATEGORY**.
  - Run the **Distribution** node.

The results appear as follows



Value	Proportion	%	Count
$null$		2.43	10
1		53.28	219
2		26.76	110
3		17.52	72

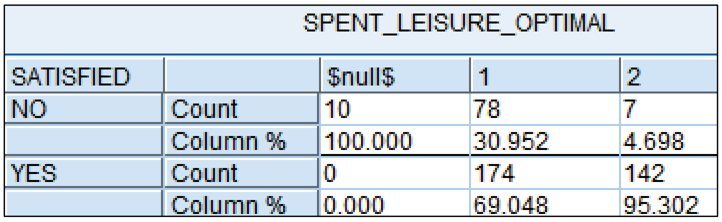
The third category has the smallest number of customers, as could be expected because the tiling method was such that the sum of values of SPENT_LEISURE was the same in the categories.

Notice that 10 records have a $null$ value for the new field (they had an undefined value on the source field SPENT_LEISURE).

Task 5. Bin a field optimally with respect to a target.

- From the **Field Ops** palette, add a **Binning** node downstream from the **first Binning** node.
- Edit the **Binning** node:
  - Beside **Bin fields**, select **SPENT_LEISURE**.
  - Beside **Binning method**, click **Optimal** from the list.
  - Beside **Supervisor field**, select **SATISFIED**.
- From the **Output** palette, add a **Matrix** node downstream from the **second Binning** node.
- Edit the **Matrix** node:
  - Beside **Rows**, select **SATISFIED**.
  - Beside **Columns**, select **SPENT_LEISURE_OPTIMAL**.
  - Click the **Appearance** tab, and then enable the **Percentage of column** option.
  - Run the **Matrix** node.

The results appear as follows:



		SPENT_LEISURE_OPTIMAL		
SATISFIED		$null$	1	2
NO	Count	10	78	7
	Column %	100.000	30.952	4.698
YES	Count	0	174	142
	Column %	0.000	69.048	95.302

Optimal binning recoded SPENT_LEISURE into two categories. The percentage satisfied customers is 69.048 in the first category and 95.302 in the second category.

Task 6. Transform a field to modify its distribution.

- From the **Output** palette, add a **Transform** node downstream from the **second Binning** node.
- Edit the **Transform** node:
  - Beside **Fields**, select **SPENT_HOTEL**.
  - Run the **Transform** node.

  The square root transformation appears to make the distribution less skewed.
  - Click the **Square Root** thumbnail.
  - Click the **Generate** menu, and then click **Derive node**.
  - Click **OK**.
  - Close the **Transform** dialog box.
- Add the generated **Supernode_Transform** node downstream from the **second Binning** node.
- From the **Output** palette, add a **Data Audit** node downstream from the **Supernode_Transform** node.
- Edit the **Data Audit** node:
  - On the **Settings** tab, select **Use custom fields.**
  - Select **SPENT_HOTEL** and **SPENT_HOTEL_Square_Root**.
  - Run the **Data Audit** node

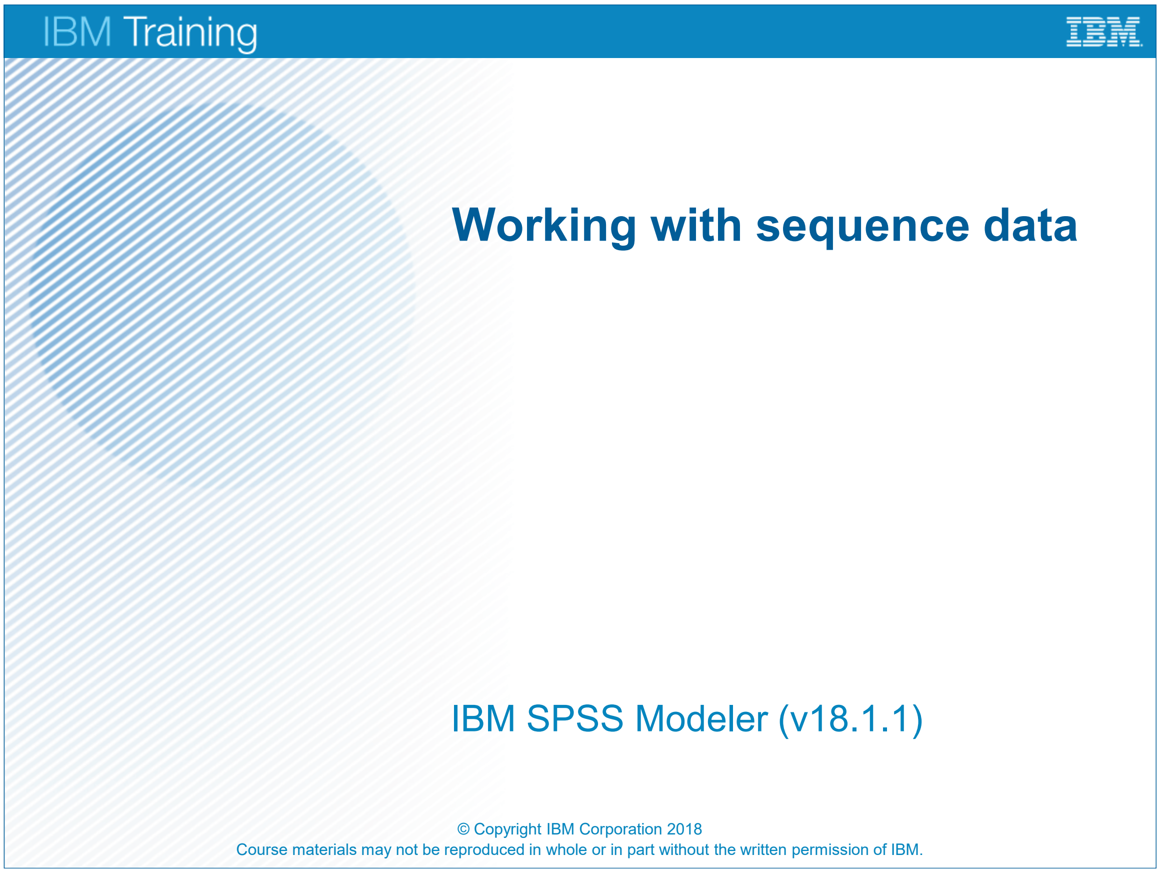
  The skewness for SPENT_HOTEL changed from 0.710 to 0.061 by applying the square root transformation.

Task 7. Exit IBM SPSS Modeler.

- From the **File** menu, click **Exit**, and then exit **IBM SPSS Modeler** without saving anything.

You will find the solution results in the **C:\Training\0A058\Solutions** folder.

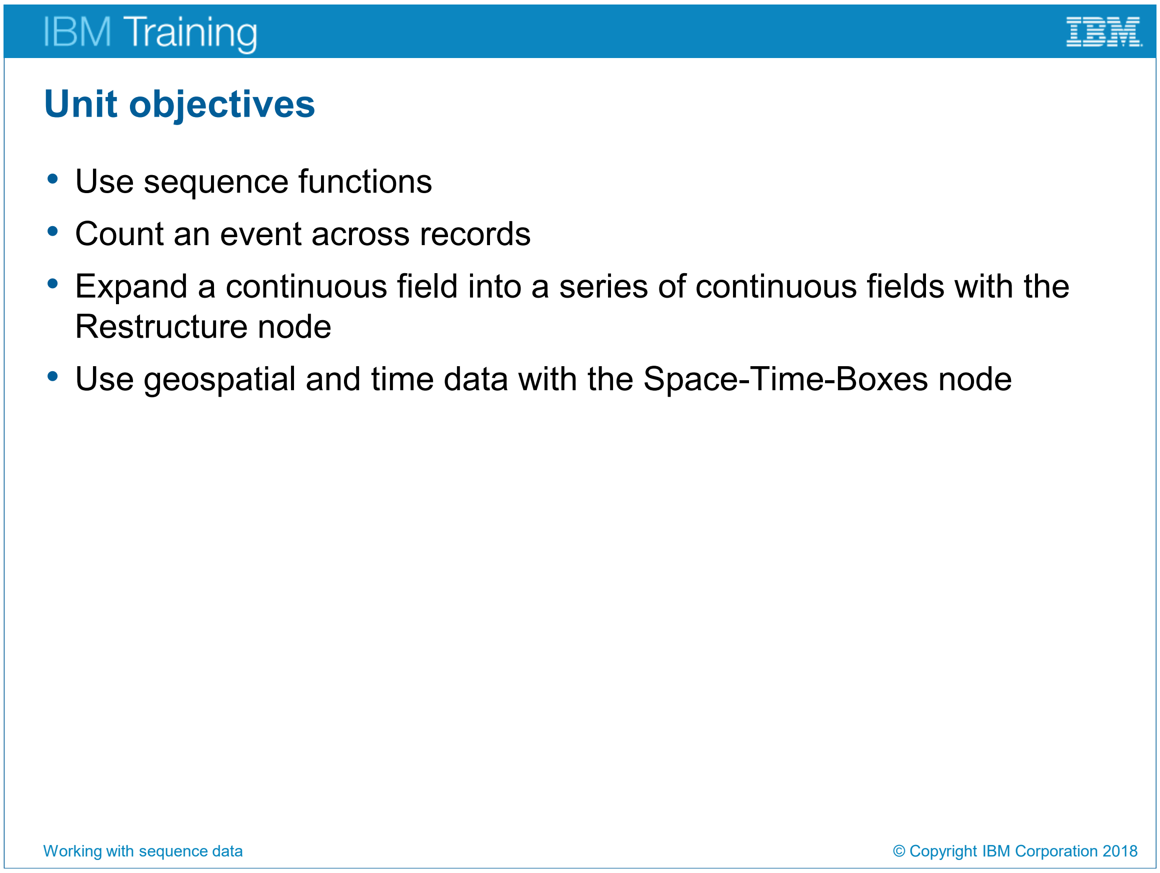
Unit 3 Working with sequence data



IBM Training

Working with sequence data

IBM SPSS Modeler (v18.1.1)



IBM Training

Unit objectives

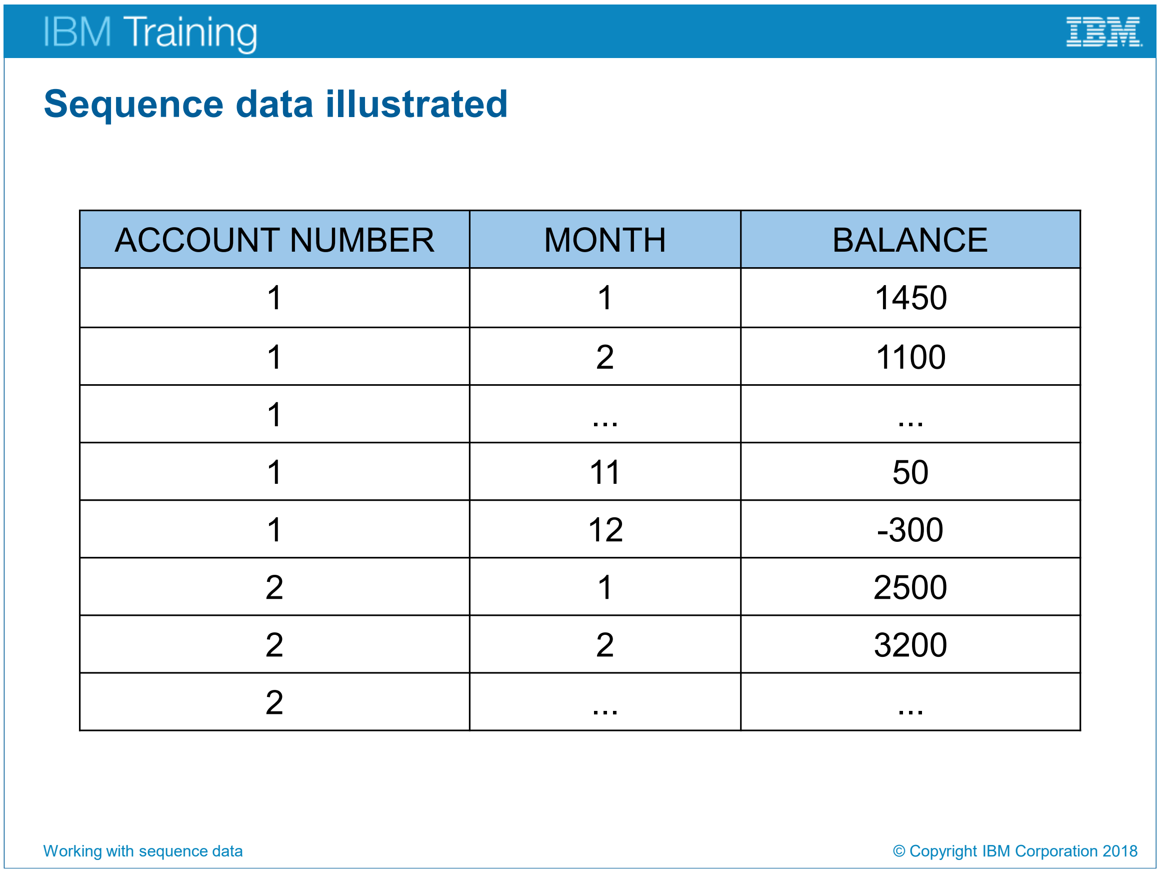
- Use sequence functions
- Count an event across records
- Expand a continuous field into a series of continuous fields with the Restructure node
- Use geospatial and time data with the Space-Time-Boxes node

Working with sequence data

Unit objectives

Before reviewing this unit, you should be familiar with:

- working with IBM SPSS Modeler (streams, nodes, SuperNodes, palettes)
- importing data (Var. File node)
- defining measurement levels, roles, blanks, and instantiating data (Type node)
- selecting fields and records (Filter node, Select node)
- CLEM and the Expression Builder
- deriving fields (Derive node)
- date and time-, conversion-, string-, statistical- and missing value functions
- the Aggregate node
- the SetToFlag node
- the Transpose node
- examining the data with graphs (Distribution node, Histogram node) and tables (Table node, Data Audit node, Statistics node, Matrix node, Means node)



IBM Training

Sequence data illustrated

ACCOUNT NUMBER	MONTH	BALANCE
1	1	1450
1	2	1100
1	...	...
1	11	50
1	12	-300
2	1	2500
2	2	3200
2	...	...

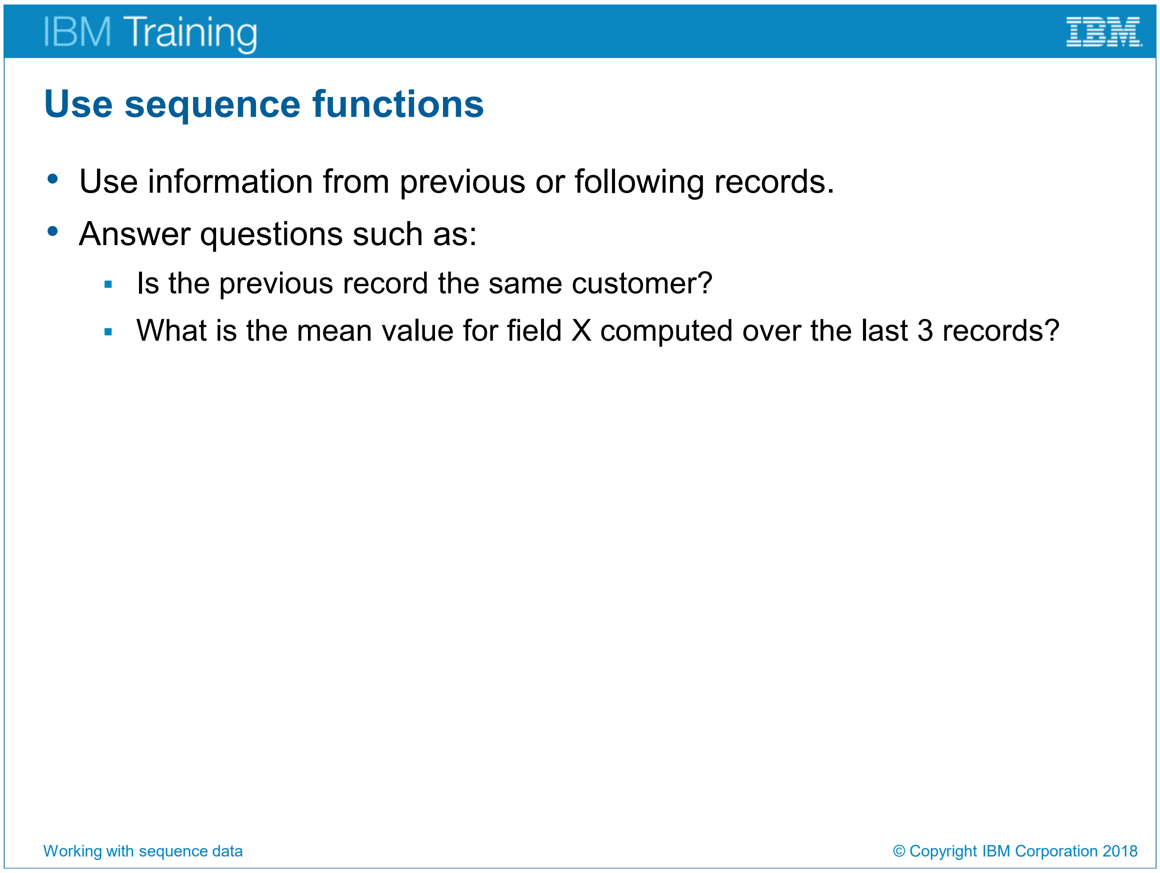
Working with sequence data

© Copyright IBM Corporation 2018

Sequence data illustrated

In many situations, each record in a dataset is considered as an individual case, independent of all others. In such instances, the order of records is unimportant for analyses. However, in some datasets the sequence of records is extremely important. This often occurs with time-structured data in which the order of the records represents a sequence of events. In this case, each record is a snapshot at a particular instant in time. The instantaneous values may be of interest, or how they vary over time.

This slide shows an example of sequence data, where a bank holds data of individual accounts, with a record for each month.


IBM Training

Use sequence functions

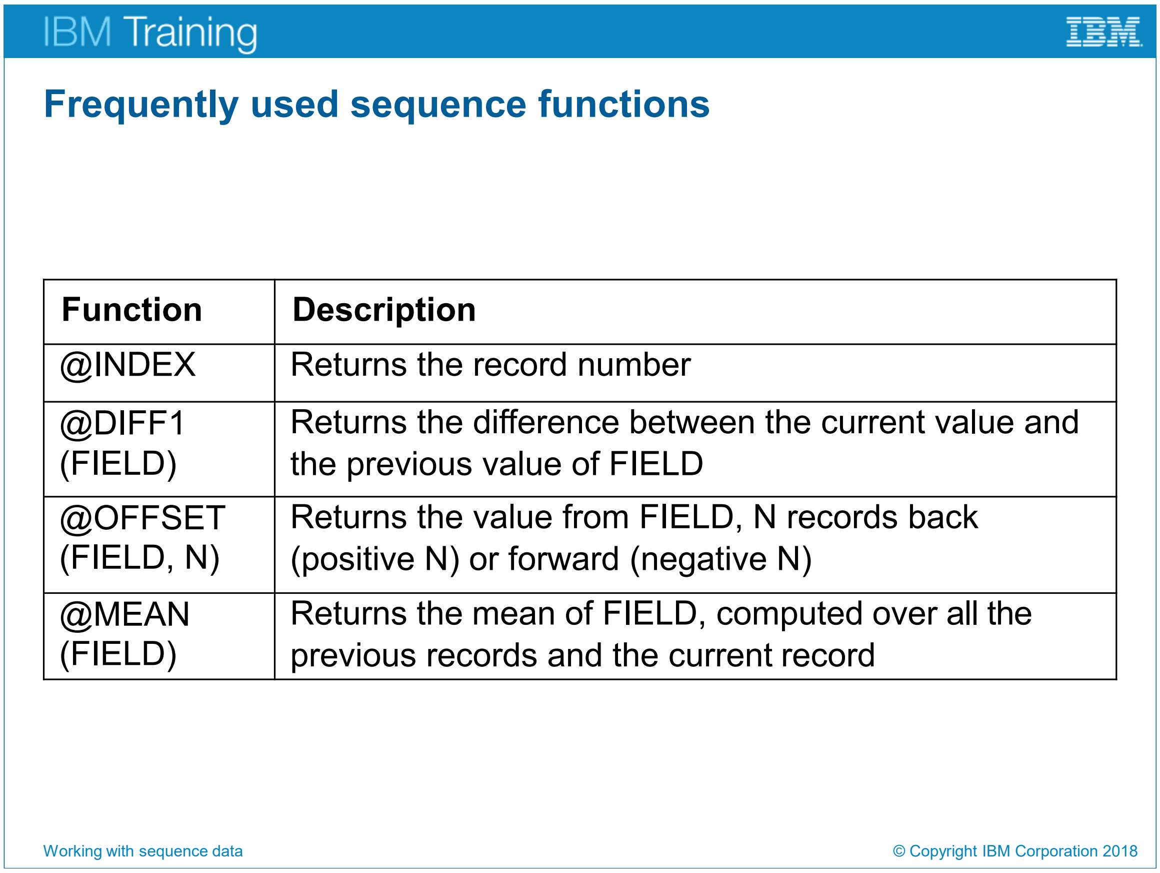
- Use information from previous or following records.
- Answer questions such as:
  - Is the previous record the same customer?
  - What is the mean value for field X computed over the last 3 records?

Working with sequence data © Copyright IBM Corporation 2018

Use sequence functions

Sequence functions use information from previous or following records.

These functions are typically used to check if the previous record is the same customer as the current record, or to compute a moving average over the last N records. The latter is especially relevant when you have time series data. For example, a company may collect data on their monthly revenues and create a moving average time series that smooths (flattens the peaks and troughs) the original time series to see the trend more clearly.



IBM Training

Frequently used sequence functions

Function	Description
@INDEX	Returns the record number
@DIFF1 (FIELD)	Returns the difference between the current value and the previous value of FIELD
@OFFSET (FIELD, N)	Returns the value from FIELD, N records back (positive N) or forward (negative N)
@MEAN (FIELD)	Returns the mean of FIELD, computed over all the previous records and the current record

Working with sequence data

Frequently used sequence functions

Sequence functions are prefixed with the at sign (@) and are in upper case.

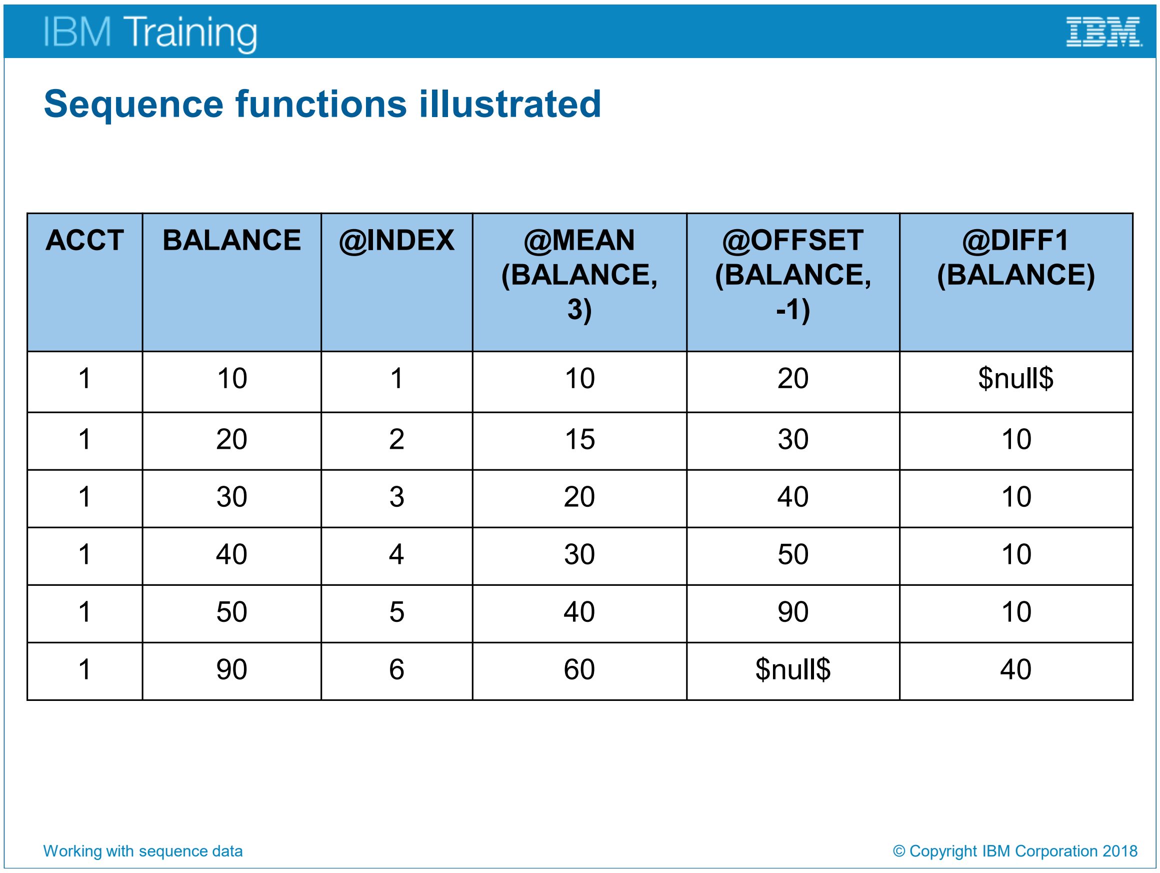
The simplest function is @INDEX; which returns the consecutive number of the record in the dataset. A field that stores the record number lets you sort the records back into their original order at any moment downstream.

The @OFFSET function is used to retrieve values in previous or following records. The @OFFSET function requires a field name and an integer N as arguments. The integer specifies how many records should be looked back (when N is a positive integer) or looked ahead (when N is a negative integer).

The @MEAN function returns the mean of a certain field, over all previous records and the current record, or over the previous N-1 records and the current record. In the latter case you need a second argument N for this function. The @MEAN function is frequently used to compute a moving average.

Similar to the MEAN function there are functions for the sum (@SUM) minimum (@MIN), maximum (@MAX), and standard deviation (@SDEV).

In the Expression Builder, these functions, among others, are found in the Sequence function group.



IBM Training

Sequence functions illustrated

ACCT	BALANCE	@INDEX	@MEAN (BALANCE, 3)	@OFFSET (BALANCE, -1)	@DIFF1 (BALANCE)
1	10	1	10	20	$null$
1	20	2	15	30	10
1	30	3	20	40	10
1	40	4	30	50	10
1	50	5	40	90	10
1	90	6	60	$null$	40

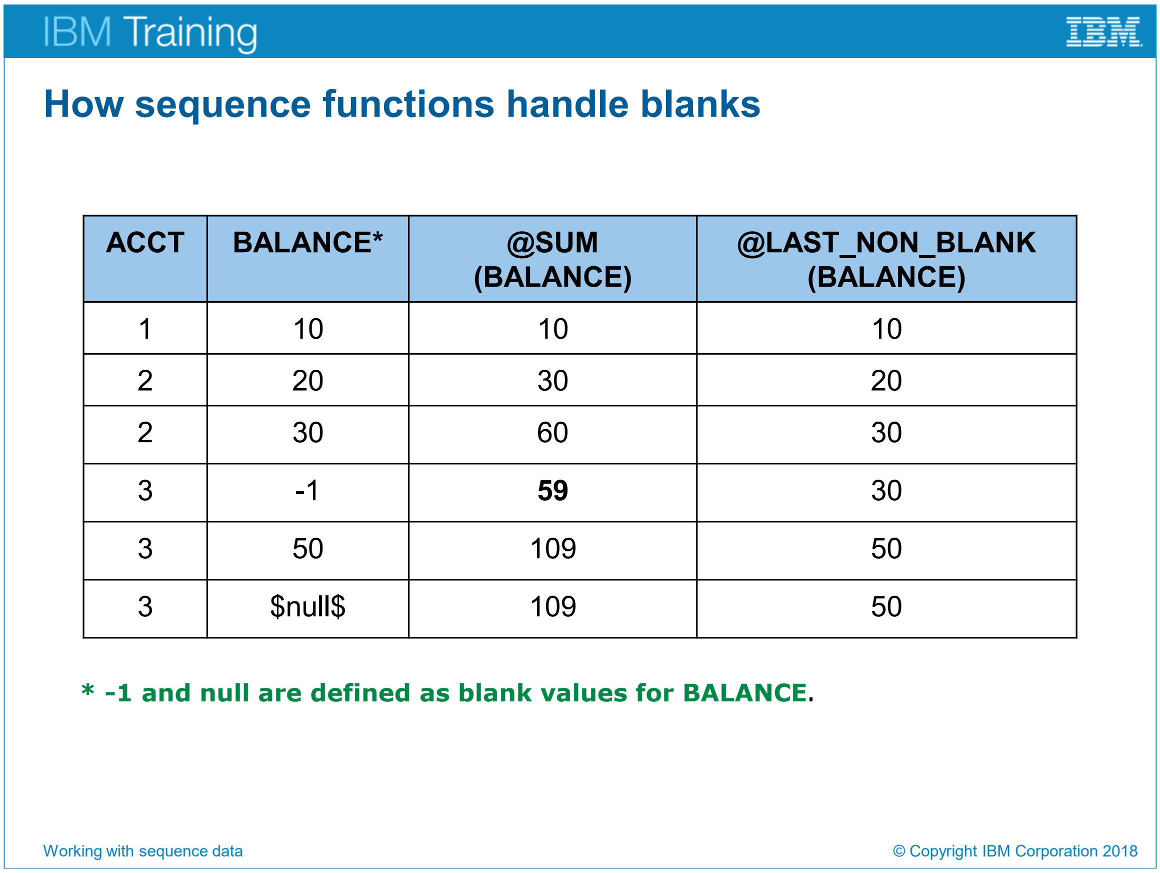
Working with sequence data

© Copyright IBM Corporation 2018

Sequence functions illustrated

This slide illustrates a number of sequence functions:

- @INDEX returns the record number.
- @MEAN returns the moving average over the last three records (the two preceding records and the current one). For the first two records, the function uses the available information, rather than returning an undefined value ($null$).
- @OFFSET returns the value of the next record because of the negative second argument.
- @DIFF1 returns the difference between the current record and the preceding record. The first record has an undefined ($null$) value.



IBM Training

How sequence functions handle blanks

ACCT	BALANCE*	@SUM (BALANCE)	@LAST_NON_BLANK (BALANCE)
1	10	10	10
2	20	30	20
2	30	60	30
3	-1	**59**	30
3	50	109	50
3	$null$	109	50

*** -1 and null are defined as blank values for BALANCE**.

Working with sequence data

© Copyright IBM Corporation 2018

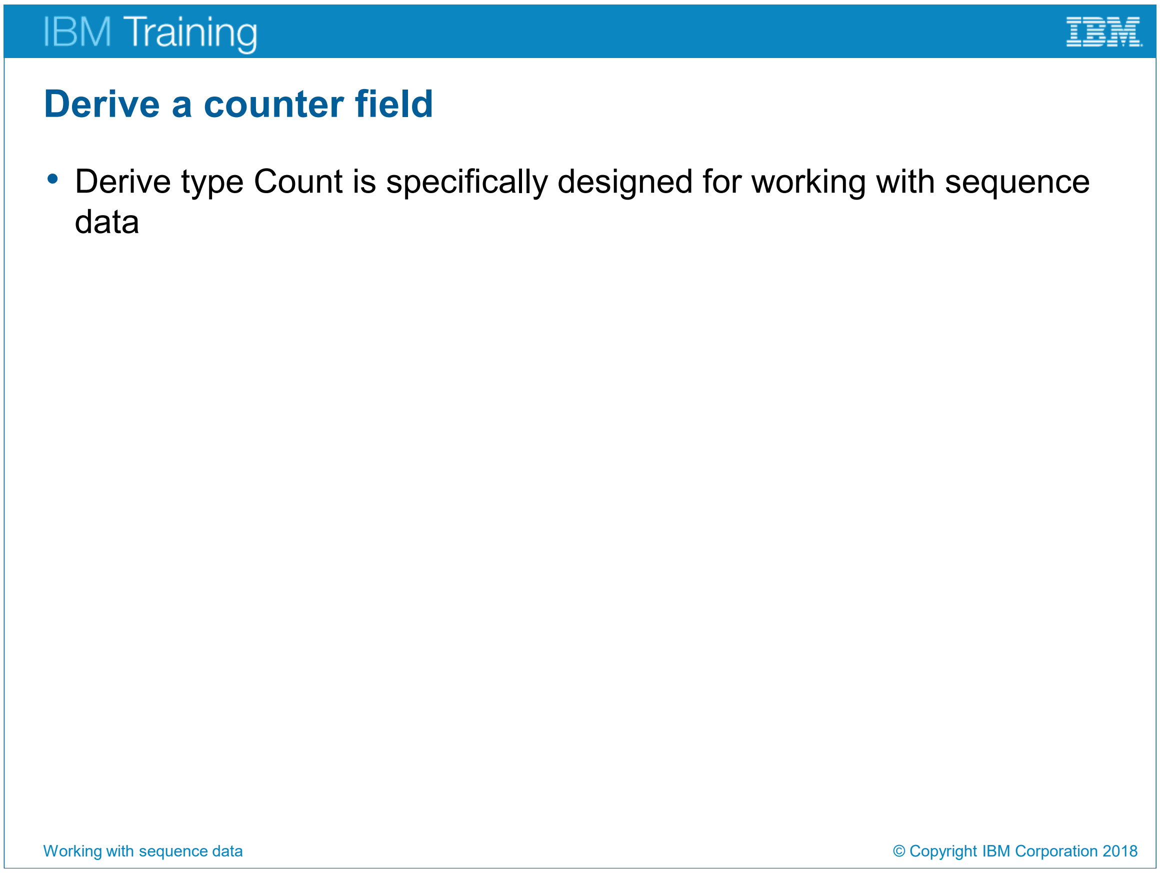
How sequence functions handle blanks

This slide shows an example of how sequence functions handle blanks. For BALANCE, -1 and the undefined ($null) value are declared as blank value.

The @SUM function does not take into account that -1 is declared as blank and includes the value in its computation. In general, you should be forewarned when you have user-defined blank values because sequence functions ignore blank definitions.

When the source field is undefined ($null$), the @SUM function will be equal to the sum of the previous record. This also suggests a solution for the issue that a sequence ignores user-defined blanks: Use a Filler node upstream to replace blank values with the undefined ($null$) value (refer to the *Using additional field transformations* unit in this course for a presentation of the Filler node).

A sequence function that specifically deals with blanks is @LAST_NON_BLANK. This function returns the last value that was not blank. In this example, @LAST_NON_BLANK (BALANCE) returns the value 30 for record #4, and 50 for record #6. If the undefined ($null$) value would not have been declared as blank for BALANCE, the result for record #6 would be $null$.



IBM Training

Derive a counter field

- Derive type Count is specifically designed for working with sequence data

Working with sequence data

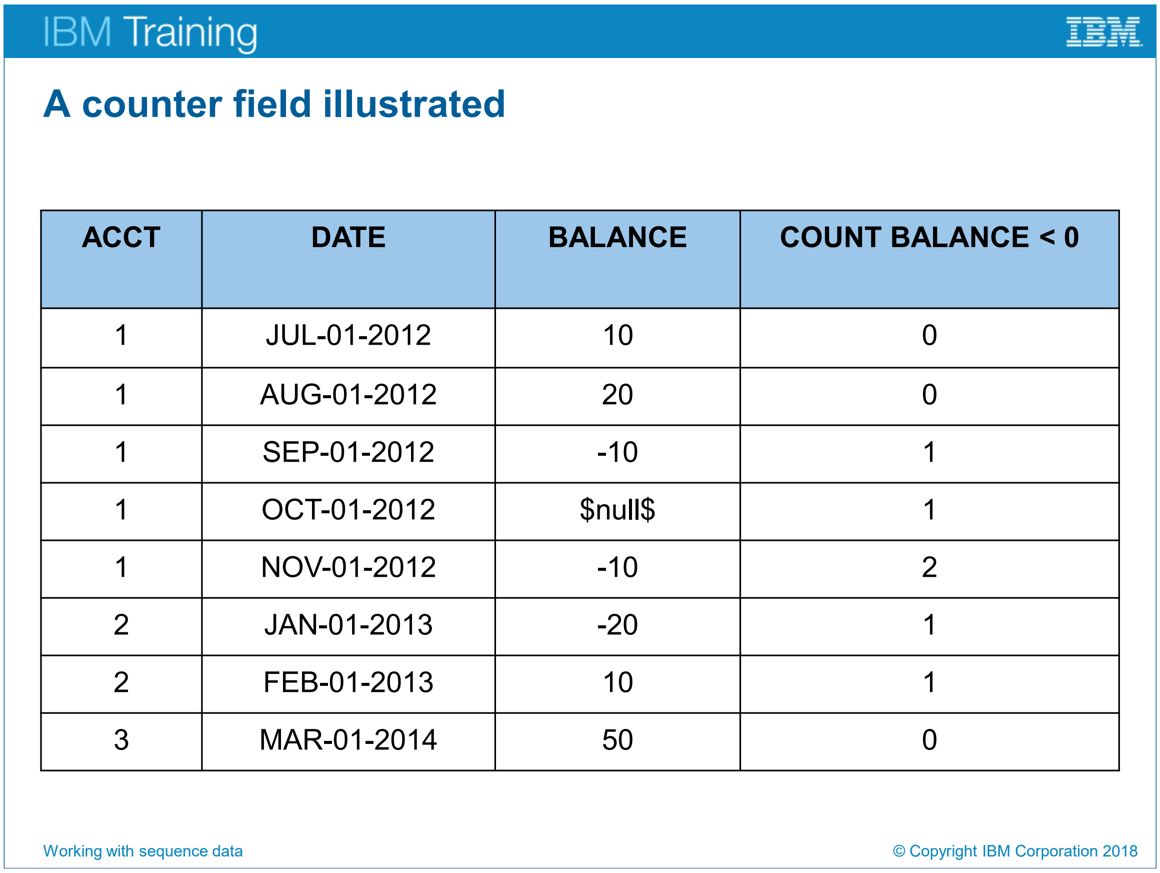
© Copyright IBM Corporation 2018

Derive a counter field

In the Derive dialog box, you have the option to derive a field as Formula, Flag, Nominal, Conditional, Count, and State. Refer to the *Introduction to IBM SPSS Modeler and Data Science* course for more information about the first four options.

The Derive type Count in the Derive node is specifically designed for sequence data. A field that is derived as Count calculates the number of times that a condition has been met across records.

The Derive type State is a subtle variation on the Count option; it is almost never used. Refer to the Help for more information.



IBM Training

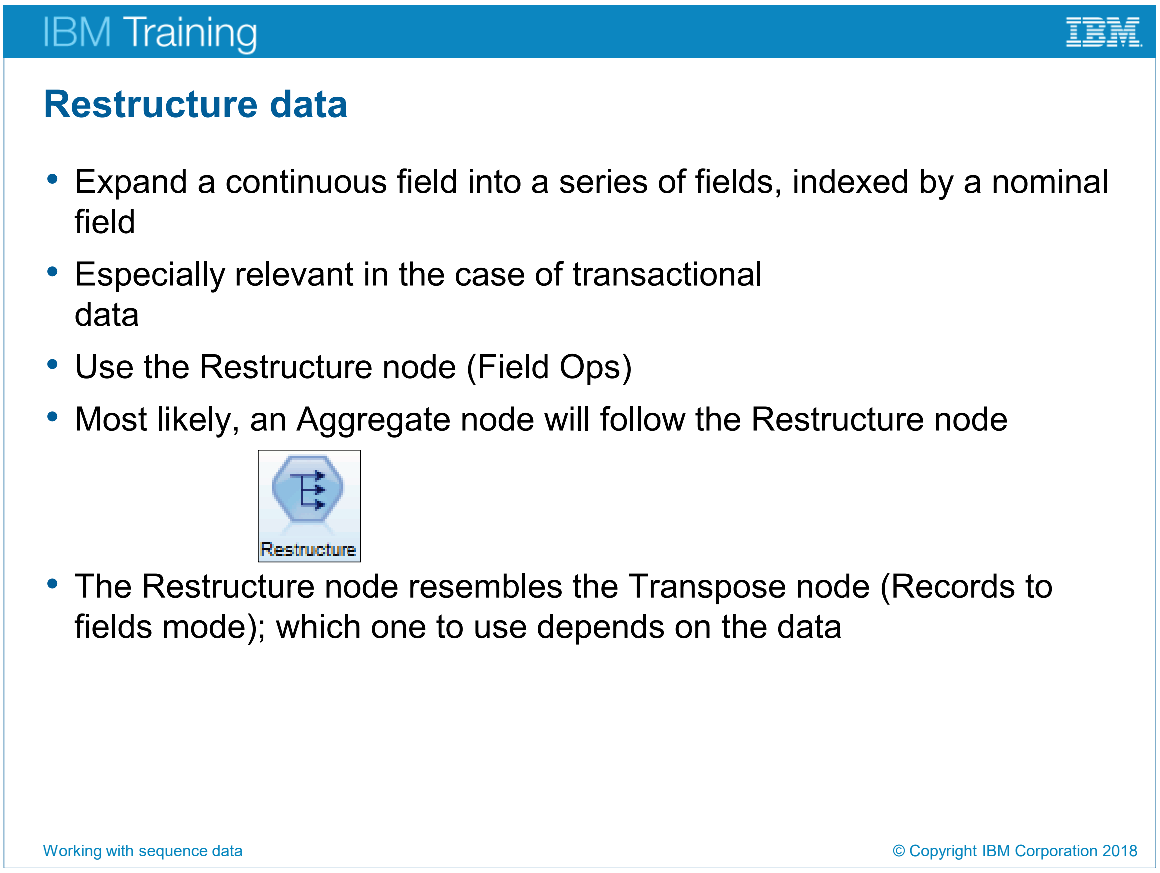
A counter field illustrated

ACCT	DATE	BALANCE	COUNT BALANCE < 0
1	JUL-01-2012	10	0
1	AUG-01-2012	20	0
1	SEP-01-2012	-10	1
1	OCT-01-2012	$null$	1
1	NOV-01-2012	-10	2
2	JAN-01-2013	-20	1
2	FEB-01-2013	10	1
3	MAR-01-2014	50	0

Working with sequence data

A counter field illustrated

The COUNT BALANCE < 0 field counts the number of times that the BALANCE is negative, across records. The counter initializes at 0, and is incremented by 1 when a negative value is encountered. When the value is positive or undefined ($null$) the counter remains the same. The field resets to 0 when a new account is encountered.



IBM Training

Restructure data

- Expand a continuous field into a series of fields, indexed by a nominal field
- Especially relevant in the case of transactional data
- Use the Restructure node (Field Ops)
- Most likely, an Aggregate node will follow the Restructure node

Restructure

- The Restructure node resembles the Transpose node (Records to fields mode); which one to use depends on the data

Working with sequence data © Copyright IBM Corporation 2018

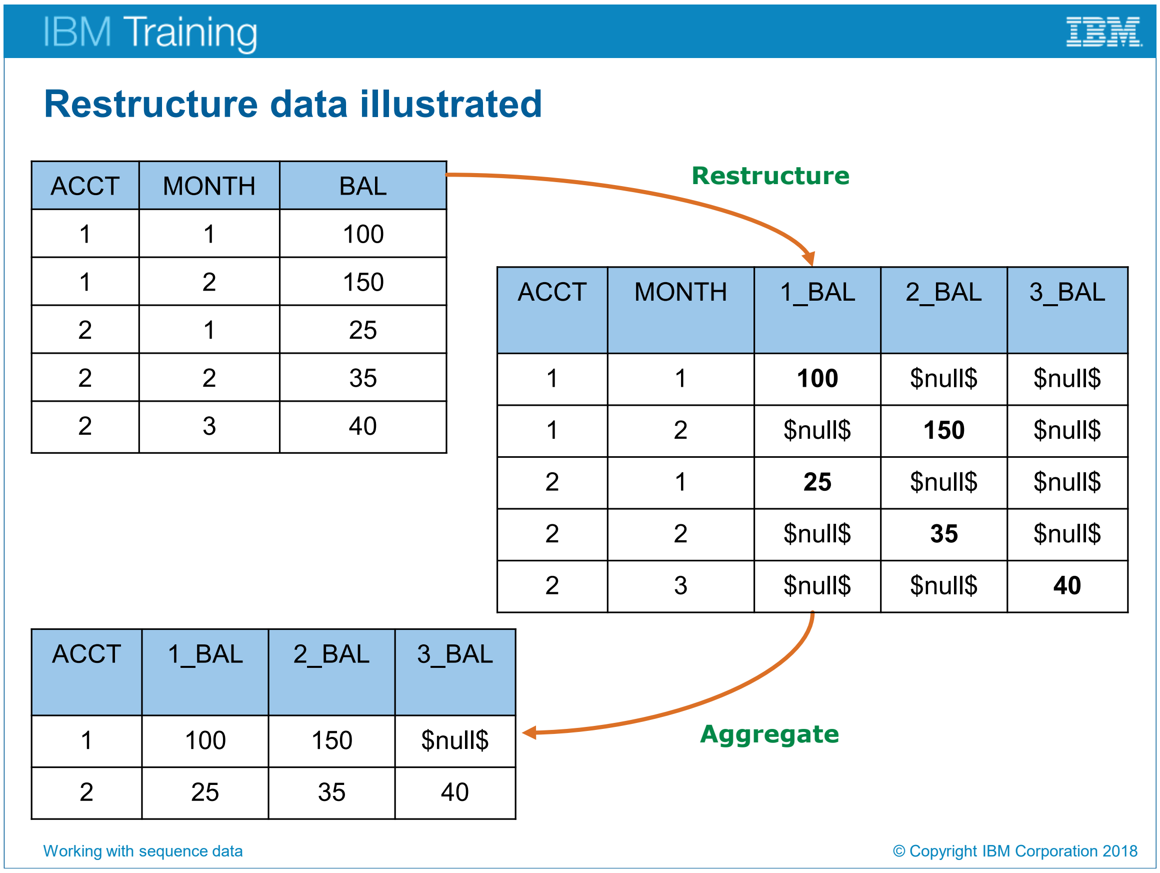
Restructure data

Occasionally you will have a data structure that is not suited for the analyses that you want to run. For example, a customer may have purchased products at different times, and each purchase is a record. Data of this type, where not persons but transactions define unique records, is referred to as transactional data.

In transactional datasets you might want to know if the revenue for product A is related to the revenue for product B. To answer this question, however, you need a data structure that has only one record per customer with the revenue for A and the revenue for B as fields.

The Restructure node, can spread out the information contained in one field into multiple fields. The next step is to aggregate the data. This cannot be done in the Restructure node itself, but requires a separate Aggregate node.

The Transpose node, presented in the *Introduction to IBM SPSS Modeler and Data Science* course, also lets you restructure data. Which node to choose will depend on the data, as illustrated in the demonstration that follows.



Restructure data illustrated

ACCT	MONTH	BAL
1	1	100
1	2	150
2	1	25
2	2	35
2	3	40

ACCT	MONTH	1_BAL	2_BAL	3_BAL
1	1	**100**	$null$	$null$
1	2	$null$	**150**	$null$
2	1	**25**	$null$	$null$
2	2	$null$	**35**	$null$
2	3	$null$	$null$	**40**

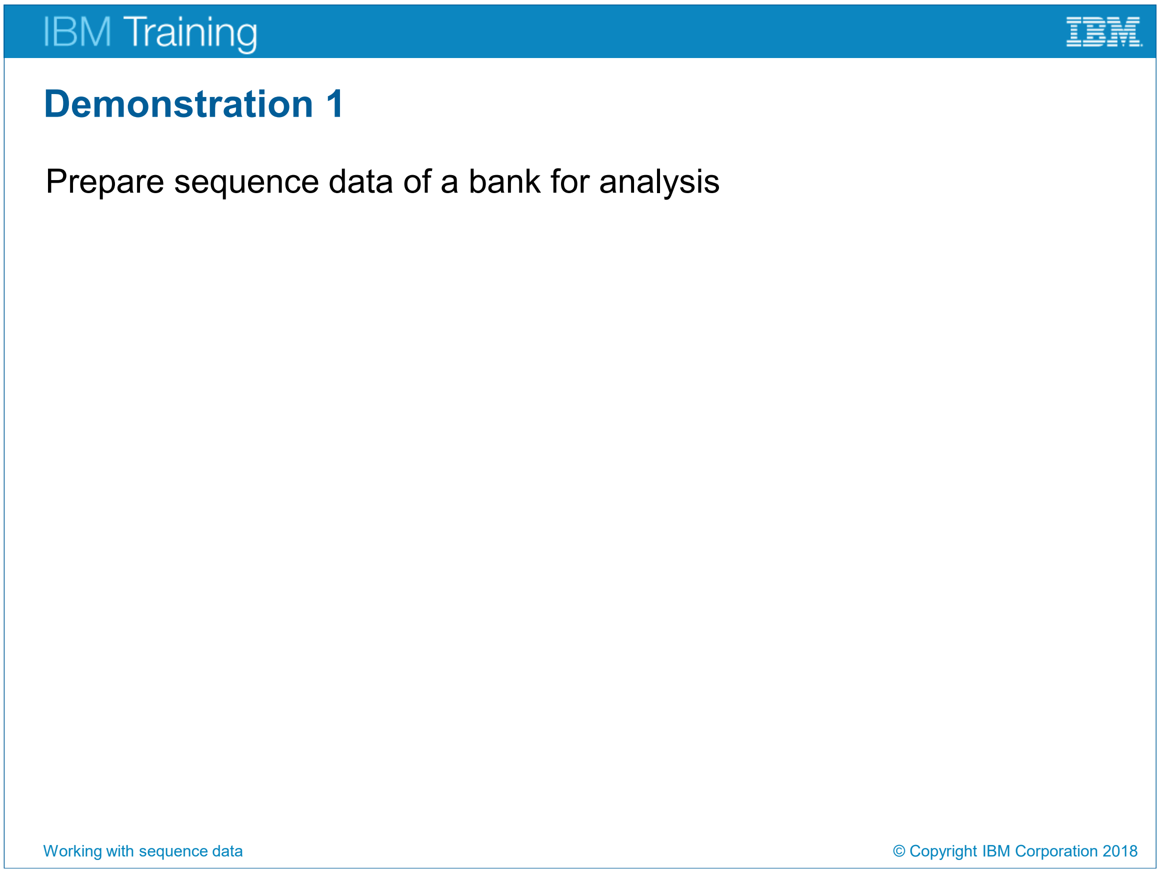
ACCT	1_BAL	2_BAL	3_BAL
1	100	150	$null$
2	25	35	40

Restructure data illustrated

This slide illustrates how a transactional dataset sis transformed into a dataset with one record per customer.

The Restructure operation expands the BAL field into a number of fields, which are indexed by month. For example, 1_BAL stores the balance for the first month.

Restructure does not affect the data structure because a customer still has as many records as there were transactions. Thus, an Aggregate node is required in the second step to create a dataset with one record per customer.



IBM Training

Demonstration 1

Prepare sequence data of a bank for analysis

Working with sequence data

© Copyright IBM Corporation 2018

Demonstration 1: Prepare sequence data of a bank for analysis

Demonstration 1: Prepare sequence data of a bank for analysis

Purpose:
You will apply various transformations to sequence data, such as creating a record identifier and a moving average. You will restructure a transactional dataset into a dataset with one record per customer, which is the required data structure for correlational analyses.

Data file:	**bank_balances.csv**
	bank_services.csv
Data folder:	**C:\Training\0A058**

Task 1. Start IBM SPSS Modeler and set the working folder.

1. From the **Start** menu, expand **IBM SPSS Modeler 18.1**, and then click **IBM SPSS Modeler 18.1**. When a splash window displays, click **Cancel**.

   If you have already configured IBM SPSS Modeler in a previous demonstration or exercise, you can skip to Task 2.
2. From the **File** menu, click **Set Directory**, navigate to the **C:\Training\0A058** folder and then click **Set**.

Task 2. Create a record identifier.

You will derive a field that serves as record identifier. Such a field can be used to resort the data downstream back into the original order.

1. From the **Sources** palette, add a **Var. File** node to the stream canvas, edit the node, and import data from **bank_balances.csv**, located in the **C:\Training\0A058** folder.
2. From the **Field Ops** palette, add a **Derive** node downstream from the **Var. File** node.
3. Edit the **Derive** node.
4. Under **Derive field**, type **RECORD ID**.
5. Under **Formula**, enter **@INDEX**.
6. Click **Preview**.

   A field has been added that uniquely identifies the records.
7. Close the **Preview** output window.
8. Close the **Derive** dialog box.

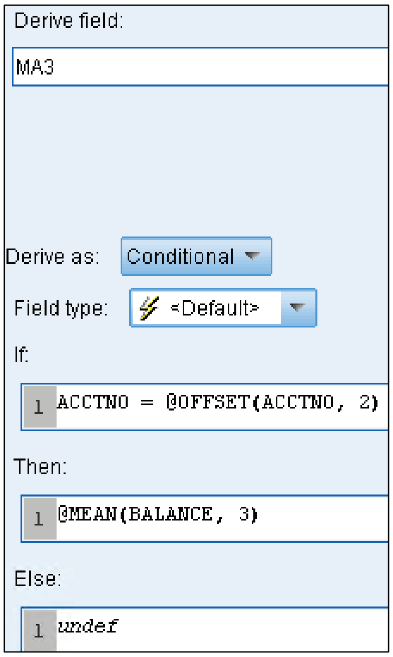
Task 3. Create a moving average.

You will create a three-record moving average for BALANCE, which enables you to determine a trend (the latter is not pursued here). The value will reinitialize when a new account is encountered.

The mean will not be calculated unless the second record before the current one has the same account number. Thus, the mean will not be calculated for the first two months for each account because the data are sorted by ACCTNO and MONTH.

1. From the **Field Ops** palette, add a **Derive** node downstream from the **Derive** node named **RECORD ID**.
2. Edit the **Derive** node.
3. Under **Derive** field, type **MA3**.
4. Beside **Derive as**, select **Conditional** from the drop-down list.
5. Under **If**, enter **ACCTNO = @OFFSET(ACCTNO, 2)**. If you use the **Expression Builder**, locate the **@OFFSET** function in the **Sequence** function group.
6. Under **Then**, enter **@MEAN(BALANCE, 3)**. If you use the **Expression Builder**, locate the **@MEAN** function in the **Sequence** function group.
7. Under **Else**, type **undef**.

   The results appear as follows:



Derive field: MA3

Derive as: Conditional

Field type: <Default>

If:
```
ACCTNO = @OFFSET(ACCTNO, 2)
```

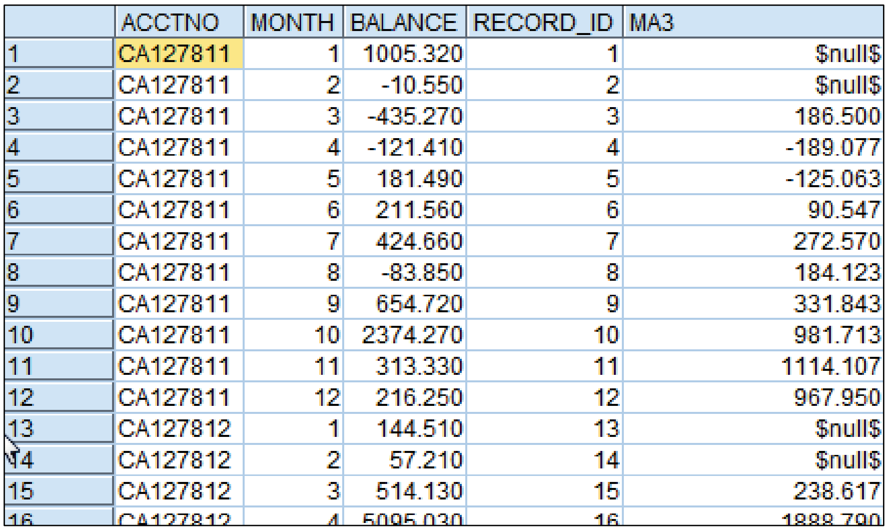
Then:
```
@MEAN(BALANCE, 3)
```

Else:
```
undef
```

8. Close the **Derive** dialog box.

9. From the **Output** palette, add a **Table** node downstream from the **Derive** node, and then run the **Table** node.

   The results appear as follows:



	ACCTNO	MONTH	BALANCE	RECORD_ID	MA3
1	CA127811	1	1005.320	1	$null$
2	CA127811	2	-10.550	2	$null$
3	CA127811	3	-435.270	3	186.500
4	CA127811	4	-121.410	4	-189.077
5	CA127811	5	181.490	5	-125.063
6	CA127811	6	211.560	6	90.547
7	CA127811	7	424.660	7	272.570
8	CA127811	8	-83.850	8	184.123
9	CA127811	9	654.720	9	331.843
10	CA127811	10	2374.270	10	981.713
11	CA127811	11	313.330	11	1114.107
12	CA127811	12	216.250	12	967.950
13	CA127812	1	144.510	13	$null$
14	CA127812	2	57.210	14	$null$
15	CA127812	3	514.130	15	238.617
16	CA127812	4	5095.030	16	1888.790

   For each customer, the first two months are undefined ($null$,) and the records for month 3 through 12 store the moving average computed over the previous two months and the current month.

10. Close the **Table** output window.

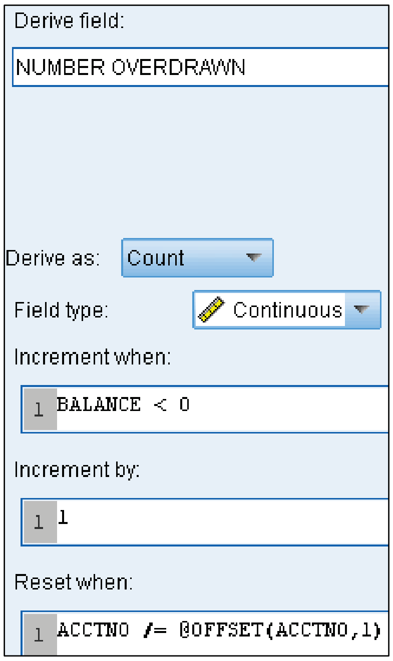
Task 4. Count the number of negative balances.

You will derive a field that counts the number of times to date that the account has been overdrawn (balance falls below 0). The count will be reset when the first entry of a new account is read.

1. From the **Field Ops** palette, add a **Derive** node downstream from the **Derive** node named **MA3**.
2. Edit the **Derive** node.
3. Under **Derive field**, enter **NUMBER OVERDRAWN**.
4. Beside **Derive as**, select **Count** from the drop-down list.
5. Beside **Initial value**, ensure that the value is **0**.
6. Under **Increment when**, enter **BALANCE < 0**.
7. Under **Increment by**, ensure that the value is **1**.

8. Under **Reset when**, enter **ACCTNO /= @OFFSET(ACCTNO,1)**.

   The results appear as follows:



Derive field: NUMBER OVERDRAWN

Derive as: Count

Field type: Continuous

Increment when:

```
BALANCE < 0
```

Increment by:

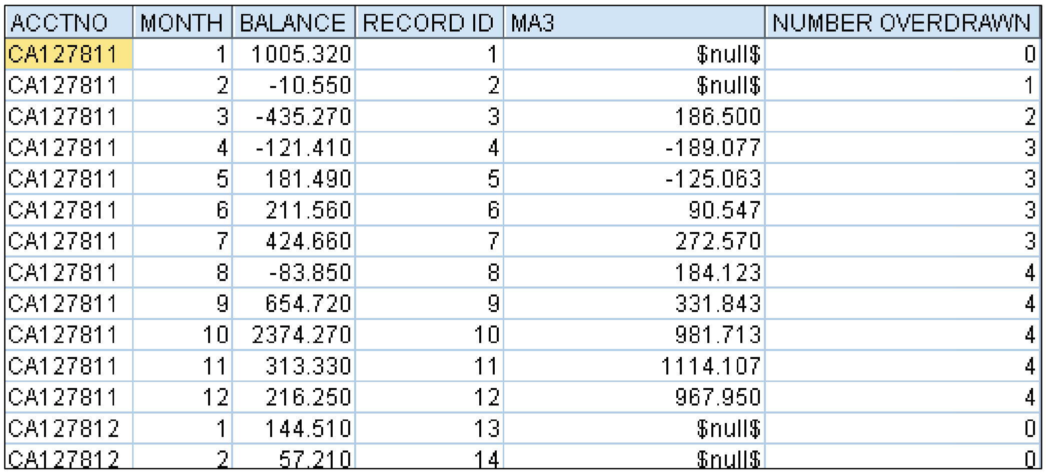
```
1
```

Reset when:

```
ACCTNO /= @OFFSET(ACCTNO,1)
```

9. Close the **Derive** dialog box.
10. From the **Output** palette, add a **Table** node downstream from the **Derive** node, and then run the **Table** node.

    The results appear as follows:



ACCTNO	MONTH	BALANCE	RECORD ID	MA3	NUMBER OVERDRAWN
CA127811	1	1005.320	1	$null$	0
CA127811	2	-10.550	2	$null$	1
CA127811	3	-435.270	3	186.500	2
CA127811	4	-121.410	4	-189.077	3
CA127811	5	181.490	5	-125.063	3
CA127811	6	211.560	6	90.547	3
CA127811	7	424.660	7	272.570	3
CA127811	8	-83.850	8	184.123	4
CA127811	9	654.720	9	331.843	4
CA127811	10	2374.270	10	981.713	4
CA127811	11	313.330	11	1114.107	4
CA127811	12	216.250	12	967.950	4
CA127812	1	144.510	13	$null$	0
CA127812	2	57.210	14	$null$	0

The first customer has overdrawn his/her account 4 times. The second account starts with 0 again.

It would be interesting to examine what the mean is for the new field. To answer that question, however, you need a data structure with one record per account and the number of times the account was overdrawn. Thus, you would need an Aggregate dowstream, aggregating on ACCTNO and saving the maximum value of NUMBER OVERDRAWN per account.

11. Close the **Table** output window.

Task 5. Restructure data.

1. From the **Sources** palette, add a **Var. File** node to the stream canvas, edit the node, and import data from **bank_services.csv**, located in the **C:\Training\0A058** folder.
2. Click **Preview**.

   The data is comprised of services purchased by a bank's customers. Notice that a customer can have purchased the same service multiple times. Also, notice that services are encoded.
3. Close the **Preview** output window.

   Required is a dataset with one record per customer, with the sum, mean, and maximum of SERVICE PRICE. In the *Introduction to IBM SPSS Modeler and Data Science* course, the Transpose node was used for this transformation, but then only one statistic, the sum, was requested.

   You will explore the Transpose dialog box, to identify the difference with using the Restructure node shortly.
4. From the **Field Ops**, add a **Transpose** node to the **Var. File** node.
5. Edit the **Transpose** node.

   The Transpose node supports three ways of restructuring data. To transform records into fields, required in this example, choose the Records to fields mode.
6. Beside **Transpose method**, click **Records to fields**.

   In the lower left area, there is only one statistic at a time that you can select. In this example, sum, mean, and max SERVICE PRICE have to be computed, per service per customer. Thus, this would require three separate Transpose nodes, plus a Merge node downstream to combine the three datasets. And if more statistics were required, this would bring additional Transpose nodes.
7. Click **Cancel** to close the **Transpose** dialog box.

   As a faster alternative, use the Restructure node.

   The Restructure node requires that the field that serves as index for the restructure is categorical. At this point, however, SERVICE CODE is continuous because its values are numeric.
8. From the **Field Ops** palette, add a **Type** node downstream from the **Var. File** named **bank_services.csv**.
9. Edit the **Type** node.
10. Click the cell in the **SERVICE CODE** row, **Measurement** column, and then click **Categorical** from the list.
11. Click **Read Values**.

    SERVICE CODE is instantiated as a nominal field, with values 1, 2, 3, 4, and 5.

12. Close the **Type** dialog box.
13. From the **Field Ops** palette, add a **Restructure** node downstream from the **Type** node.
14. Edit the **Restructure** node.
15. Click the **Settings** tab, if necessary.
16. Under Available fields, select the field that serves as index for the restructure operation. Only categorical (flag, nominal, ordinal) fields are eligible. If you need a continuous field as index field, you need to change the field's measurement level to nominal in an upstream Type node, as you did in step 10 of this demonstration.
17. Under **Available fields**, select **SERVICE CODE**.

    The area under Available values will be populated with the categories of the selected categorical field, provided that the field is instantiated.

    Move the values for which you want to create fields to the Create restructured fields area.
18. Under **Available values**, click **1**, and then press **Ctrl+A** to select all values.
19. Click the **Add selected fields** button to add them into the **Create restructured fields** box.

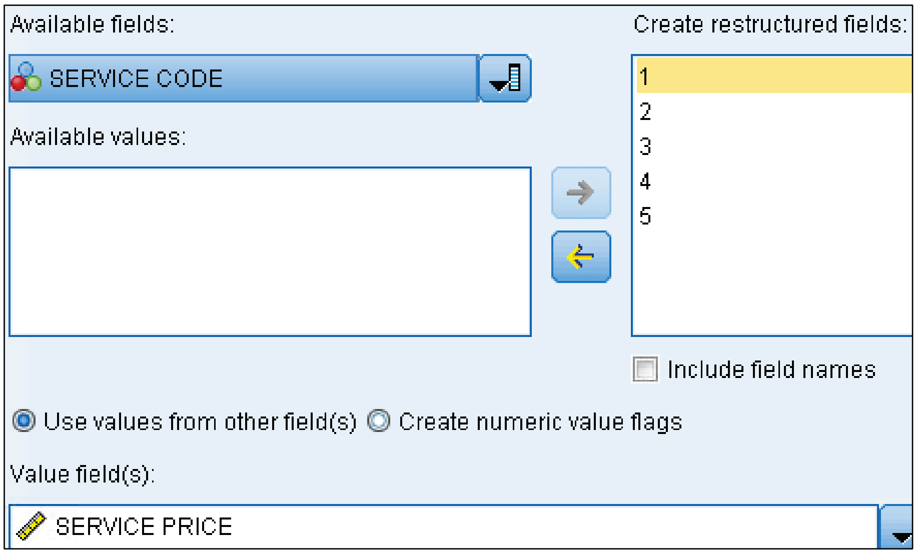
    When the Include field names option is enabled, the name of the index field will be used as prefix for the new fields. You will not use that option here.
20. Disable the **Include field names** option.

    The Restructure dialog box offers two modes. The Create numeric flags option will create flag fields, with values 0 and 1. This feature mimics the SetToFlag node, but the SetToFlag node offers more flexibility, such as an aggregate in the node itself. Thus, you probably will prefer the SetToFlag node to create flag fields for the categories. The Use values from other field(s) option is appropriate when a continuous field must be expanded into a series of new fields indexed by the values of a categorical field.

    In this case, select the continuous field that must be expanded into a series of new continuous fields, SERVICE PRICE.

21. Under **Value field(s)**, select **SERVICE PRICE**.

    The results appear as follows:



Available fields:

SERVICE CODE

Available values:

Create restructured fields:

1
2
3
4
5

Include field names

[✓ Use values from other field(s)] Create numeric value flags

Value field(s):

SERVICE PRICE

22. Click **Preview**.

    The fields storing the price per service have been created, but there are still as many records per customer as services that were purchased. Aggregating cannot be done in the Restructure node (in this respect it differs from the SetToFlag node), so you will need a separate Aggregate node downstream from the Restructure node.
23. Close the **Preview** output window.
24. Close the **Restructure** dialog box.
25. From the **Record Ops** palette, add an **Aggregate** node downstream from the **Restructure** node.
26. Edit the **Aggregate** node.
27. For **Key fields**, select **ACCTNO**.
28. Under **Basic Aggregates**, **Aggregate fields**, select **1_SERVICE PRICE** to **5_SERVICE PRICE**.
29. Beside **Default mode**, enable the **Sum**, **Mean**, and **Max** options.
30. Click the **Apply default operations to all fields** button.
31. Click **Preview**.

    There is one record per account, with 5 * 3 fields storing the sum, mean, and max of SERVICE PRICE, for each product, as required.
32. Close the **Preview** output window.

33. Close the **Aggregate** dialog box.

    In summary, when only one statistic is required, the Transpose node is the most efficient option. However, when multiple statistics are required, use the Restructure in conjunction with the Aggregate node.

    As a note, customers in this dataset have purchased the same service multiple times. If you are sure that a particular service is purchased at most once by a customer, statistics such as mean, sun, min, and max will all return the same result, and Transpose will be the most efficient option.

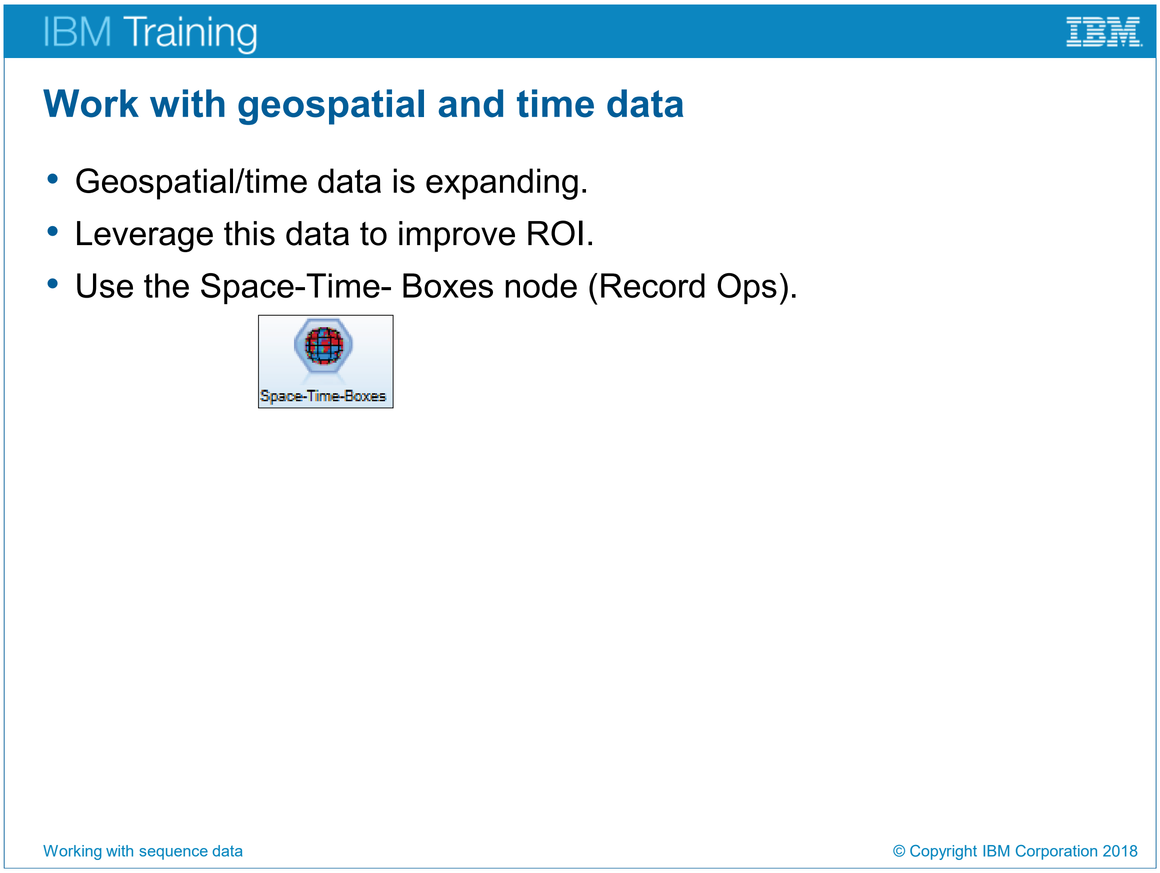
    This completes the first demonstration. You will create a clean state for the next demonstration.

34. From the **File** menu, click **Close Stream**. Click **No** when asked to save the stream.
35. From the **File** menu, click **New Stream**.

    Leave IBM SPSS Modeler open for the next demonstration.

Results:
You have applied sequence functions to derive new fields and you have restructured the data so that you can perform the required analyses.

You will find the completed stream in the **C:\Training\0A058\Solutions** folder.



IBM Training

Work with geospatial and time data

- Geospatial/time data is expanding.
- Leverage this data to improve ROI.
- Use the Space-Time- Boxes node (Record Ops).

Space-Time-Boxes

Working with sequence data

© Copyright IBM Corporation 2018

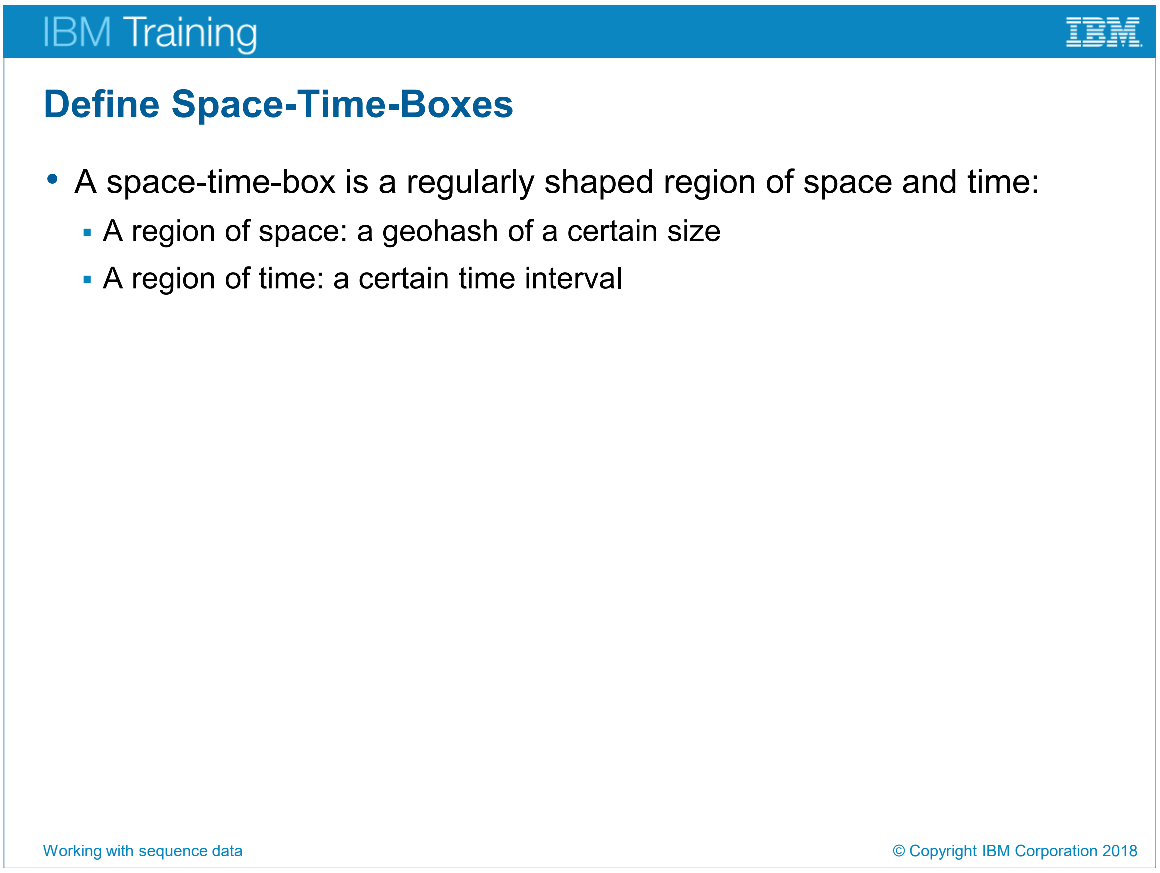
Work with geospatial and time data

Today, data is everywhere, and the analysis of big data is critical to a company's success.

As an example, it is estimated that mobile devices generate about 600 billion transactions a day. Every call, text message, e-mail and data transfer creates a data point with space/time coordinates. You could use this space and time information to target this person in location-based promotions.

To leverage this information, it is important to know how geospatial and time data can be analyzed. IBM SPSS Modeler's Space-Time-Boxes node is designed for this.

Note: There are a growing number of privacy laws related to location data emerging around the world. Thinking about the legal and policy aspects in such projects is considered a best practice. Privacy by Design (PbD) is an approach you might consider.



IBM Training

Define Space-Time-Boxes

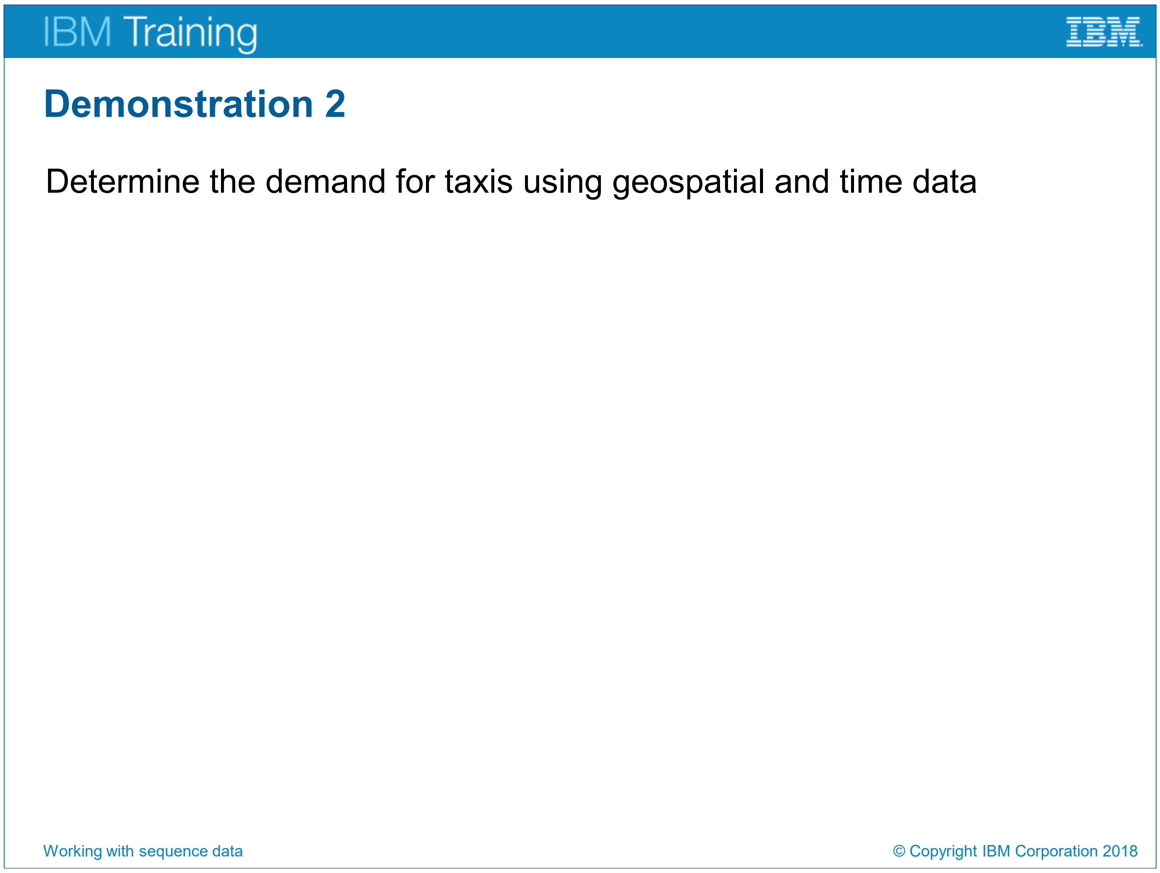
- A space-time-box is a regularly shaped region of space and time:
  - A region of space: a geohash of a certain size
  - A region of time: a certain time interval

Working with sequence data

Define Space-Time-Boxes

Space-time-boxes are used to describe where an entity is, at what point in time, by combining a spatial grid with a time interval.The spatial grid is defined by so-called geohashes. Geohashes define an area of a certain size and have a hierarchical structure. On the highest level earth is divided into 32 boxes, each of them is subdivided into 32 smaller boxes, and so forth. For example, geohash u is one of the 32 areas at the highest level and covers a part of Europe. Geohash u0 is one of the 32 boxes within geohash u and covers a great part of France, and geohash u09 covers Paris. At the deepest level, staying in Paris, u09tunqbwevy is the entrance of the Eiffel Tower in the south pillar. Thus, the longer the geohash is, the smaller the size of the area and the more precise the location.

In the same way, time can be thought of as being hierarchically divided into years, months, days, and so forth, down to the deepest level of seconds.A space-time-box encodes a location (given by latitude and longitude) and a timestamp into a geohash appended with the start and the end of the time interval. For example, coordinates [48.85767317994797, 02.2947075963020324] (the entrance of the Eiffel Tower in the south pillar), timestamp 2014-APR-02 12:23:58 is encoded into u09tunq|2014-04-02 12:00:00|2014-04-02 13:00:00 (using a density of approximately 76 meters for the geohash, and a one hour time interval).



IBM Training

Demonstration 2

Determine the demand for taxis using geospatial and time data

Working with sequence data

© Copyright IBM Corporation 2018

Demonstration 2: Determine the demand for taxis using geospatial and time data

Demonstration 2: Determine the demand for taxis using geospatial and time data

Purpose:
You will use the Space-Time-Boxes node to determine the demand for taxis, using data from customers who opted in on a cell phone app.

Data file: **taxi cab customers app data.csv**

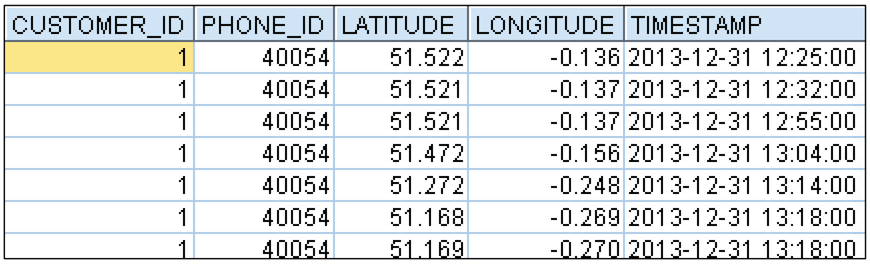
Data folder: **C:\Training\0A058**

Task 1. Analyze geospatial and time data, using the Individual Records mode.

A taxi company needs to know the demand on New Year's Eve. The company has geospatial and time data for phones that have opted into their smart-taxi cell phone app.

1. From the **Sources** palette, add a **Var. File** node to the stream canvas, edit the **Var. File** node, and then import data from **taxi cab customers app data.csv**, located in the **C:\Training\0A058** folder.
2. From the **Output** palette, add a **Table** node downstream from the **Var. File** node.
3. Run the **Table** node.

   The results appear similar to the following:



CUSTOMER_ID	PHONE_ID	LATITUDE	LONGITUDE	TIMESTAMP
1	40054	51.522	-0.136	2013-12-31 12:25:00
1	40054	51.521	-0.137	2013-12-31 12:32:00
1	40054	51.521	-0.137	2013-12-31 12:55:00
1	40054	51.472	-0.156	2013-12-31 13:04:00
1	40054	51.272	-0.248	2013-12-31 13:14:00
1	40054	51.168	-0.269	2013-12-31 13:18:00
1	40054	51.169	-0.270	2013-12-31 13:18:00

   Each record is a geospatial ping from the app. A customer is identified by an ID, linked to his PHONE_ID (thus, a customer has only one phone id).

   To determine the demand you will use the Space-Time-Boxes node to get the number of customers within a specific space and time density, for example within an area of 2.4 kilometers and a one hour time interval.

4. Close the **Table** output window.
5. From the **Record Ops** palette, add a **Space-Time-Boxes** node downstream from the **Var. File** node.

6. Edit the **Space-Time-Boxes** node.
7. Click the **Settings** tab, if necessary.

   The Space-Time-Boxes dialog box has two modes.

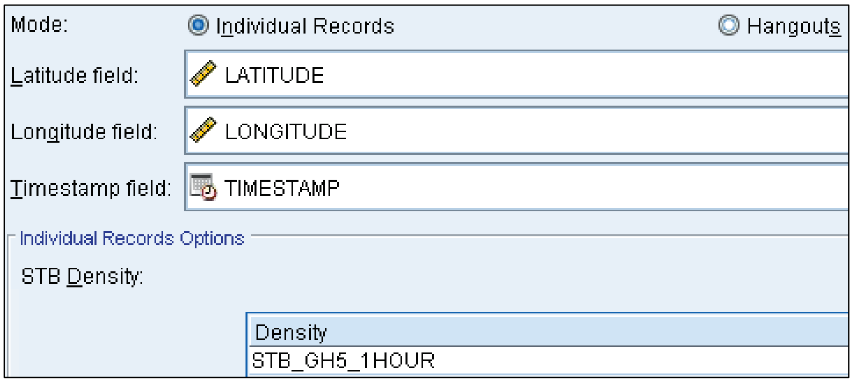
   - Individual Records. Based on the record's spatial coordinates and timestamp, a field is created with the geohash that encloses the location, and the start time and end time of the timestamp field. Typically, this mode is used in conjunction with the Aggregate node to determine how many entities exist at specific densities.
   - Hangouts. A hangout can be thought of as a location and/or time in which an entity is continually or repeatedly found. This option will aggregate the data, with the combination of entity and space-time-boxes defining the records. (This is the reason why this node is located in the Record Ops palette.)

   Both modes require the same basic specifications. The latitude and longitude field define the coordinates, the timestamp field defines the date and/or time.

   You will explore the Individual Records option in this task.
8. Beside **Latitude field**, select **LATITUDE**.
9. Beside **Longitude field**, select **LONGITUDE**.
10. Beside **Timestamp field**, select **TIMESTAMP**.

    You will set the size for the geo density and the time interval.
11. In the **Individual Records Options** section, click the **Add density** button, and then for **Geo density** select **GH5 (approx. 2.4 kilometers)**, for **Time interval** select **1 Hour**.
12. Close the **Define Space-Time-Box Density** sub dialog box.

    The results appear as follows:



Mode: [✓ Individual Records] Hangouts

Latitude field: LATITUDE

Longitude field: LONGITUDE

Timestamp field: TIMESTAMP

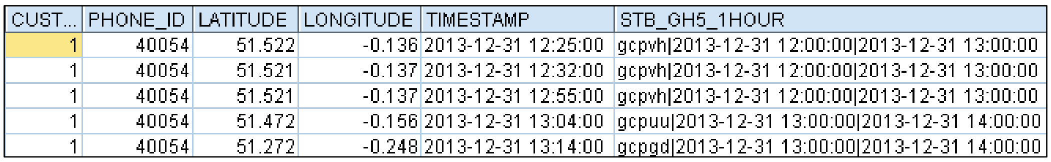
Individual Records Options

STB Density:

Density
STB_GH5_1HOUR

13. Click **Preview**.

    The results appear similar to the following:



CUST...	PHONE_ID	LATITUDE	LONGITUDE	TIMESTAMP	STB_GH5_1HOUR
1	40054	51.522	-0.136	2013-12-31 12:25:00	gcpvh|2013-12-31 12:00:00|2013-12-31 13:00:00
1	40054	51.521	-0.137	2013-12-31 12:32:00	gcpvh|2013-12-31 12:00:00|2013-12-31 13:00:00
1	40054	51.521	-0.137	2013-12-31 12:55:00	gcpvh|2013-12-31 12:00:00|2013-12-31 13:00:00
1	40054	51.472	-0.156	2013-12-31 13:04:00	gcpuu|2013-12-31 13:00:00|2013-12-31 14:00:00
1	40054	51.272	-0.248	2013-12-31 13:14:00	gcpgd|2013-12-31 13:00:00|2013-12-31 14:00:00

    The first three records, all from the same customer, are in the same space-time-box (geohash gcpvh, between 12:00h and 13:00h). The fourth record tells you that this customer is in another area (as defined by the size of the geo density) in the next hour.

14. Close the **Preview** output window.
15. Close the **Space-Time-Boxes** dialog box.

    You will aggregate the data to know the number of persons in a certain geo density, in a particular time interval. However, when you aggregate the data at this point using **STB_GH5_1HOUR** as the key field, you will get the number of pings in each space-time-box, not the number of persons. For example, the first customer has three records in the same space-time-box because he had three pings in that particular space-time-box, but you want this customer only to count once when you assess the demand.

    Therefore, you will first aggregate on both CUSTOMER_ID and STB_GH5_1HOUR. In this aggregate, you will also save the number of pings. That is not necessary to answer the question, but it will clarify the difference between customers and pings.

16. From the **Record Ops** palette, add an **Aggregate** node downstream from the **Space-Time-Boxes** node.
17. Edit the **Aggregate** node.
18. For **Key fields**, select **CUSTOMER_ID** and **STB_GH5_1HOUR**.
19. Ensure that the **Include record count in field** option is enabled, and type **NUMBER OF PINGS PER CUSTOMER IN STB**.
20. Click **Preview**.

    The first customer has three pings in geohash gcpvh, between 2013-12-31 12:00:00 and 2013-12-31 13:00:00, in agreement with previous findings.

    To obtain the number of persons in each space-time-box, aggregate the data on **STB_GH5_1HOUR** and count the number of records (customers).

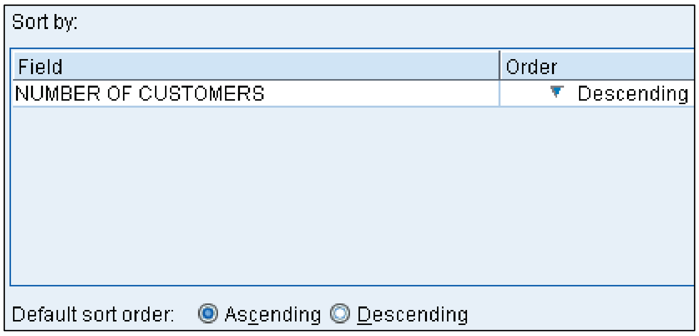
21. Close the **Preview** output window.
22. Close the **Aggregate** dialog box.
23. From the **Record Ops** palette, add a **second Aggregate** node downstream from the **first Aggregate** node.
24. Edit the **Aggregate** node.

25. For **Key fields**, select **STB_GH5_1HOUR**.
26. Ensure that the **Include record count in field** option is enabled, and type **NUMBER OF CUSTOMERS**.
27. Click **Preview**.

    There were 23 persons in geohash gcpvh, between 2013-12-31 12:00:00 and 2013-12-31 13:00:00.

    You will sort the data descending on NUMBER OF CUSTOMERS to know in which area and time interval the demand was highest.
28. Close the **Preview** output window.
29. Close the **Aggregate** dialog box.
30. From the **Record Ops** palette, add a **Sort** node downstream from the **Aggregate** node.
31. Edit the **Sort** node.
32. In the **Sort by** section, select **NUMBER OF CUSTOMERS**, and then, in the **Order** column, click **Ascending**, so that it changes to **Descending** (beside Default sort order, leave the Ascending option as it is).

    The results appear as follows:



Sort by:

Field	Order
NUMBER OF CUSTOMERS	Descending

Default sort order: [✓ Ascending] Descending

33. Click **Preview**.

    The demand was highest in geohash gcpvj, between 2013-12-31 09:00:00|2013-12-31 10:00:00.

    Knowing this, you could align the number of taxi cabs with the demand. Or, you could plot the demand for a specific time frame, using mapping software. In the latter case, you can use IBM SPSS Modeler's tb_centroid_latitude and stb_centroid_longitude function to create a field that gives the center of the geohash, so that you can plot the frequencies on the center of the geohash areas. The stb_centroid_latitude and stb_centroid_longitude function are available under General Functions in the Expression Builder.

    Note: Another useful function is to_geohash, which returns the geohashed string corresponding to a specified latitude, longitude and density.
34. Close the **Preview** output window.
35. Close the **Sort** dialog box.

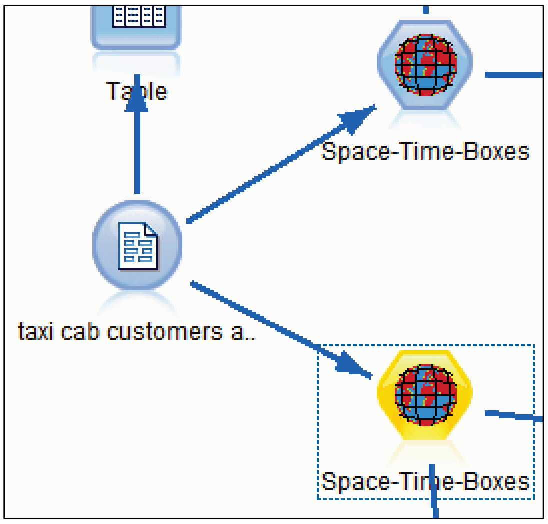
Task 2. Analyze geospatial and time data, using the Hangouts mode.

In the previous task, you have determined the demand by counting the number of customers in each space-time-box. A more sophisticated way to determine the demand is to use the Hangouts mode in the Space-Time-Boxes node. For example, if one pings at 12:55h in geohash gcpvj and another ping from the same person is at 13:10h in the same geohash, you cannot conclude from the previous analysis that this person hangs out in geohash gcpvj because of the two different time windows. By allowing space-time-boxes to overlap, as the Hangouts mode does, IBM SPSS Modeler will qualify these events as a hangout (in this example: for geohash gcpvj, in the 13:00h - 14:00h time interval).

To demonstrate the Hangouts mode, you will use the same dataset to get the number of people hanging out in each of the space-time-boxes.

1. From the **Record Ops** palette, add a **Space-Time-Boxes** node downstream from the **Var. File** node.

   The results appear similar to the following:

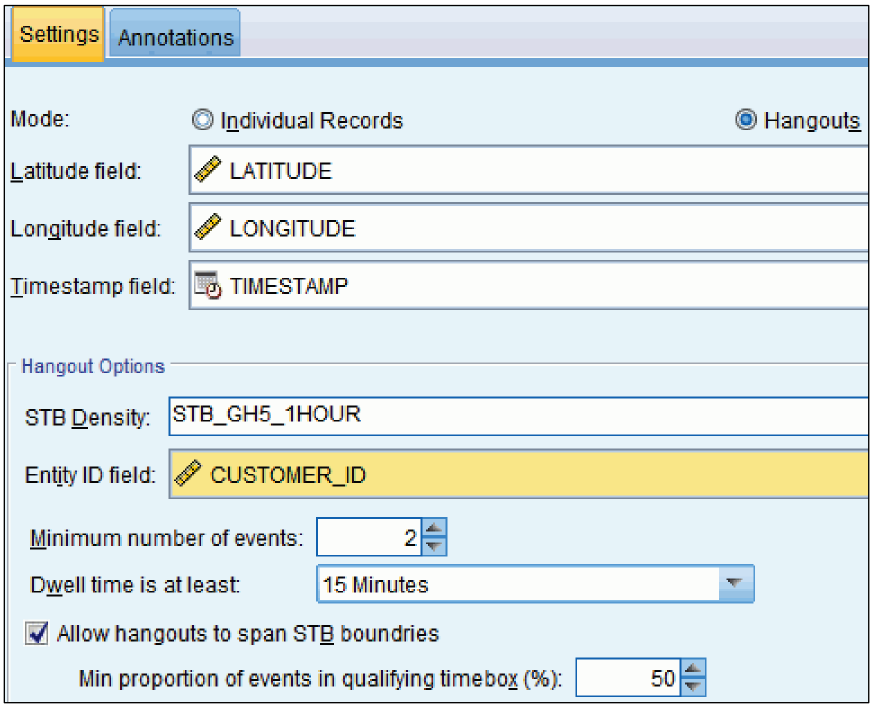


2. Edit the **Space-Time-Boxes** node.
3. Click the **Settings** tab, if necessary.
4. Beside **Mode**, enable the **Hangouts** option.
5. Beside **Latitude field**, select **LATITUDE**.
6. Beside **Longitude field**, select **LONGITUDE**.
7. Beside **Timestamp** field, select **TIMESTAMP**.
8. Beside **STB Density**, select **GH5 (approx 2.4 kilometers)**, and a **1 Hour** time interval.

9. Beside **Entity ID** field, select **CUSTOMER_ID**.

   When the Hangouts option is selected, the minimum number of events must be specified for the entity to be qualified as hanging out in a space-time-box. Also, the minimum dwell time (the minimum time that the entity stays at the same place) must be specified.
10. Beside **Minimum number of events**, ensure that the value is **2** (a customer should have at least 2 records in a certain space-time-box to be qualified as a hangout for that space-time-box).
11. Beside **Dwell time is at least**, ensure that the value is **15 Minutes** (a customer should at least be 15 minutes in a certain space-time-box to be qualified as a hangout for that space-time-box).
12. Ensure that the **Allow hangouts to span STB boundaries** option is enabled.
13. Ensure that the **Min proportion of events in qualifying timebox (%)** value is **50** (at least 50% of the events should be in the specific time frame; for example, suppose that a customer has events in the same geohash at 12:50h, 12:55h, 12:59h, 13:04h and 13:10h, then this will not qualify as a hangout for 13:00h - 14:00h because only two out of the five = 40% of the events is in the qualifying time box).

    The results appear as follows:

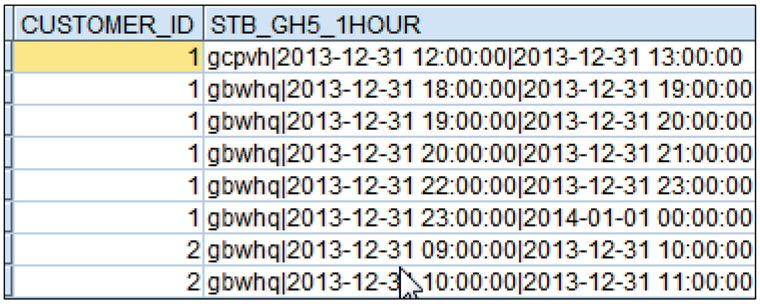


Mode: Individual Records [✓ Hangouts]
Latitude field: LATITUDE
Longitude field: LONGITUDE
Timestamp field: TIMESTAMP

Hangout Options
STB Density: STB_GH5_1HOUR
Entity ID field: CUSTOMER_ID
Minimum number of events: 2
Dwell time is at least: 15 Minutes
✓ Allow hangouts to span STB boundries
Min proportion of events in qualifying timebox (%): 50

14. Click **Preview**.

    The results appear as follows:



CUSTOMER_ID	STB_GH5_1HOUR
1	gcpvh|2013-12-31 12:00:00|2013-12-31 13:00:00
1	gbwhq|2013-12-31 18:00:00|2013-12-31 19:00:00
1	gbwhq|2013-12-31 19:00:00|2013-12-31 20:00:00
1	gbwhq|2013-12-31 20:00:00|2013-12-31 21:00:00
1	gbwhq|2013-12-31 22:00:00|2013-12-31 23:00:00
1	gbwhq|2013-12-31 23:00:00|2014-01-01 00:00:00
2	gbwhq|2013-12-31 09:00:00|2013-12-31 10:00:00
2	gbwhq|2013-12-3 10:00:00|2013-12-31 11:00:00

    The first customer especially hung out in geohash gbwhq in the evening hours.

15. Close the **Preview** output window, and then close the **Space-Time-Boxes** dialog box.

    You will aggregate the records on the space-time-box field, to know how many people were hanging out in a specific space-time-box.

16. From the **Record Ops** palette, add an **Aggregate** node downstream from the **Space-Time-Boxes** node.
17. Edit the **Aggregate** node.
18. For **Key fields**, select **STB_GH5_1HOUR**.
19. In the **Include record count in field** text box, type **NUMBER OF CUSTOMERS**.
20. Close the **Aggregate** dialog box.

    To know in which area and time interval the demand is highest, you will sort the data descending on NUMBER OF CUSTOMERS.

21. From the **Record Ops** palette, add a **Sort** node downstream from the **Aggregate** node.
22. Edit the **Sort** node.
23. Select **NUMBER OF CUSTOMERS**.
24. Set **Order** to **Descending** (in the **Order** column).
25. Click **Preview**.

    The busiest location and time is gcpvj, from 2013-12-31 23:00:00 to 2014-01-01 00:00:00. This was the third busiest space-time-box when individual records were examined.

26. Close the **Preview** output window.
27. Close the **Sort** dialog box.

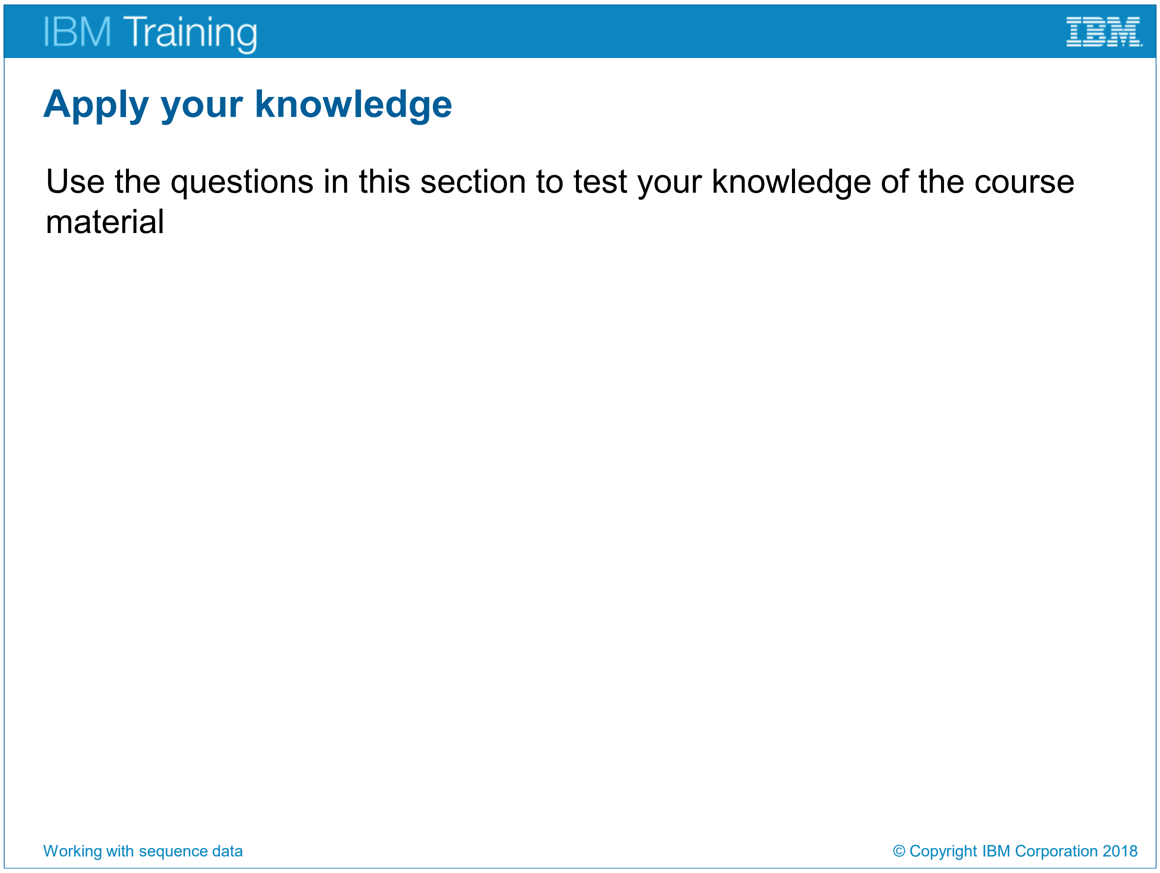
    This completes the second demonstration. You will create a clean state for the exercises.

28. From the **File** menu, click **Close Stream**, and click **No** when asked to save changes.
29. From the **File** menu, click **New Stream**.

    Leave IBM SPSS Modeler open for the exercises.

Results:
You have used the Space-Time-Boxes node to determine the most popular place and time where entities are located or hang out.

You will find the completed stream in the **C:\Training\0A058\Solutions** folder.



IBM Training

Apply your knowledge

Use the questions in this section to test your knowledge of the course material

Working with sequence data

© Copyright IBM Corporation 2018

Apply your knowledge

Questions

Question 1: Refer to the table that follows. Which value is computed for the first two records when you use the @MEAN (X, 3) function ?

A. Record 1 and record 2 both are assigned the undefined ($null$) value.

B. Record 1 is assigned the value 1; record 2 is assigned the value 2.

C. Record 1 is assigned the value 1; record 2 is assigned the value 1.5.

D. None of the above statements are correct.

X	**@MEAN (X, 3)**
1	**?**
2	**?**
3	2
4	3
5	4

Question 2: Refer to the table below. What was the condition used for the true value, for the derived flag field Z?

A. CUSTOMER_ID = @OFFSET(CUSTOMER_ID, 1)

B. CUSTOMER_ID = @OFFSET(CUSTOMER_ID, -1)

C. CUSTOMER_ID = @INDEX (CUSTOMER_ID, 1)

D. PRODUCT = @OFFSET(PRODUCT, 1)

CUSTOMER_ID	**PRODUCT**	**Z**
1	A	F
2	A	F
2	C	T
3	B	F
3	C	T
3	F	T

Question 3: Refer to the dataset and Derive dialog box below. True or false: Record #3 (RECORD_ID = 3) has value 2 for Z.

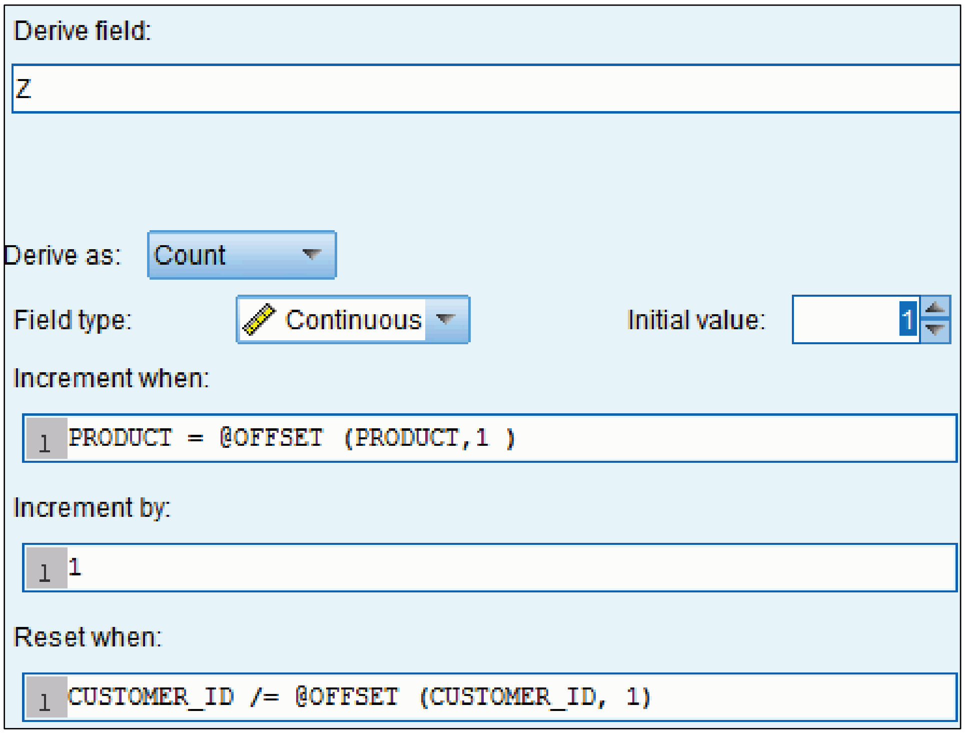
A. True

B. False

Data

RECORD_ID	CUSTOMER_ID	PRODUCT
1	1	A
2	2	B
3	2	B
4	3	C
5	3	D
6	3	E

Derive dialog box for Z



Derive field: Z

Derive as: Count

Field type: Continuous

Initial value: 1

Increment when:

```
PRODUCT = @OFFSET (PRODUCT,1 )
```

Increment by:

```
1
```

Reset when:

```
CUSTOMER_ID /= @OFFSET (CUSTOMER_ID, 1)
```

Question 4: True or false: The Restructure node is located in the Record Ops palette.

A. True

B. False

Question 5: True or false: The Restructure node has the option to aggregate the data using a key field.

A. True

B. False

Question 6: True or false: A geohash relates to a time zone.

A. True

B. False

Question 7: True or false: The area defined by geohash wx4fbutzzbr is part of the area defined by geohash wx4fbutz.

A. True

B. False

Question 8: True or false: When you use the Hangouts mode in the Space-Time-Boxes node and you use a time interval of one day, you will get more records output from the Space-Time-Boxes node than when you use a time interval of 1 hour (all other settings equal).

A. True

B. False

Answers to questions

Answer 1: C. @MEAN (X, 3) computes the mean over the previous two records and the current record. Thus, the result for the first record is the value itself, 1. The result for the second record is the mean computed over records 1 and 2, which equals 1.5.

Answer 2: A. The derived field is true when CUSTOMER_ID equals CUSTOMER_ID of the previous record, else it is F. This is what the expression CUSTOMER_ID = @OFFSET(CUSTOMER_ID, 1) flags.

Answer 3: A. True. The field Z initializes at 1 for each CUSTOMER_ID and increments by 1 when the customer's product is the same as the product of the previous record. The third record represents the same customer as the second record and the product is the same (product B), so the outcome equals 2.

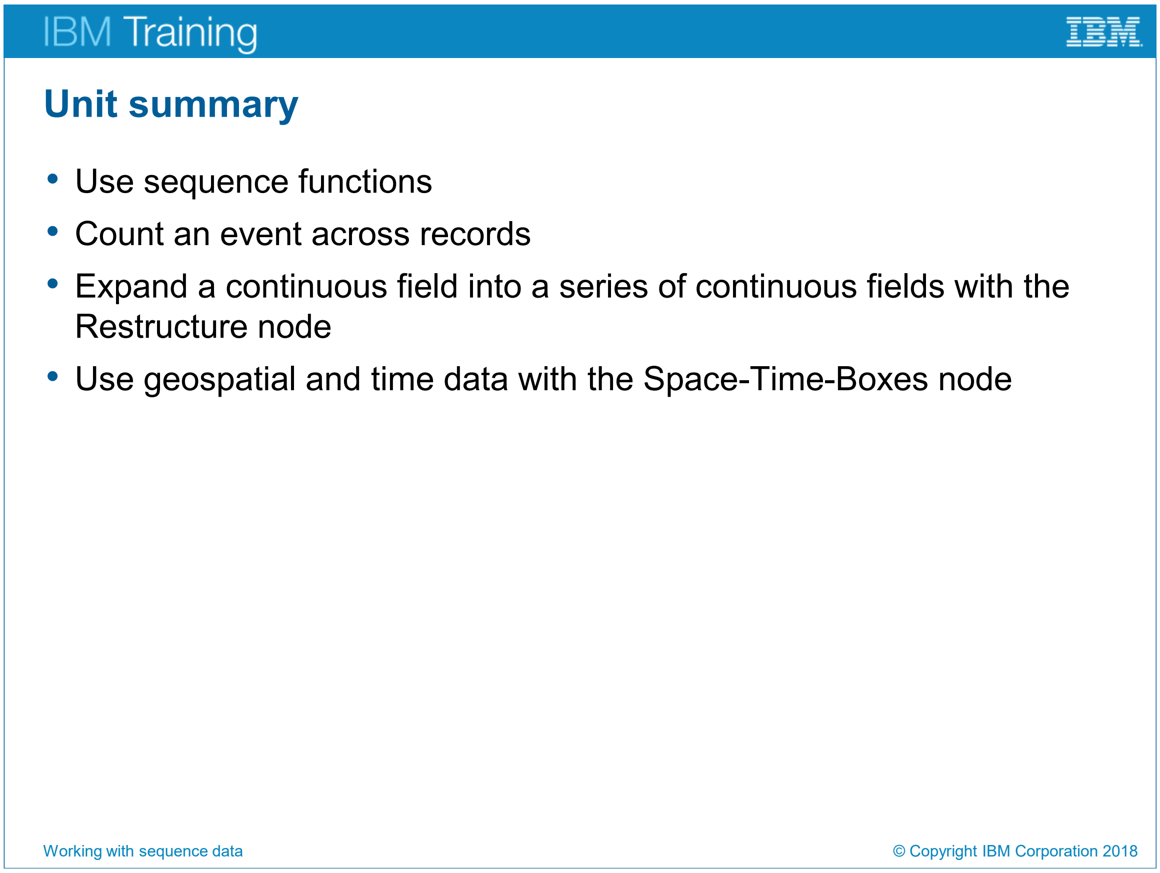
Answer 4: B. False. The Restructure node is located in the Field Ops palette, not in the Record Ops palette.

Answer 5: B. False. The Restructure node does not have an option to aggregate the data. Use the Aggregate node downstream from the Restructure node to aggregate the data.

Answer 6: B. False. A geohash relates to a spatial area.

Answer 7: A. True. Geohashes are hierarchical, so the area defined by geohash wx4fbutz encloses the area defined by geohash wx4fbutzzbr.

Answer 8: B. False. A time interval of one day will return fewer records than a time-interval of one hour.



IBM Training

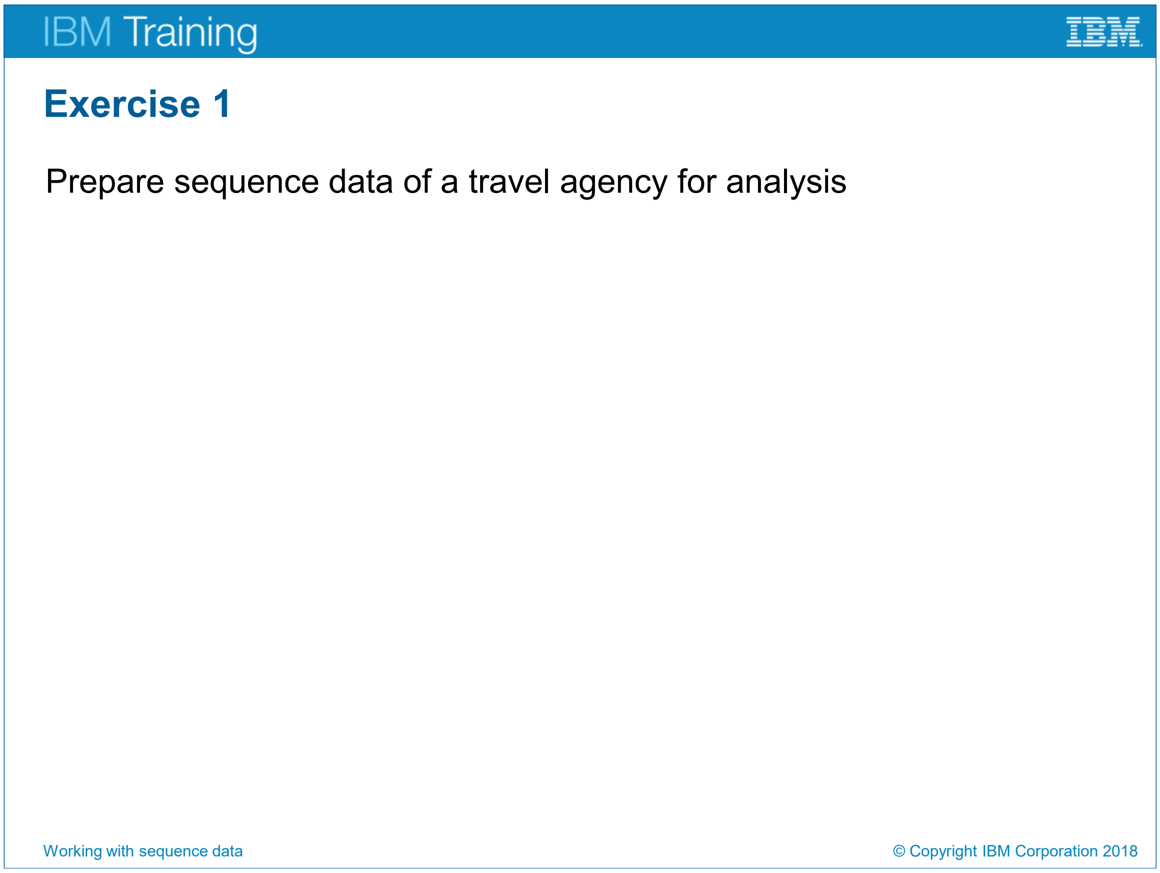
Unit summary

- Use sequence functions
- Count an event across records
- Expand a continuous field into a series of continuous fields with the Restructure node
- Use geospatial and time data with the Space-Time-Boxes node

Working with sequence data

© Copyright IBM Corporation 2018

Unit summary



IBM Training

Exercise 1

Prepare sequence data of a travel agency for analysis

Working with sequence data

© Copyright IBM Corporation 2018

Exercise 1: Prepare sequence data of a travel agency for analysis

Exercise 1: Prepare sequence data of a travel agency for analysis

Data file:	**customers_and_holidays.csv**
Data folder:	**C:\Training\0A058**

In this exercise you will work with a dataset that contains information about where people go to for their holiday. You will derive new fields to answer questions such as "What is the mean age of the customers?", "What was the most popular country?" and so forth.

- Use a **Var. File** node to import the data from **customers_and_holidays.csv**, located in the **C:\Training\0A058** folder.
- Derive a field named **RECORD NUMBER** that stores the record number.
- Derive a field named **CUSTOMER NUMBER** running from 1 to the number of customers. Records having the same CUSTID are the same person, so must have the same value for **CUSTOMER NUMBER** (because a customer can have multiple records, this field will differ from the RECORD NUMBER field).

  Hint: Ensure that the records are sorted by **CUSTID**. Then, add a **Derive** node downstream from the **Sort** node, edit the **Derive** node and for **Derive as** select **Count**. Increment when **CUSTID** differs from that of the previous record, and ensure that the field is never reset (do not specify the condition when to reset, or specify a condition that is impossible to satisfy).

  Run a **Statistics** node so that you will know the number of customers.
- You want to examine how money spent on leisure (SPENT_LEISURE) in one country correlates with money spent on leisure in another country. Hereto, SPENT_LEISURE must be expanded in a series of fields, storing the total amount of money spent on leisure in each country.

  Use the Restructure node to create fields that store the total amount of money spent on leisure in AUSTRIA, money spent on leisure in FRANCE; money spent on leisure in GERMANY, and so forth.

Hints:

- Cleanse **COUNTRY** with a **Filler** node, so that the country names are in upper case (use a **Filler** node, and the **lowertoupper** function to accomplish this).
- Add a **Type** node downstream from the **Filler** node, and instantiate the data so that IBM SPSS Modeler knows the values for **COUNTRY**.
- Add a **Restructure** node downstream from the **Type** node to expand **SPENT_LEISURE** by **COUNTRY**.

After running the Restructure procedure, the dataset will still have multiple records per customer.

- Create a dataset that has one record per **CUSTID** and the money that the customer has spent on leisure in each country.

  Note: Having this dataset in place, you could explore the correlations between the fields. This is not pursued in this course.
- Close the stream without saving, and then create a new stream.

Exercise 1: Tasks and Results

Task 1. Import the data.

- Use a **Var. File** node to import the data from **customers_and_holidays.csv**, located in the **C:\Training\0A058** folder.

Task 2. Derive a record identifier.

- From the **Field Ops** palette, add a **Derive** node downstream from the **Var. File** node.
- Edit the **Derive** node:
  - Under **Derive field**, type **RECORD NUMBER**.
  - Under **Formula**, type **@INDEX**.
- Preview the data to check the results.

Task 3. Derive a customer identifier.

- From the **Record Ops** palette, add a **Sort** node downstream from the **Derive** node named **RECORD NUMBER**.
- Edit the **Sort** node, under **Sort by**, select **CUSTID**. Ensure that the sort order is **Ascending**.
- From the **Field Ops** palette, add a **Derive** node downstream from the **Sort** node.
- **Edit the Derive** node:
  - Under **Derive field**, type **CUSTOMER NUMBER**.
  - Beside **Derive as**, select **Count**.
  - Beside **Initial value**, type **1**.
  - Under **Increment when**, enter **CUSTID /= @OFFSET (CUSTID, 1)**.
  - Under **Increment by**, ensure that the value is **1**.
  - Leave the **Reset when** box empty (alternatively, enter a condition that will never be true, such as 1=0).
- From the **Output** palette, add a **Statistics** node downstream from the **Derive** node named **CUSTOMER NUMBER**.

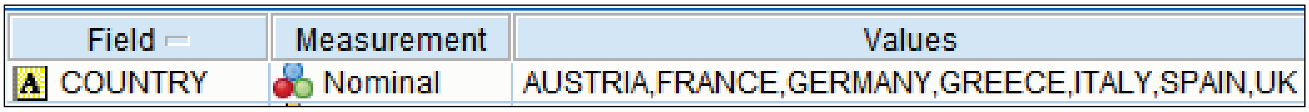
- Edit the **Statistics** node:
  - Beside **Examine**, select **CUSTOMER NUMBER**.
  - Ensure that the **Max** option is enabled, and disable all other statistics.
  - Click **Run**.

  There are 376 customers.

Task 4. Restructure the dataset.

- From the **Field Ops** palette, add a **Filler** node downstream from the **Derive** node named **CUSTOMER NUMBER**.
- Edit the **Filler** node:
  - Under **Fill in fields**, select **COUNTRY**.
  - Beside **Replace**, select **Always**.
  - Under **Replace with**, enter **lowertoupper (@FIELD)**.
- From the **Field Ops** palette, add a **Type** node downstream from the **Filler** node.
- Edit the **Type** node, and then click **Read Values** to instantiate the data (so IBM SPSS Modeler knows the countries).

  The results appear similar to the following:



Field	Measurement	Values
COUNTRY	Nominal	AUSTRIA,FRANCE,GERMANY,GREECE,ITALY,SPAIN,UK

- From the **Field Ops** palette, add a **Restructure** node downstream from the **Type** node.
- Edit the **Restructure** node:
  - Under **Available fields**, select **COUNTRY**.
  - Under **Available values**, press **Ctrl+A** to select all values.
  - Add them to the **Create restructured fields** box.
  - Disable the **Include field names** option (if preferred).
  - Ensure the **Use values from other field(s)** option is selected.
  - Under **Value field(s)**, select **SPENT_LEISURE**.

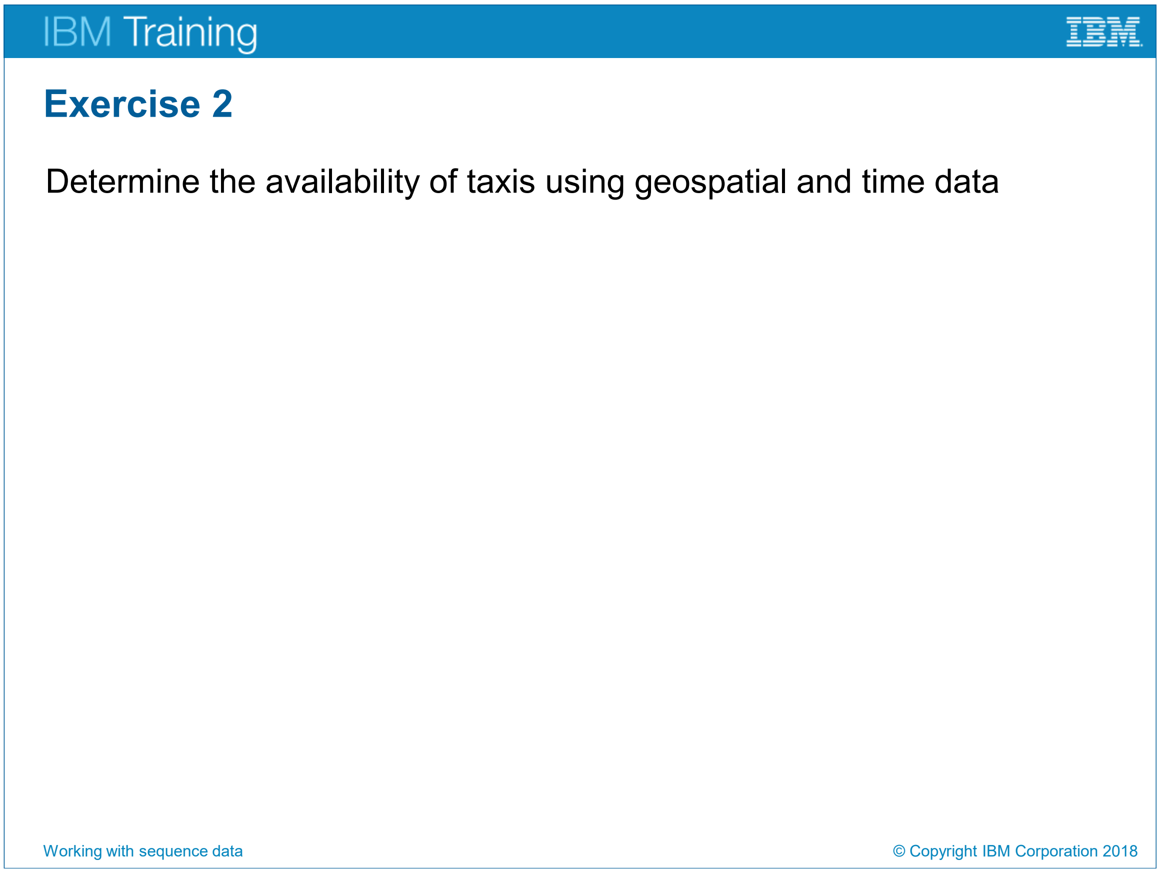
Task 5. Use Aggregate to have one record per customer.

- From the **Record Ops** palette, add an **Aggregate** node downstream from the **Restructure** node.
- Edit the **Aggregate** node:
    - Under **Key fields**, select **CUSTID**.
    - Under **Aggregate** fields, select **AUSTRIA_SPENT_LEISURE** to **UK_SPENT_LEISURE**.
    - Select the **Sum** statistic (disable all other statistics).
    - Disable the **Include record count in field** option.
- Preview the data to check the results.

Task 6. Create a clean state.

- From the **File** menu, click **Close Stream**. Click **No** when asked to save the stream.
- From the **File** menu, click **New Stream**.

You will find the solution results in **C:\Training\0A058\Solutions** folder.



IBM Training

Exercise 2

Determine the availability of taxis using geospatial and time data

Working with sequence data

© Copyright IBM Corporation 2018

Exercise 2: Determine the availability of taxis using geospatial and time data

Exercise 2: Determine the availability of taxis using geospatial and time data

Data file:	**taxi_locations_and_times.csv**
Data folder:	**C:\Training\0A058**

You will continue with the taxi cab example. In the demonstration, you analyzed pings from people who opted in on the taxi cab company's cell phone app. Apart from this customer dataset, there is a dataset that stores the location and time of taxi cabs on Dec-31-2013.

- Import data from **taxi_locations_and_times.csv**, located in the **C:\Training\0A058** folder.
- Determine the top 3 space-time-boxes where taxis hang out (the three space-time-boxes with the highest number of taxi cabs). Use a 2.4 kilometers density for the geohash, and a 15 minutes time interval. Furthermore, a hangout qualifies as such if the dwell time is at least 5 minutes. (Use defaults for the other parameters.)

  Note: The stream **taxi_availabity_stb.str** that ships with IBM SPSS Modeler uses cell phone information and taxi times/locations to match the demand with the supply. Those interested are referred to this sample stream for more information on space-time-boxes.
- Exit IBM SPSS Modeler without saving anything.

For more information about where to work and the exercise results, refer to the Task and Results section that follows. If you need more information to complete a task, refer to earlier demonstrations for detailed steps.

Exercise 2: Tasks and Results

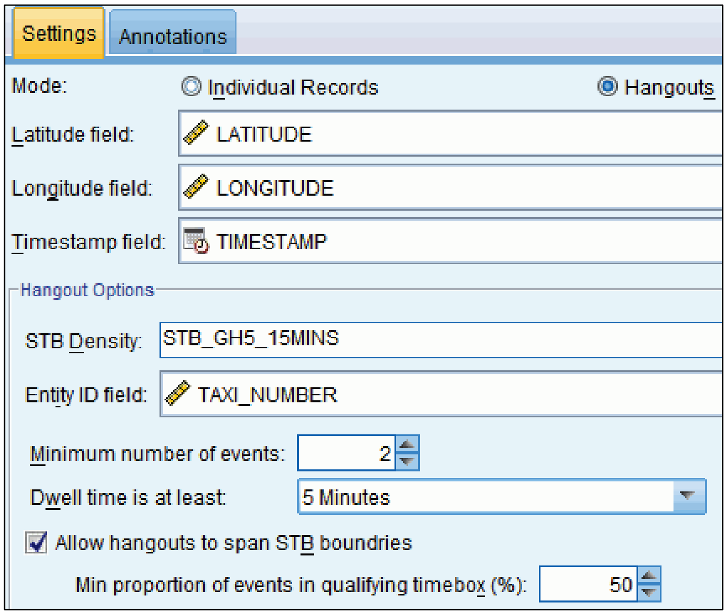
Task 1. Analyze geospatial and time data.

- From the **Sources** palette, add a **Var. File** node to the stream canvas and import data from **taxi_locations_and_times.csv**, located in the **C:\Training\0A058** folder.

Task 2. Determine the top 3 space-time boxes.

- From the **Record Ops** palette, add a **Space-Time-Boxes** node downstream from the **Var. File** node.
- Edit the **Space-Time-Boxes node**:
  - Beside **Mode**, select **Hangouts**.
  - Beside **Latitude** field, select **LATITUDE**.
  - Beside **Longitude** field, select **LONGITUDE**.
  - Beside **Timestamp** field, select **TIMESTAMP**.
  - Beside **STB Density**, select **GH5 (approx 2.4 kilometers)** and a **15 Minutes** time interval.
  - Beside **Entity ID** field, select **TAXI_NUMBER**.
  - Ensure that **Minimum number of events** is **2**.
  - Beside **Dwell time is at least**, select **5 Minutes**.

  The results appear as follows:

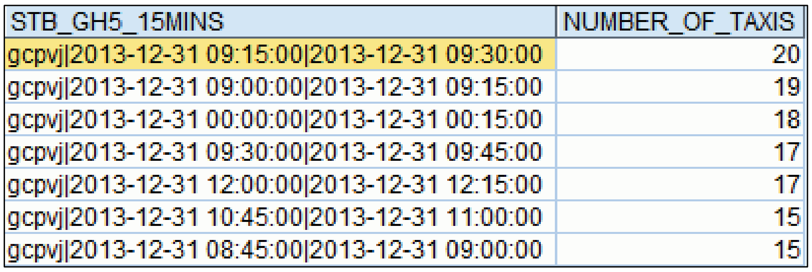


Mode: Individual Records [✓ Hangouts]
Latitude field: LATITUDE
Longitude field: LONGITUDE
Timestamp field: TIMESTAMP
Hangout Options
STB Density: STB_GH5_15MINS
Entity ID field: TAXI_NUMBER
Minimum number of events: 2
Dwell time is at least: 5 Minutes
[✓] Allow hangouts to span STB boundries
Min proportion of events in qualifying timebox (%): 50

- From the **Record Ops** palette, add an **Aggregate** node downstream from the **Space-Time-Boxes** node.
- Edit the **Aggregate** node:
    - For **Key fields**, select **STB_GH5_15MINS**.
    - Ensure that the **Include record count in field** option is enabled, and type **NUMBER OF TAXIS** in the text box.

  To know in which area and time interval the availability is highest, sort the data descending on NUMBER OF TAXIS.
- From the **Record Ops** palette, add a **Sort** node downstream from **Aggregate** node.
- Edit the **Sort** node:
    - Set the **Sort by** field to **NUMBER OF TAXIS**.
    - Set **Order** to **Descending**.
    - Click **Preview**.

    The results appear as follows:



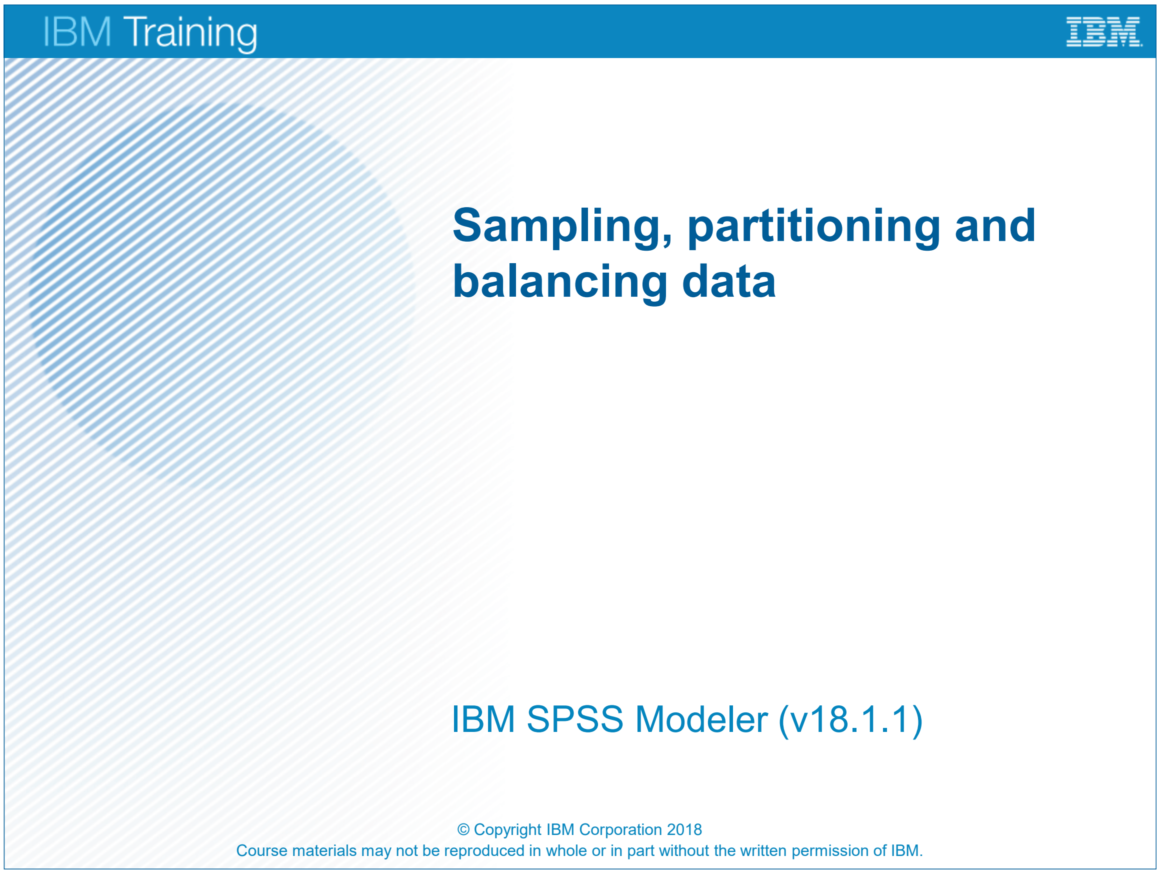
STB_GH5_15MINS	NUMBER_OF_TAXIS
gcpvj|2013-12-31 09:15:00|2013-12-31 09:30:00	20
gcpvj|2013-12-31 09:00:00|2013-12-31 09:15:00	19
gcpvj|2013-12-31 00:00:00|2013-12-31 00:15:00	18
gcpvj|2013-12-31 09:30:00|2013-12-31 09:45:00	17
gcpvj|2013-12-31 12:00:00|2013-12-31 12:15:00	17
gcpvj|2013-12-31 10:45:00|2013-12-31 11:00:00	15
gcpvj|2013-12-31 08:45:00|2013-12-31 09:00:00	15

Task 3. Exit IBM SPSS Modeler

- Exit **IBM SPSS Modeler** without saving.

You will find the solution results in the **C:\Training\0A058\Solutions** folder.

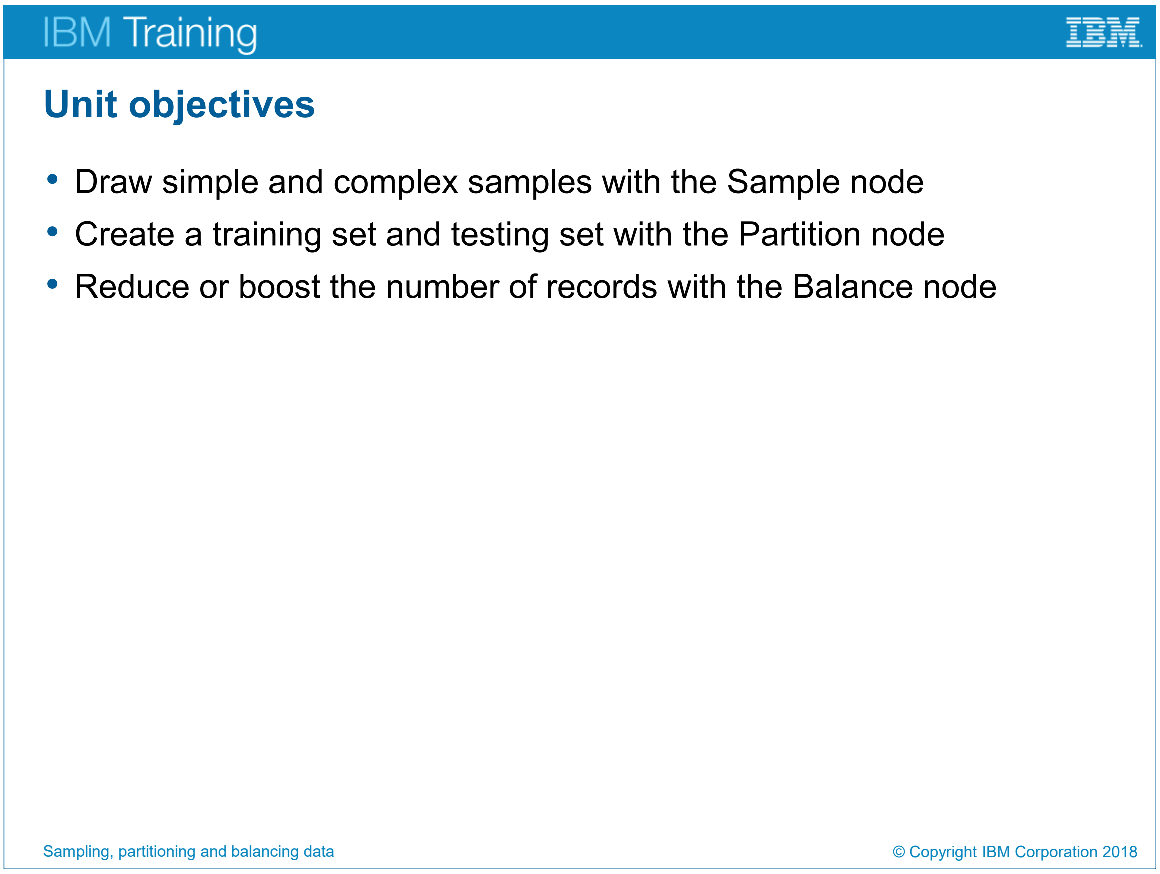
Unit 4 Sampling, partitioning and balancing data



IBM Training

Sampling, partitioning and balancing data

IBM SPSS Modeler (v18.1.1)


IBM Training

Unit objectives

- Draw simple and complex samples with the Sample node
- Create a training set and testing set with the Partition node
- Reduce or boost the number of records with the Balance node

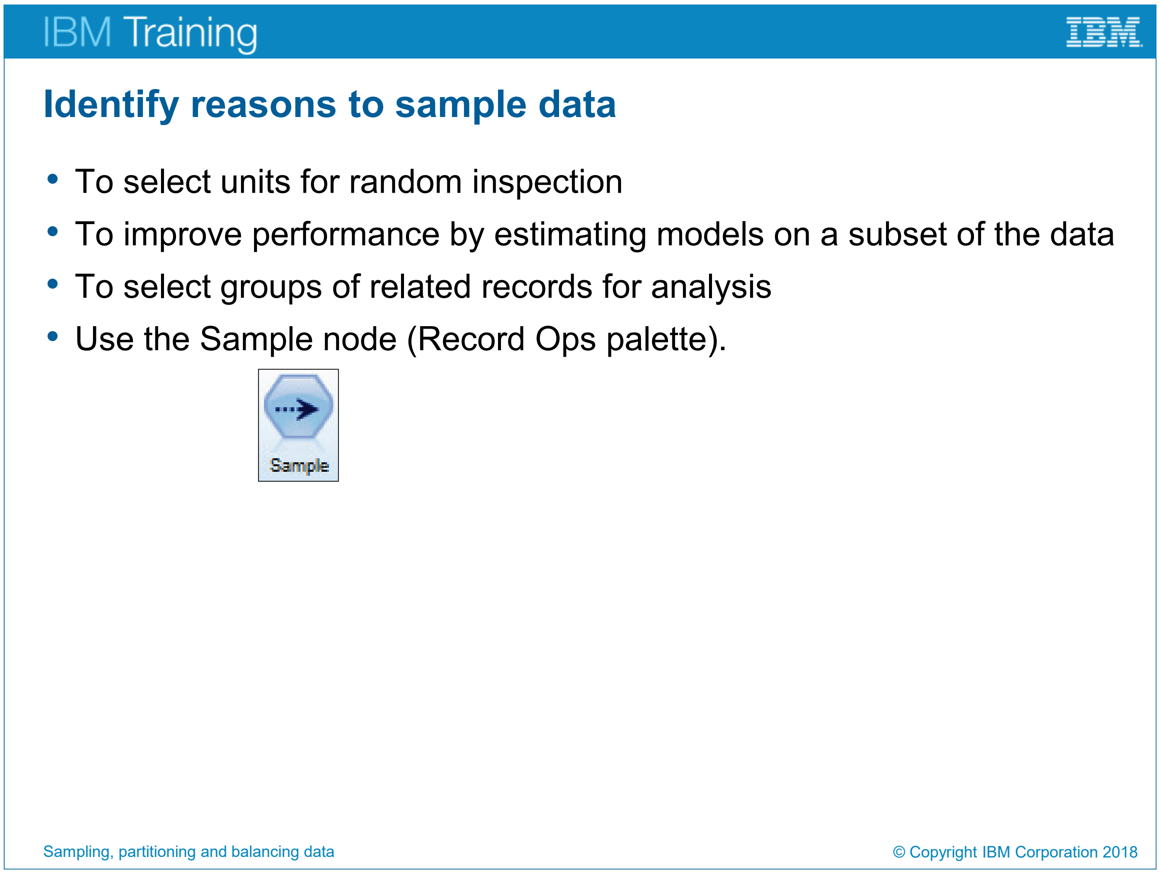
Sampling, partitioning and balancing data

© Copyright IBM Corporation 2018

Unit objectives

Before reviewing this unit, you should be familiar with:

- working with IBM SPSS Modeler (streams, nodes, SuperNodes, palettes)
- importing data (Var. File node)
- defining measurement levels, roles, blanks, and instantiating data (Type node)
- selecting fields and records (Filter node, Select node)
- CLEM and the Expression Builder
- deriving fields (Derive node)
- running a model and examining the generated model nugget
- examining the data with graphs (Distribution node, Histogram node) and tables (Table node, Data Audit node, Statistics node, Matrix node, Means node)



IBM Training

Identify reasons to sample data

- To select units for random inspection
- To improve performance by estimating models on a subset of the data
- To select groups of related records for analysis
- Use the Sample node (Record Ops palette).

Sample

Sampling, partitioning and balancing data

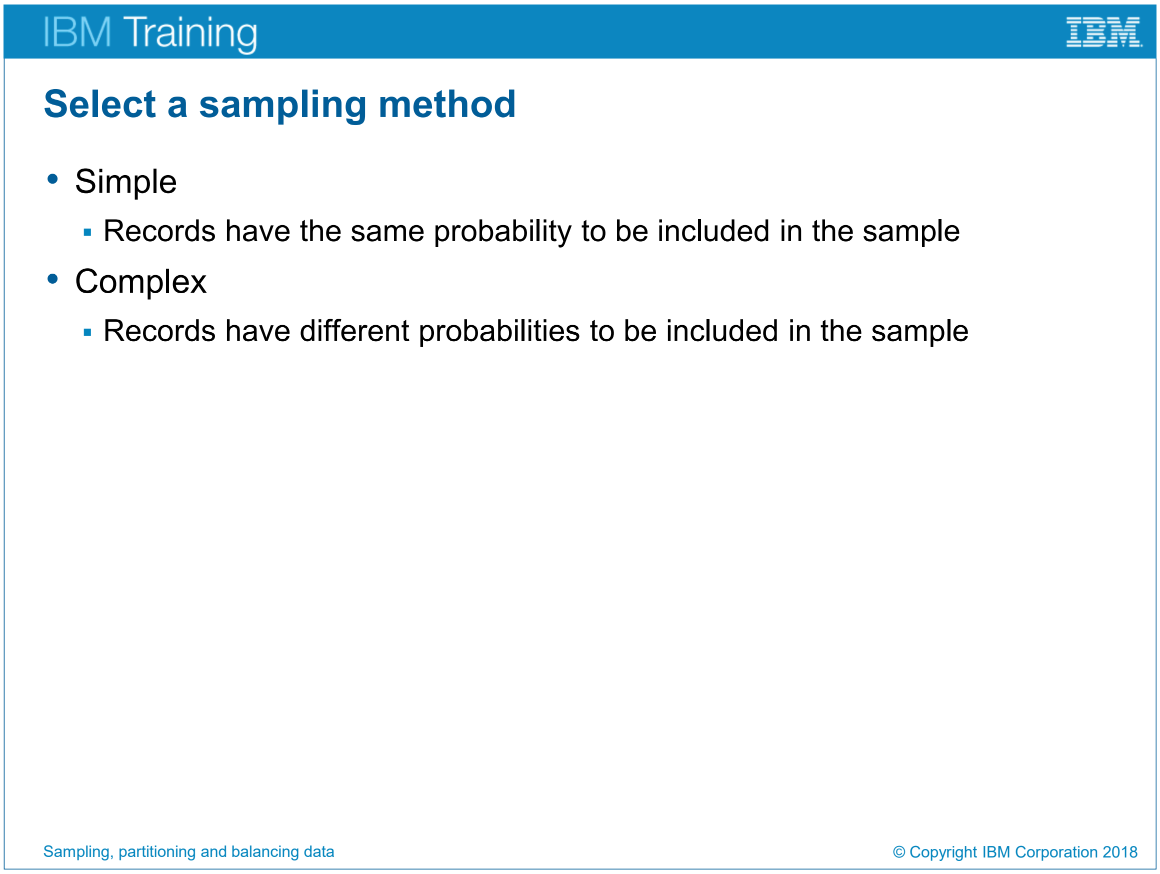
© Copyright IBM Corporation 2018

Identify reasons to sample data

Sampling can be used for several reasons:

- Select records for random inspection in the interest of quality assurance, fraud prevention, or security.
- To improve performance by estimating models on a subset of the data. Models estimated from a sample are often as accurate as those derived from the full dataset, and may be more so if the improved performance allows the user to experiment with different methods one might not otherwise have attempted.
- To select groups of related records or transactions for analysis, such as selecting all the items in an online shopping cart (or market basket), or all the properties in a specific neighborhood.

Use the Sample node to sample data; the Sample node is located in the Record Ops palette.


IBM Training

Select a sampling method

- Simple
  - Records have the same probability to be included in the sample
- Complex
  - Records have different probabilities to be included in the sample

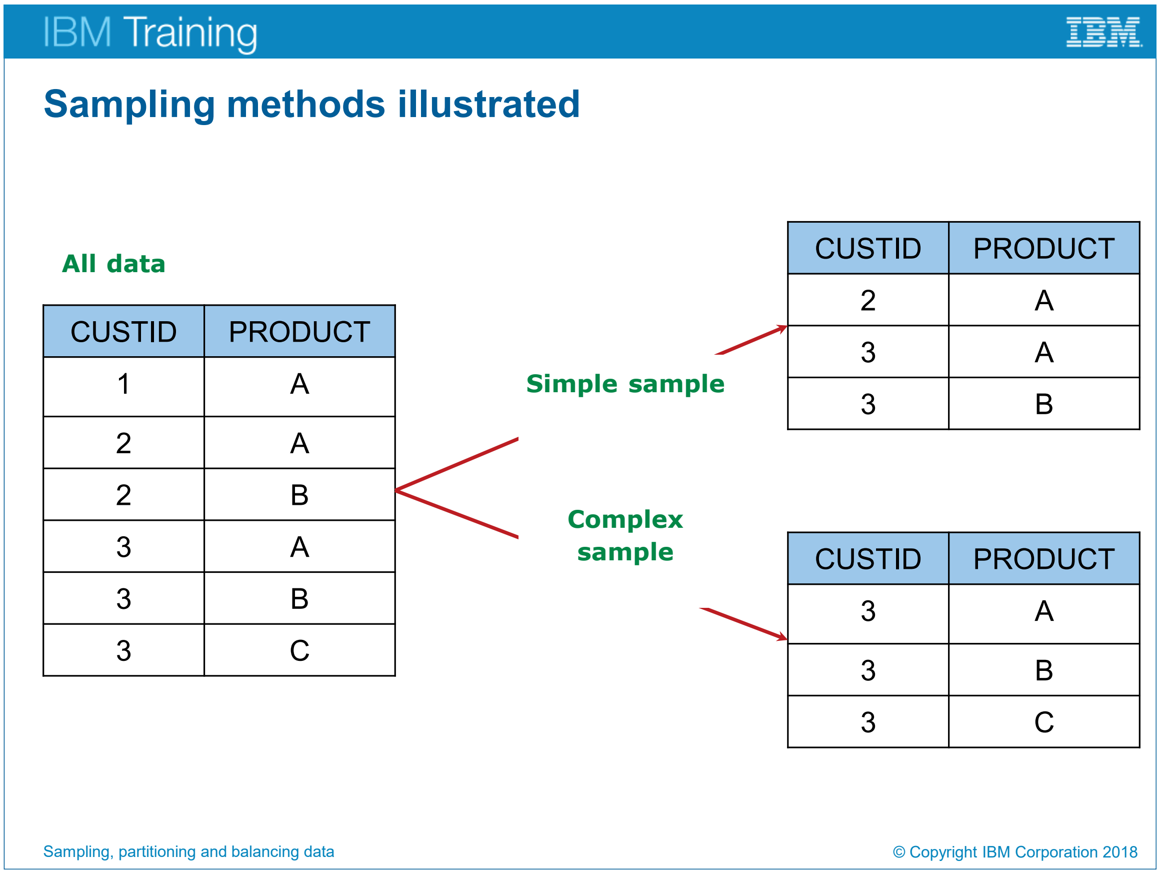
Sampling, partitioning and balancing data © Copyright IBM Corporation 2018

Select a sampling method

IBM SPSS Modeler supports two sampling methods:

- Simple: Drawing a sample is achieved by a simple mechanism, such as selecting a sample where every record has the same probability of being included in the sample. Refer to the next slide for other simple sampling techniques.
- Complex: Records have a different probability of being included in the sample. For example, if there are small subgroups in the data but if you want to ensure that records are sampled from these subgroups, random sampling does not suffice because you would run the risk of not including any person from these subgroups.

The sampling method must be fit to the problem at hand. For example, if you have a database of similar customers and you simply want to reduce the dataset size for modeling, you can use one of the simple methods. But if you need to sample directly from a database with customers of different types, you may want to draw a complex sample.


IBM Training

Sampling methods illustrated

All data

CUSTID	PRODUCT
1	A
2	A
2	B
3	A
3	B
3	C

Simple sample

CUSTID	PRODUCT
2	A
3	A
3	B

Complex sample

CUSTID	PRODUCT
3	A
3	B
3	C

Sampling, partitioning and balancing data

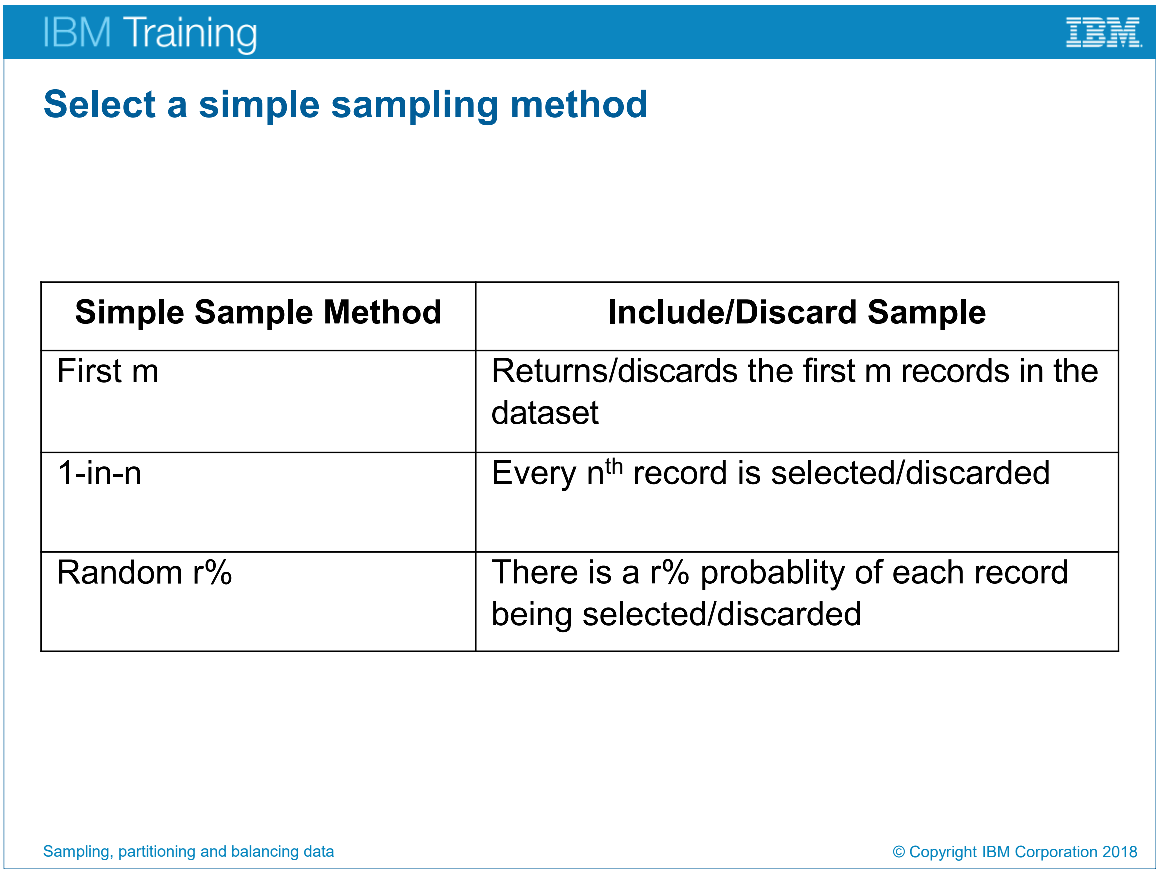
© Copyright IBM Corporation 2018

Sampling methods illustrated

The dataset depicted on the left side of this slide includes 6 records, representing data from 3 customers.

A simple sample of size three, with each record having the same probability of being included, may result in a dataset with three records, with data from two customers.

The complex sample did not draw a sample from records, but a sample from customers. Thus, customers have the same probability of being in the sample, not records. Having randomly selected customers, all records for the selected customers were included in the sample. In this example, a random selection of customers resulted in one customer being selected, CUSTID 3. All records of this customer were output to arrive at the complex sample. In this example, a business question might be to find popular product combinations, an analysis known as market basket analysis. Imagine a dataset such as the one depicted on the left side, but then with millions of records. An algorithm may have difficulties to find associations in a dataset of this size, and you may consider drawing a sample. Then, which sample would you draw? When you look for product combinations it is important to have all products for a certain customer in the sample, so a complex sample would be appropriate rather than a simple sample.


IBM Training

Select a simple sampling method

Simple Sample Method	Include/Discard Sample
First m	Returns/discards the first m records in the dataset
1-in-n	Every n^{th} record is selected/discarded
Random r%	There is a r% probablity of each record being selected/discarded

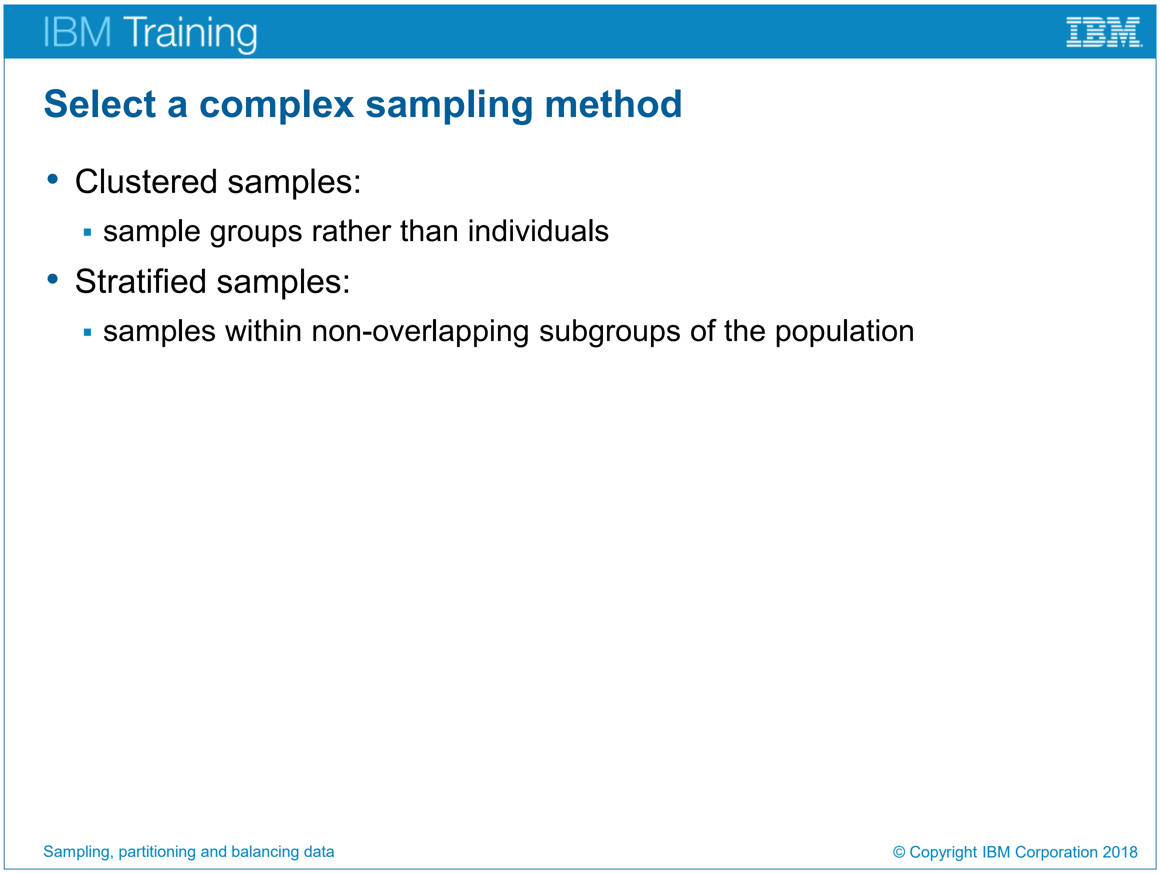
Sampling, partitioning and balancing data

© Copyright IBM Corporation 2018

Select a simple sampling method

This table lists the methods of simple sampling that IBM SPSS Modeler supports.

When a sample is drawn, the records in the sample can either be retained or discarded. The table describes each of the simple sample methods and their effect when used with the include sample or discard sample modes. For example, when a random sample of size r% is drawn in conjunction with the option to discard methods, r% of the records will be discarded, leaving 100%-r% in the sample.



IBM Training

Select a complex sampling method

- Clustered samples:
  - sample groups rather than individuals
- Stratified samples:
  - samples within non-overlapping subgroups of the population

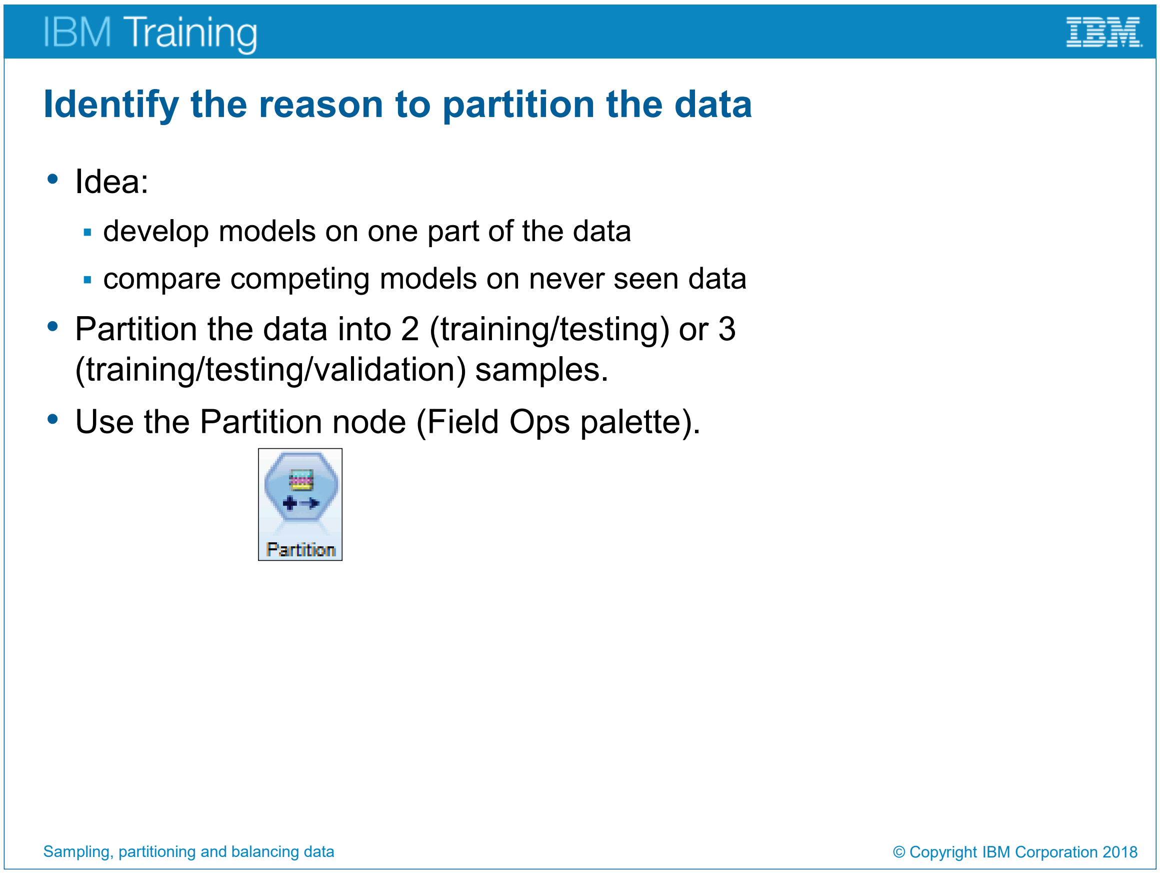
Sampling, partitioning and balancing data

© Copyright IBM Corporation 2018

Select a complex sampling method

IBM SPSS Modeler supports two complex sample methods:

- Clustered samples: Sample from groups of records rather than from individual records. Examples are:
  - Sample students by clustering by school. First randomly select schools, and then pick every student from each selected school. This will reduce costs because you only have to visit a limited number of schools.
  - Sample transactions by clustering by customer. First randomly select customers, and then include all transactional records from each customer in the sample. This ensures that all items from a customer are included in the sample. In market basket analysis, with millions of transactions, this will help to reduce the size of the dataset.
- Stratified samples: Sample independently within subgroups, called strata in this context. For example, you can stratify by gender to ensure that men and women are sampled in equal proportions, or by region to ensure that each region within an urban population is represented. It is also possible to specify a different sample size for each stratum. For example, sampling for a survey, you can give men a higher probability to be included in the sample than women when you think that the willingness to participate in the survey is lower for men than for women.



IBM Training

Identify the reason to partition the data

- Idea:
  - develop models on one part of the data
  - compare competing models on never seen data
- Partition the data into 2 (training/testing) or 3 (training/testing/validation) samples.
- Use the Partition node (Field Ops palette).

Partition

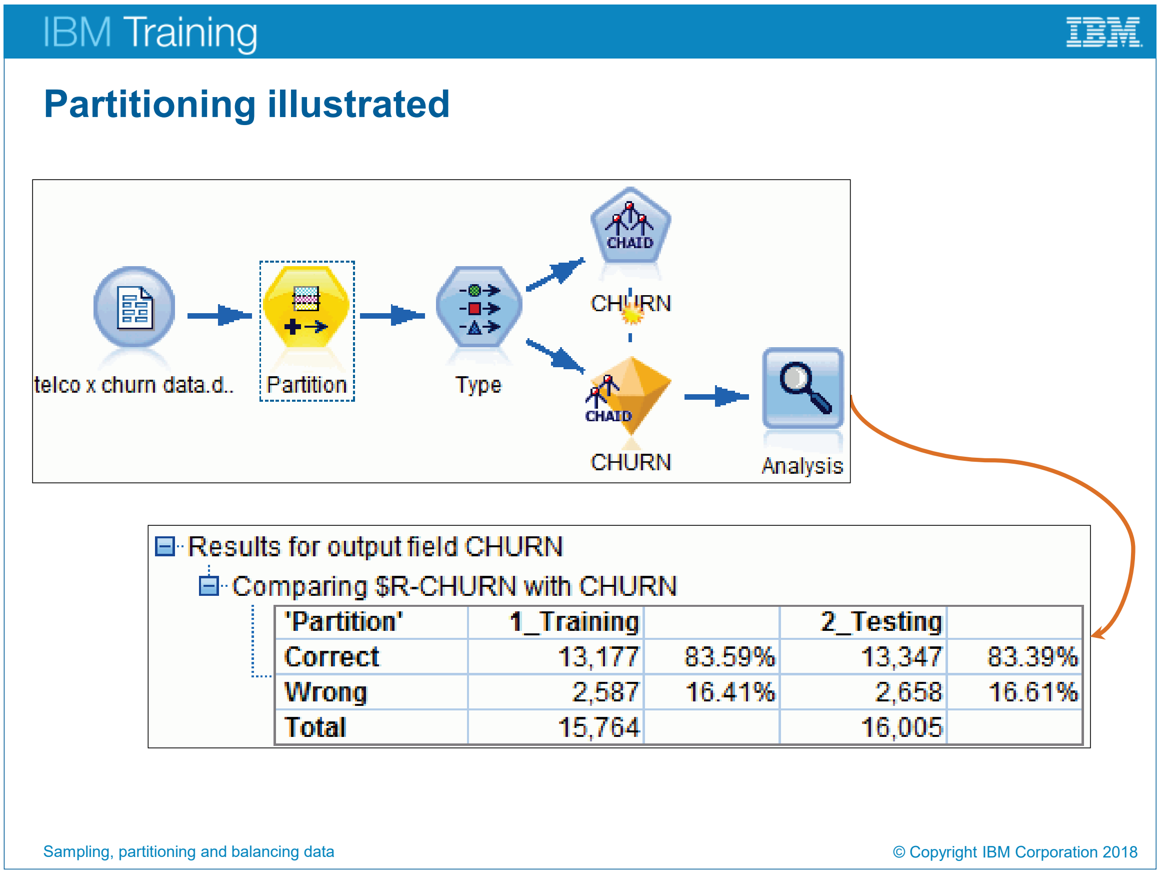
Sampling, partitioning and balancing data

© Copyright IBM Corporation 2018

Identify the reason to partition the data

Partitioning splits the data randomly into a training dataset and a testing dataset. The model is built on the training set, and the model is applied to the testing set to establish its credibility. Or, you might build competing models and test them on never seen data to get a more reliable estimate of the models' accuracies.

You can subdivide the data into three samples instead of just two. The model will still be built on the training set and tested on the testing set. However, the testing set can then be used to further refine the model. For example, if you believe that the performance of the model needs improvement, the parameters can be changed, and the model is then rebuilt using the training sample, after which the performance on the testing set is examined. You may have to do this several times before the results are satisfactory. The validation sample, which unlike the training and testing sets played no role in developing the final model, is then used to assess the model's performance against yet unseen data.Use the Partition node, located in the Field Ops palette, to create the partitions. Rather than physically creating two datasets, the Partition node adds a new field that assigns a record to one of the partitions, retaining one dataset.



IBM Training

Partitioning illustrated

Results for output field CHURN

Comparing $R-CHURN with CHURN

'Partition'	1_Training		2_Testing	
Correct	13,177	83.59%	13,347	83.39%
Wrong	2,587	16.41%	2,658	16.61%
Total	15,764		16,005	

Sampling, partitioning and balancing data

© Copyright IBM Corporation 2018

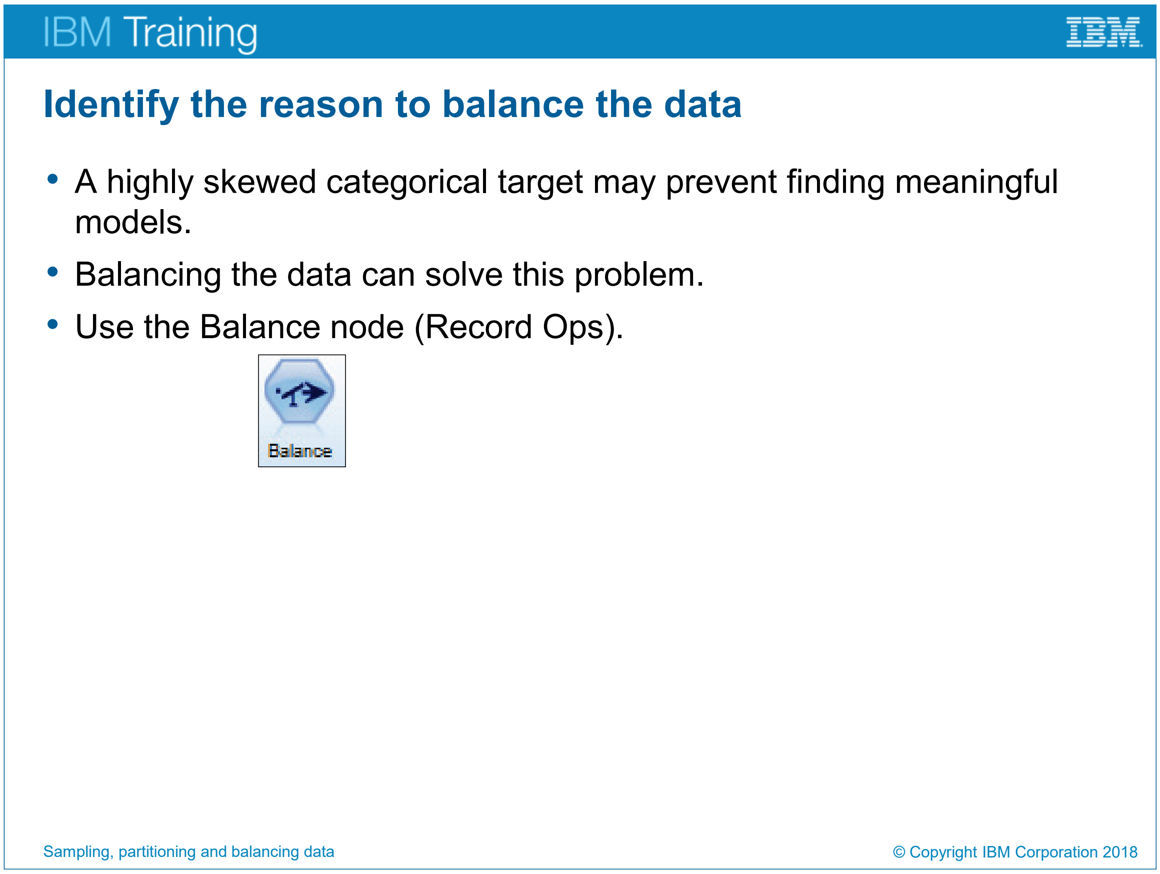
Partitioning illustrated

To make use of the Partition node, insert the node upstream from the Type node that sets the roles for the fields in modeling.

When you compute performance measures, the results will be reported for all partitions, so models can be compared easily.

This slide presents an example of partitioning the data. CHAID is used to predict CHURN. The Partition node is found upstream from the Type node. When you run the CHAID model, only the records that are assigned to the training set are used to build the model. When you evaluate the CHAID model nugget with an Analysis node, the Analysis node will show the results for both the training set and testing set.

When you would have tested multiple models, the Analysis node would have presented the results for all models, for both partitions, so that you can easily determine which model performed best on the testing set.



IBM Training

Identify the reason to balance the data

- A highly skewed categorical target may prevent finding meaningful models.
- Balancing the data can solve this problem.
- Use the Balance node (Record Ops).

Sampling, partitioning and balancing data

© Copyright IBM Corporation 2018

Identify the reason to balance the data

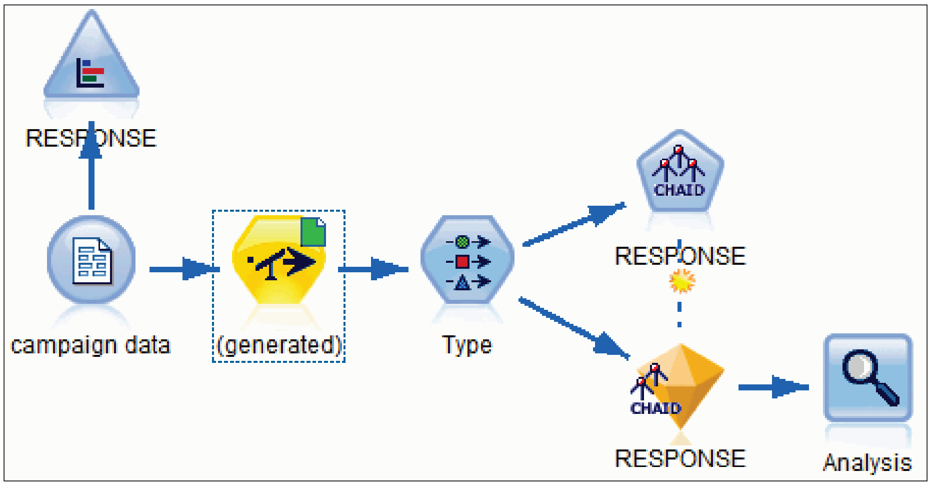
A target field may have too few records in certain categories to build models for it. For example, in predicting response to a mailing, in a dataset of 100,000 customers, there may only be 500 customers who responded positively to the mailing. Models may not be able to identify these customers when the distribution is that skewed. Balancing makes the distribution of a categorical field more equal, so better models can be built.

A balanced dataset is achieved by duplicating ("boosting") records in the lower-frequency categories or discarding ("reducing") records in the higher-frequency categories. In general, reducing records in the higher-frequency categories is preferred over boosting records in the lower-frequency categories. The reason is that when you boost record you will run the risk of duplicating anomalies in the data. But when a dataset is very small, or when the number of records in specific categories is too few, boosting may be the only method that makes it possible to develop a reasonable model. Therefore, models developed with balanced data must be validated, validated on a dataset that matches the (skewed) population distribution of the target.

Use the Balance node, located in the Record Ops palette, to balance the data. You can also generate a Balance node from the output of the Distribution node, as will be illustrated in the demonstration.

IBM Training

Balancing records illustrated



Sampling, partitioning and balancing data

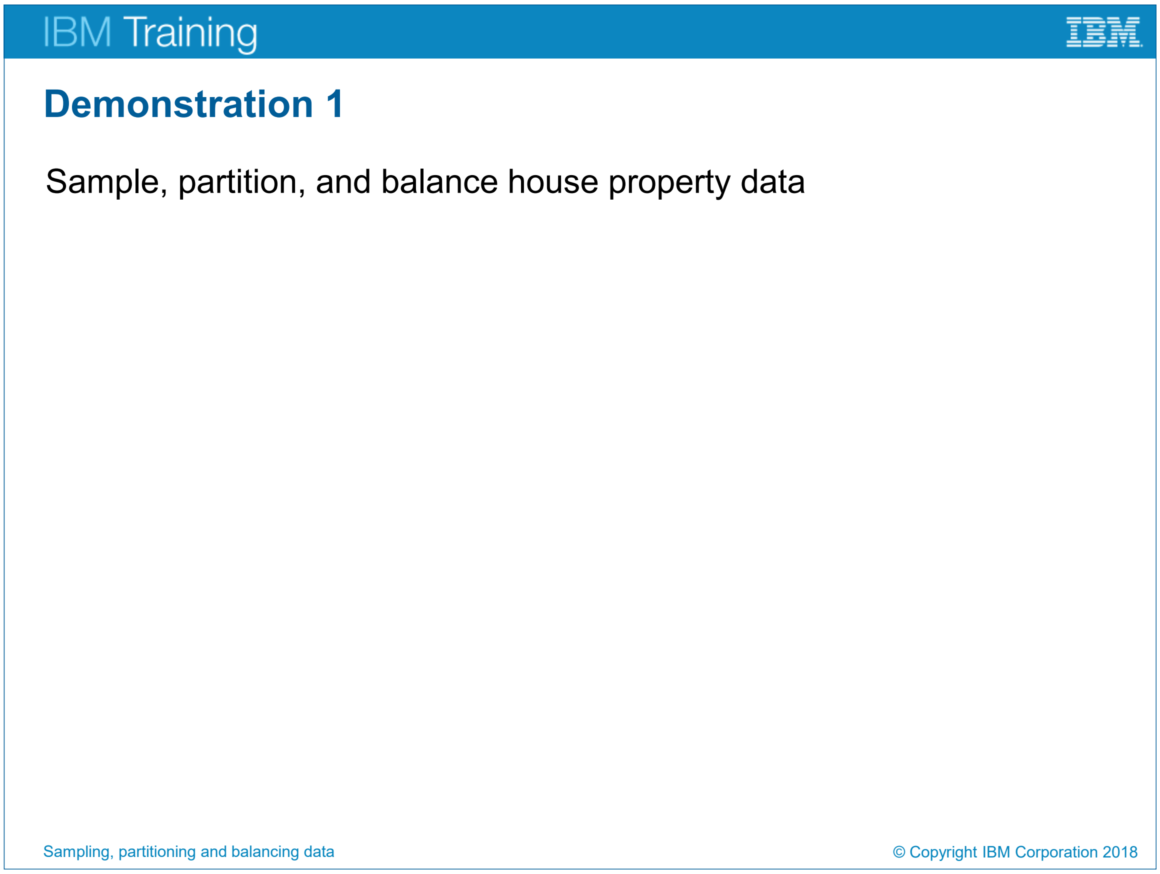
© Copyright IBM Corporation 2018

Balancing records illustrated

In the stream depicted on this slide, a Balance node is placed upstream from the Type node that sets the roles for modeling.

The Balance node has no option to set the random seed and each time that records pass through the node other records will be output because of the random mechanism. If you want to replicate the results, cache the data at the Balance node.

Note: Since the release of IBM SPSS Modeler 18.1, IBM SPSS Modeler provides a second node to balance the data, SMOTE. Refer to the *Advanced Predictive Modeling Using IBM SPSS Modeler* course for more information.



IBM Training

Demonstration 1

Sample, partition, and balance house property data

Sampling, partitioning and balancing data

Demonstration 1: Sample, partition, and balance house property data

Demonstration 1:
Sample, partition, and balance house property data

Purpose:
You will sample data using various techniques, and use partitioning to select the best predictive model.

Data file:	**property_assessment.csv**
Data folder:	**C:\Training\0A058**
Stream:	**unit_4_demonstration_1_start.str**
Stream folder	**C:\Training\0A058\Start**

Task 1. Start IBM SPSS Modeler and set the working folder.

1. From the **Start** menu, expand **IBM SPSS Modeler 18.1**, and then click **IBM SPSS Modeler 18.1**. When a splash window displays, click **Cancel**.

   If you have already configured IBM SPSS Modeler in a previous demonstration or exercise, you can skip to Task 2.
2. From the **File** menu, click **Set Directory**.
3. Beside **Look in**, navigate to **C:\Training\0A058**, and then click **Set**.

Task 2. Open the start stream and examine the data.

1. From the **File** menu, click **Open Stream**, navigate to the **C:\Training\0A058\Start** folder, click **unit_4_demonstration_1_start.str**, and then click **Open**.

   The stream imports the data, sets appropriate measurement levels and roles, and instantiates the data.
2. Scroll to the comment reading **Simple and complex sampling**.
3. Run the **Table** node.

   The dataset stores property values across a range of locations. The dataset contains 11,128 records with information on a property's county and township. There are 5 counties, and 49 townships. Each property is one record in the dataset. The value at the last appraisal is the value in thousands. For example, 192.6 represents a value of 192,600.
4. Close the **Table** output window.

Task 3. Draw a simple and complex sample.

Properties must be visited for inspection, for example to investigate if the value at the last appraisal was appropriate. In order to cut costs in the project, not all properties will be visited, but only 40% of them. Thus, the task is to randomly select 40% of the properties.

You will use simple sampling and complex sampling to select, approximately, 40% of the properties.

1. From the **Record Ops** palette, add a **Sample** node downstream from the **Type** node.
2. Edit the **Sample** node.

   The Simple sample method is selected by default. You can choose whether to pass records in the sample (Include sample) or discard them (Discard sample). You can then select the type of simple sampling and enter the appropriate value for sampling, for example the sample percentage. If you want to obtain the same sample every time, you can use the Repeatable partition assignment option and set the value of the seed of the sampling algorithm.
3. Beside **Sample method**, ensure that **Simple** is selected.
4. For **Sample**, select **Random %**, and set the value to **40**.
5. Click the **Repeatable partition assignment** check box to select it, and then, for **Seed**, type **1**.

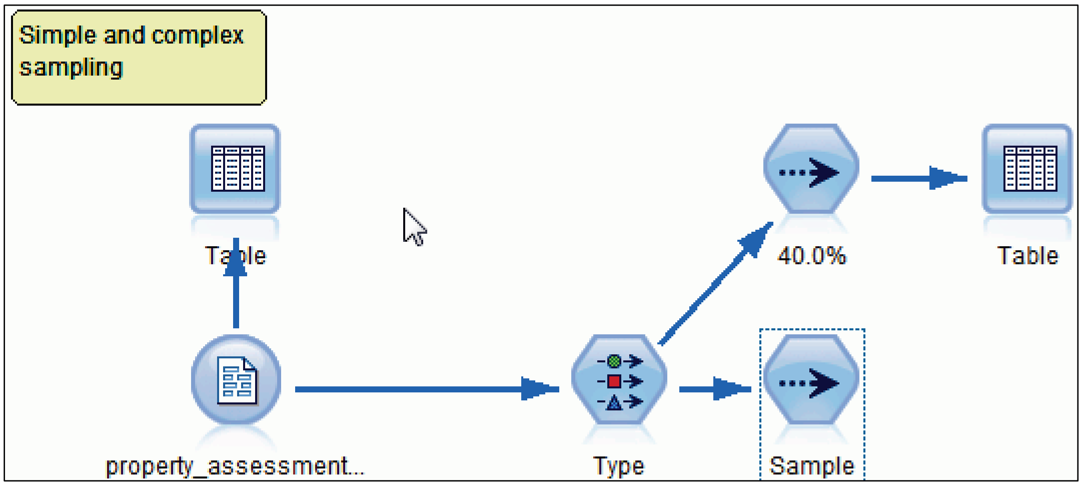
   Every sample will be drawn using the same parameter (seed value) for the sampling algorithm. Thus, every sample will be comprised of the same records.
6. Close the **Sample** dialog box.
7. From the **Output** palette, add a **Table** node downstream from the **Sample** node, and then run the **Table** node.

   4,398 records were sampled, which is 39.5% of the full dataset (11,128 records). Thus, approximately 40% of all records were selected.
8. Close the **Table** output window.

   One could use this sample and visit the selected properties, but one will run the risk that properties from some counties in the sample are under-represented, which might be a necessary condition for the project. Therefore, rather than drawing a simple random sample you will draw a sample that is stratified by county. Also, suppose that you want to reduce the costs to visit properties. Then, you will not sample from properties, but from townships. Thus, you will draw a clustered sample from townships; you will then have fewer townships to visit. For a township that is sampled, all properties will be included in the sample.

9. From the **Record Ops** palette, add a second **Sample** node downstream from the **Type** node.

   The results appear similar to the following:



10. Edit the **Sample** node.
11. For **Sample method**, select **Complex**.

    You can sample by specifying proportions or counts, and you can customize the sample size for each stratum.

    You will draw a clustered sample based on TOWNSHIP, and stratify by COUNTY.
12. Click the **Cluster and Stratify** button.
13. Beside **Clusters**, select **TOWNSHIP**.

    Notice that you can cluster only by one field at a time.
14. Beside **Stratify** by, select **COUNTY**.

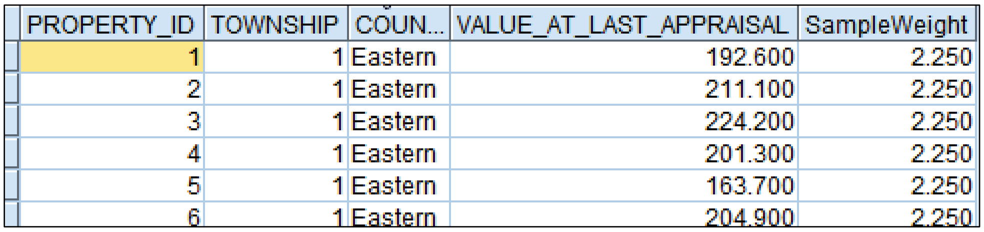
    Notice that you can stratify by multiple fields.
15. Click **OK** to return to the main dialog box.

    The sample size is set to 0.5 by default. This means that approximately half of the units are sampled within each stratum (units being the clusters (townships), strata the counties). For example, if a county has 8 townships, it is expected that 4 of them are selected. All properties within a selected township will be in the sample because of the clustering.

    In this example you will draw a sample of 40% rather than of 50%, so you will change this parameter.
16. For **Sample size**, option **Fixed**, enter **0.4**.
17. Click the **Repeatable partition assignment** check box to select it, and then, for **Seed** enter value **1**.
18. Close the **Sample** dialog box.

19. From the **Output** palette, add a **Table** node downstream from the **Sample** node named **Complex.**
20. Run the **Table** node.

    The results appear similar to the following:



PROPERTY_ID	TOWNSHIP	COUN...	VALUE_AT_LAST_APPRAISAL	SampleWeight
1	1	Eastern	192.600	2.250
2	1	Eastern	211.100	2.250
3	1	Eastern	224.200	2.250
4	1	Eastern	201.300	2.250
5	1	Eastern	163.700	2.250
6	1	Eastern	204.900	2.250

    The SampleWeight field is automatically added to the data when you draw a complex sample. This field's values correspond, approximately, to the frequency that each sampled record represents in the original data. For example, the first record's sample weight is 2.25, which means that this record represents 2.25 records in the original data. When you sum the weights you would get, approximately, the number of records in the original dataset.

    It can be verified that 20 townships were drawn, which is (20/49) = 40.8% of the townships, close to the 40% that was requested. This results in 4,338 properties drawn, which is (4,338/11,128) = 38.9% of all properties.

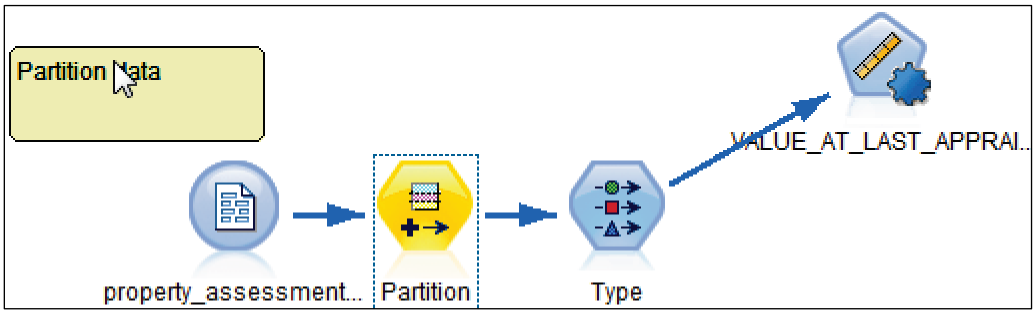
21. Close the **Table** output window.

Task 4. Partition the data into a training set and a testing set.

You want to predict the property value using a number of competing models. You will build the models on 70% of the data, and test the models on 30% of the data.

1. Scroll to the comment that reads **Partition data**.
2. From the **Field Ops** palette, add a **Partition** node to the stream, and insert it between the **Var. File** node and the **Type** node.

   The results appear similar to the following:



3. Edit the **Partition** node.
4. In the **Training partition size** text box, enter **70**.
5. In the **Testing partition size** text box, enter **30**.

6. Ensure that the **Repeatable partition assignment** option is enabled.
7. In the **Seed** text box, type **1**.
8. Close the **Partition** dialog box.
9. Run the **Auto Numeric** node named **VALUE_AT_LAST_APPRAISAL**.
10. Edit the generated **model nugget** (the yellow diamond).

    The best three models are presented, based on the results on the testing set. All three models perform equally well in terms of correlation and relative error, using the testing set. Thus, it is a matter of taste which model you prefer.

    At this point, you will not go into the modeling details. All in all, it was demonstrated how to compare a number of competing models on never seen data, by partitioning the data into a training set and a testing set.
11. Close the **model nugget** window.

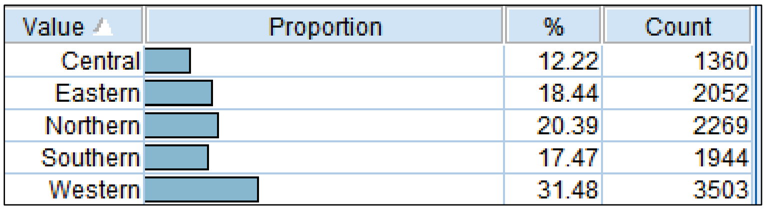
Task 5. Balance the data.

You will use the Balance node to get the same number of records in each county, by reducing the number of records in the larger categories.

Note: Balancing will in general be done on the target field. Although this demonstration does not follow this principle, it illustrates how to balance data.

1. Scroll to the third part of the stream, where the comment reads **Balancing data**.
2. Run the **Distribution** node for **COUNTY**.

    The results appear as follows.



Value	Proportion	%	Count
Central		12.22	1360
Eastern		18.44	2052
Northern		20.39	2269
Southern		17.47	1944
Western		31.48	3503

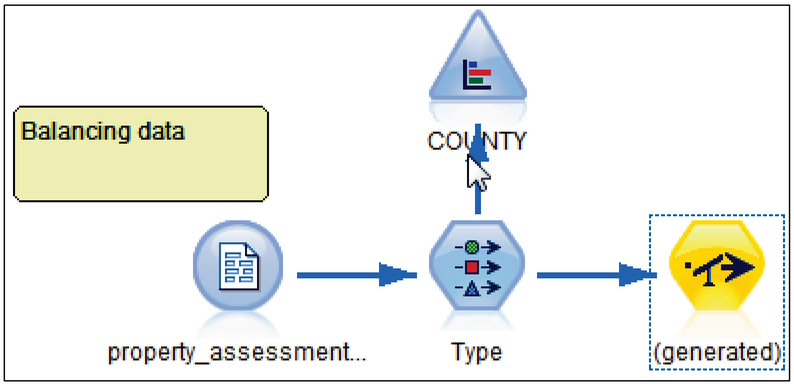
You might get slightly different results, because the Balance node has no seed value to replicate the results.

The Central category has the lowest frequency. When you reduce the number of records in the larger categories, this category will have a weight of 1. The other categories will have a smaller weight and the Western category will have the smallest weight.

3. In the **Distribution** output window, from the **Generate** menu, click **Balance Node (reduce)**.
4. Close the **Distribution** output window.

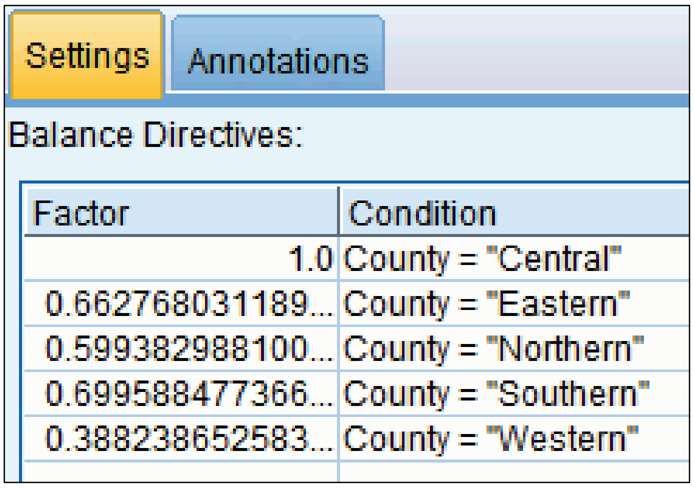
5. Add the generated **Balance** node (located in the upper left corner on the stream canvas) downstream from the **Type** node.

   The results appear similar to the following:



6. Edit the **Balance** node.

   The results appear as follows:



Balance Directives:

Factor	Condition
1.0	County = "Central"
0.662768031189...	County = "Eastern"
0.599382988100...	County = "Northern"
0.699588477366...	County = "Southern"
0.388238652583...	County = "Western"

   The weights are as expected.

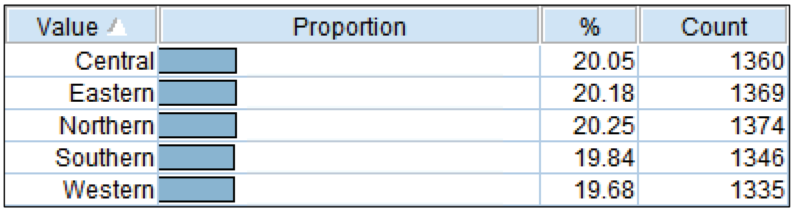
7. Close the **Balance** dialog box.

   To check the effect of balancing, you will run a Distribution node on the balanced data.

8. From the **Graphs** palette, add a **Distribution** node downstream from the **Balance** node.

9. Edit the **Distribution** node, for **Field**, select **COUNTY**, and then run the **Distribution** node.

   The results appear as follows:

Value	Proportion	%	Count
Central		20.05	1360
Eastern		20.18	1369
Northern		20.25	1374
Southern		19.84	1346
Western		19.68	1335

   The categories have approximately the same number of records, as required.

   Note: Your results might be slightly different because the Balance node has no seed value for the random mechanism and so the results will differ from run to run.

   You may want to build a new model to predict the value. We leave that as an exercise. As noticed, balancing is usually done to balance the number of records in the categories of the target field. In this respect the demonstration is a-typical, but serves its purpose.

10. Close the **Distribution** output window.

    This completes the demonstration. You will create a clean state for the exercises.

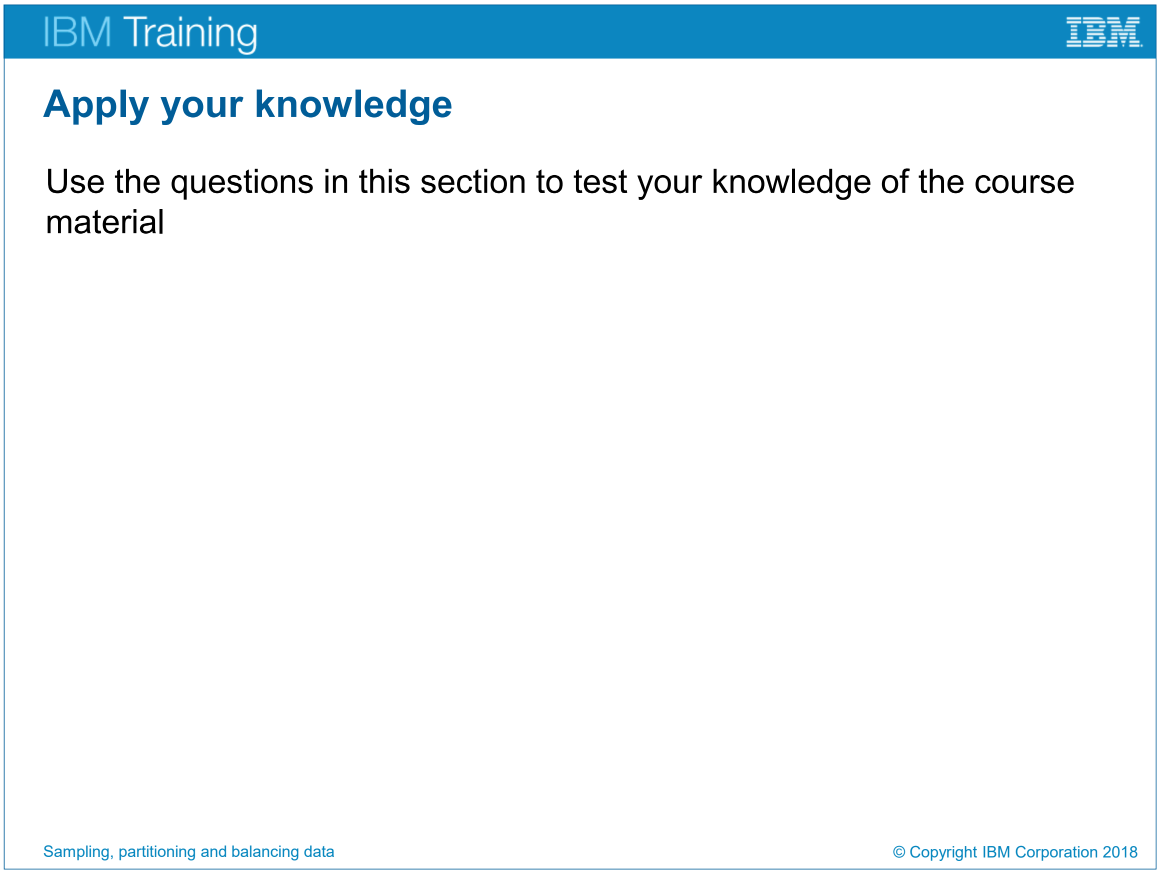
11. From the **File** menu, click **Close Stream**. Click **No** when asked to save the stream.

12. From the **File** menu, click **New Stream**.

    Leave IBM SPSS Modeler open for the exercise.

Results:
You have sampled data using various techniques and you partitioned your data so that you could choose the predictive model that performed best on a testing dataset.

This completes the demonstration for this unit. You will find the solution results in the **C:\Training\0A058\Solutions** folder.



IBM Training

Apply your knowledge

Use the questions in this section to test your knowledge of the course material

Sampling, partitioning and balancing data

© Copyright IBM Corporation 2018

Apply your knowledge

Questions

Question 1: Select all that apply. The Sample node, mode Simple, enables you to:

A. Draw a random sample of 40%.

B. Include the first 1000 records, discard records 1001 thru 2000 and include records 2001 and further.

C. Replicate a sample.

D. Draw a random sample with a maximum sample size.

Question 2: True or false: A dataset has 1,000 records, and a random sample of size 50% is drawn. Then exactly 500 records will be drawn.

A. True

B. False

Question 3: You have a categorical field that defines certain groups. You want to ensure that records from all these groups are included in the sample. The type of sample you need is a:

A. Simple sample

B. Clustered sample

C. Stratified sample

Question 4: True or false: The Sample node enables you to draw a clustered sample within a stratified sample.

A. True

B. False

Question 5: You want to first draw a clustered sample from schools (field SCHOOL), and then draw a clustered sample from classes (field CLASS) within schools. What will you do?

A. Specify SCHOOL and CLASS in the Cluster by area in the Sample node.

B. Specify SCHOOL in the Cluster by area in a first Sample node, and then add a second Sample node downstream from the first sample node and specify CLASS in the Cluster by area in this second Sample node.

Question 6: When analyzing product associations in shopping carts (each product purchased forms up one record in the dataset) you want to work with sample data but you want to ensure that all items purchased from a selected transaction are included in the sample. The type of sample you need is a:

A. Simple sample

B. Clustered sample

C. Stratified sample

Question 7: True or false: The reason for partitioning is to evaluate the model on data that is not used when the model is built.

A. True

B. False

Question 8: True or false: The partition field has measurement level continuous.

A. True

B. False

Question 9: True or false: The objective for balancing data is to build better models.

A. True

B. False

Question 10: When you want to replicate the results of the balance node within a IBM SPSS Modeler session, you need to:

A. Set the value of the random seed in the Balance node.

B. Cache the data on the Balance node.

C. Do not use the Balance node, but the Partition node to boost or reduce records.

D. None of the above statements are correct.

Answers to questions

Answer 1: A, C, D. The Sample node enables you draw a random sample of 40%, to replicate a sample, and to limit the size of the sample. You cannot sample blocks, as alternative B suggests.

Answer 2: B. False. Approximately 50% will be drawn.

Answer 3: C. When you want to ensure that records from all groups are in the sample, you need to draw a stratified sample, where the strata are defined by the categorical field.

Answer 4: A. True. The Complex sample mode allows you to specify a cluster field and a stratification field.

Answer 5: B. You can only specify one field as cluster field at a time in the Sample dialog box. In order to cluster on two fields you can add two Sample nodes to your stream.

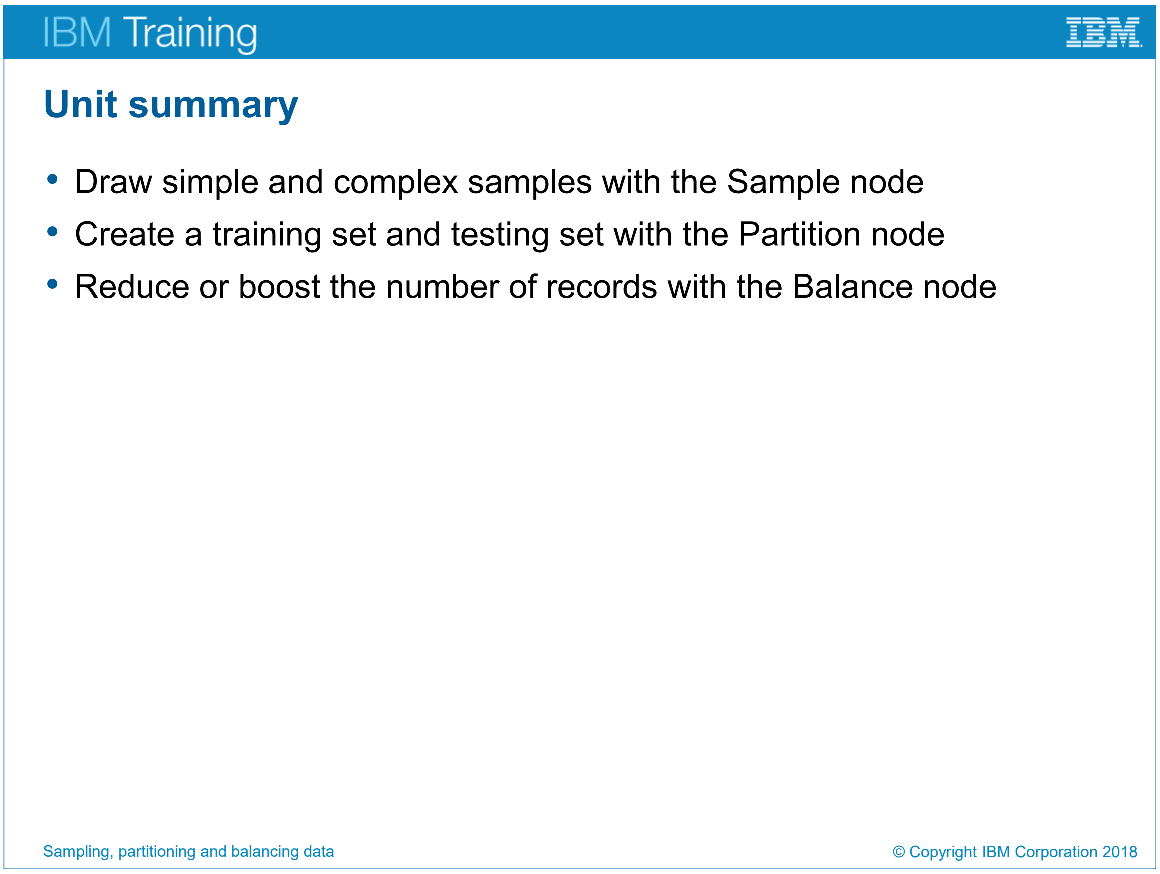
Answer 6: B. When you want to ensure that all items from a selected transactions are in the sample, you need a clustered sample.

Answer 7: A. Partitioning data enables you to evaluate the model on data that is not used when you built the model.

Answer 8: B. False, the partition field is a nominal field, not continuous.

Answer 9: A. True. Data is balanced to build better models.

Answer 10: B. The Balance node does not have the option to set the seed. If you want to replicate results within the same session you should cache the data on the Balance node.



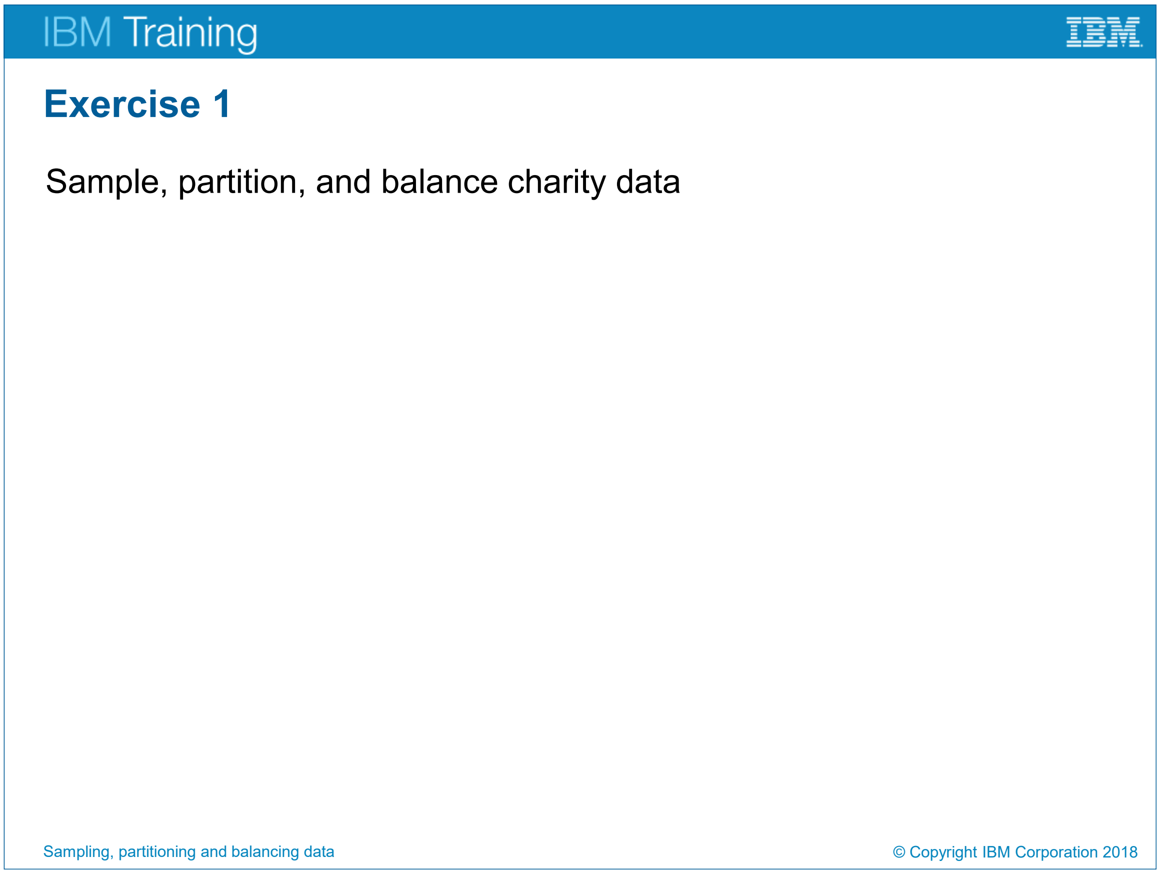
IBM Training

Unit summary

- Draw simple and complex samples with the Sample node
- Create a training set and testing set with the Partition node
- Reduce or boost the number of records with the Balance node

Sampling, partitioning and balancing data

Unit summary



IBM Training

Exercise 1

Sample, partition, and balance charity data

Sampling, partitioning and balancing data

© Copyright IBM Corporation 2018

Exercise 1: Sample, partition, and balance charity data

Exercise 1: Sample, partition, and balance charity data

Data file:	**charity.csv**
Data folder:	**C:\Training\0A058**
Stream:	**unit_4_exercise_1_start.str**
Stream folder	**C:\Training\0A058\Start**

You will use an existing stream to sample, partition and balance data.

- Open **unit_4_exercise_1_start.str**, located in the **C:\Training\0A058\Start** folder.

  Scroll to the comment that reads **Sampling and balancing data**.

  Run the **Table** node to examine the data.

  Run the **Distribution** node.

  What percentage responded to the campaign?

- Your organization wants to conduct a satisfaction survey on a 10% random sample. Sample 10% of the records, using seed value 1.

  How many records were sampled? Is this exactly 10% of all records?

- Instead of drawing a random sample, you want to draw a sample so that you have approximately the same number of responders and non-responders.

  Using the complex sample method, draw a sample of 0.5% (a proportion of 0.005) of the non-responders, and 10% (a proportion of 0.1) of the responders. Use 1 as seed value.

  Run a **Distribution** node on **RESPONSE_TO_CAMPAIGN** to check your results.

- In order to build a model on training data and evaluate the model on test data, you will partition the data. Once you have partitioned the data, you will run a model.

  - Scroll to the comment that reads **Partitioning the data**.
  - Partition the data into a **70%** training set and **30%** testing set; set the seed to **1**. (Insert the **Partition** node between the **Var. File** and **Type** node.)
  - Run the **Auto Classifier** node.
  - Edit the generated **model nugget** (the yellow diamond).

- Which model has the highest **Overall Accuracy** on the testing set?
- In the **Graph** column, double-click the thumbnail graph for the **CHAID 1** model.

How many of the customers who responded positively to the campaign are identified as such?

- Because the distribution of the target field is skewed, you will balance the data, rerun the model, and evaluate it.
  - Scroll to the comment that reads **Balancing data**.
  - Run the **Distribution** node that is attached to the **Var. File** node.
  - In the **Distribution** output window: From the **Generate** menu, click **Balance Node (reduce)**.
  - Insert the generated **Balance** node between the **Partition** node and the **Type** node, and see to it that all responders pass through the **Balance** node, and 5% (a proportion of 0.05) of the non-responders.
    - Note: A Balance node downstream from a Partition node will only balance the training set, not the testing set. Therefore, the testing set (which needs to represent the original dataset) can be used to establish the accuracy of the models.
  - To ensure that the same records are used when you would run the **Auto Classifier** multiple times, cache the data on the **Balance** node.
  - Run the **Auto Classifier** node.

  Did the overall accuracy on the testing set improve? What is your conclusion?
- Exit IBM SPSS Modeler without saving anything.

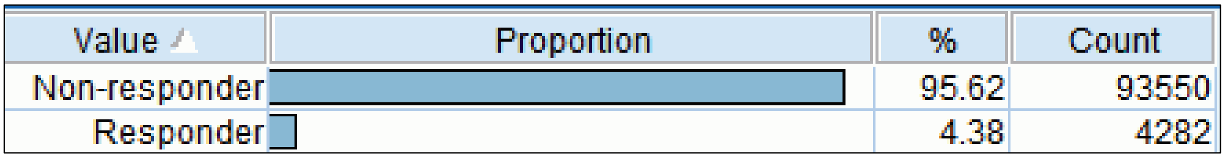
For more information about where to work and the exercise results, refer to the Task and Results section that follows. If you need more information to complete a task, refer to earlier demonstrations for detailed steps.

Exercise 1:
Tasks and Results

Task 1. Import, instantiate, and examine the data.

- Open **unit_4_exercise_1_start.str**, located in the **C:\Training\0A058\Start** folder.
- Scroll to the comment that reads **Sampling and balancing data**.
- Run the **Table** node to examine the data.
- Run the **Distribution** node.

  The results appear as follows:



Value	Proportion	%	Count
Non-responder		95.62	93550
Responder		4.38	4282

Only 4.38% responded to the campaign.

Task 2. Draw a random sample of 10%.

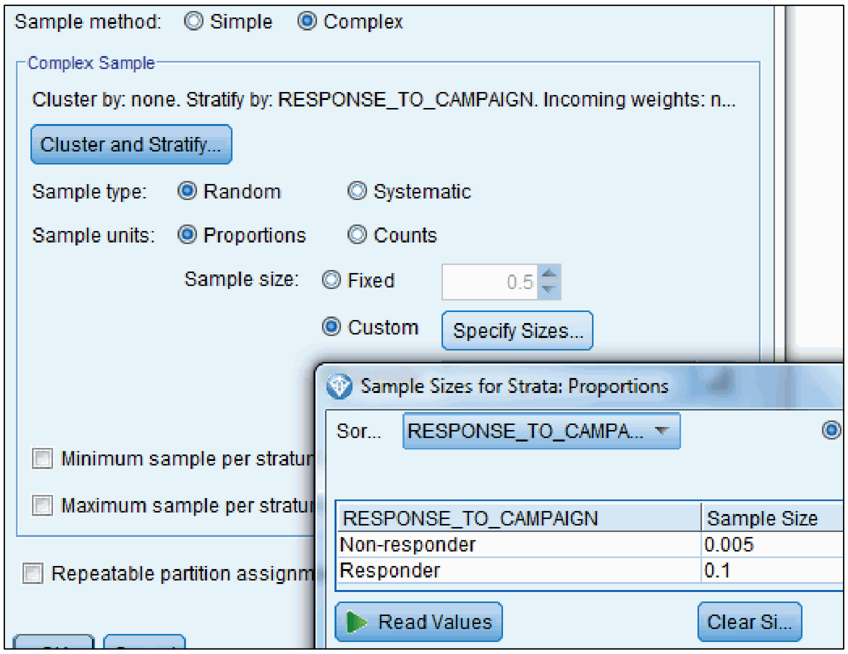
- From the **Record Ops** palette, add a **Sample** node downstream from the **Type** node.
- Edit the **Sample** node:
  - Select the **Random %** option.
  - Set the value to **10**.
  - For **Repeatable partition assignment**, set the value to **1**.
- From the **Output** palette, add a **Table** downstream from the **Sample** node, and then run it.

  The Table output shows that 10,010 records are sampled, which is (10,010 / 97,832) * 100 = 10.2318% of all records, approximately 10%. The Random r% will never sample exactly r% because each record will have an r/100 probability of being passed through the Sample node, making the outcome unpredictable for each record, and therefore the total number of records that will pass through the node cannot be predicted exactly.

Task 3. Draw a stratified sample.

- From the **Record Ops** palette, add a **Sample** node downstream from the **Type** node.
- Edit the **Sample** node:
  - For **Sample** method, select **Complex**.
  - Click **Cluster and Stratify**.
    - For **Stratify by,** select **RESPONSE_TO_CAMPAIGN**.
  - For **Sample size**, select **Custom.**
  - Click **Specify Sizes**.
    - Click **Read Values** to populate the categories.
    - For **Non-responder**, specify **0.005**.
    - For **Responder** specify **0.1**.

  The results appear as follows:



Sample method: ○ Simple [✓ Complex]

Complex Sample

Cluster by: none. Stratify by: RESPONSE_TO_CAMPAIGN. Incoming weights: n...

Sample type: [✓ Random] ○ Systematic

Sample units: [✓ Proportions] ○ Counts

Sample size: ○ Fixed 0.5 [✓ Custom]

☐ Minimum sample per stratur

☐ Maximum sample per stratur

☐ Repeatable partition assignm

Sample Sizes for Strata: Proportions

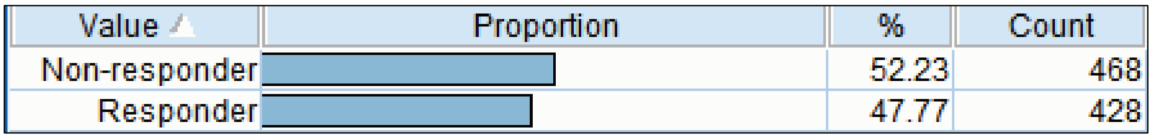
Sor... RESPONSE_TO_CAMPA...

RESPONSE_TO_CAMPAIGN	Sample Size
Non-responder	0.005
Responder	0.1

- For **Repeatable partition assignment**, set the **Seed** to **1**.

- From the **Graphs** palette, add a **Distribution** node downstream from the **Sample** node.
- Edit the **Distribution** node:
  - For **Field**, select **RESPONSE_TO_CAMPAIGN**.
  - Run the **Distribution** node.

    The results appear as follows:



Value	Proportion	%	Count
Non-responder		52.23	468
Responder		47.77	428

The number of responders and non-responders is approximately the same.

Task 4. Partition the data into a training set and testing set, and set the roles for modeling.

- Scroll to the comment that reads **Partitioning the data**.
- From the **Field Ops** palette, insert a **Partition** node between the **Var. File** node and Type node.
- Edit the **Partition** node:
  - For **Training partition size**, type **70**.
  - For **Testing partition size**, type **30**.
  - For **Repeatable partition assignment**, type **1**.
- Run the **Auto Classifier** node.
- Edit the generated **model nugget** (the yellow diamond).

  CHAID is the most accurate model on the testing set. (Actually, all three models have the same accuracy.)
- In the **Graph** column, double-click the thumbnail graph for the **CHAID 1** model.

  All customers who actually responded positively to the campaign are predicted to not respond positively. (Actually, all models predict that a customer will not respond positively to the campaign.)

Task 5. Use balancing to build models.

- Scroll to the comment that reads **Balancing data**.
- Run the **Distribution** node that is attached to the **Var. File** node.
- In the **Distribution** output window: From the **Generate** menu, click **Balance Node (reduce)**.
- Insert the generated **Balance** node (in the upper left corner of the stream canvas) between the **Partition** node and the **Type** node.
- Edit the generated **Balance** node and set the value for **Factor** to **0.05** for non-responders.
- Right-click the **Balance** node, click **Cache**, and then click **Enable**.
- Run the **Auto Classifier** node.
- Edit the generated **model nugget**.

  Although the accuracy of the best model is less than that of the accuracy of the model built on unbalanced, the best model now identifies customers who responded positively to the campaign.

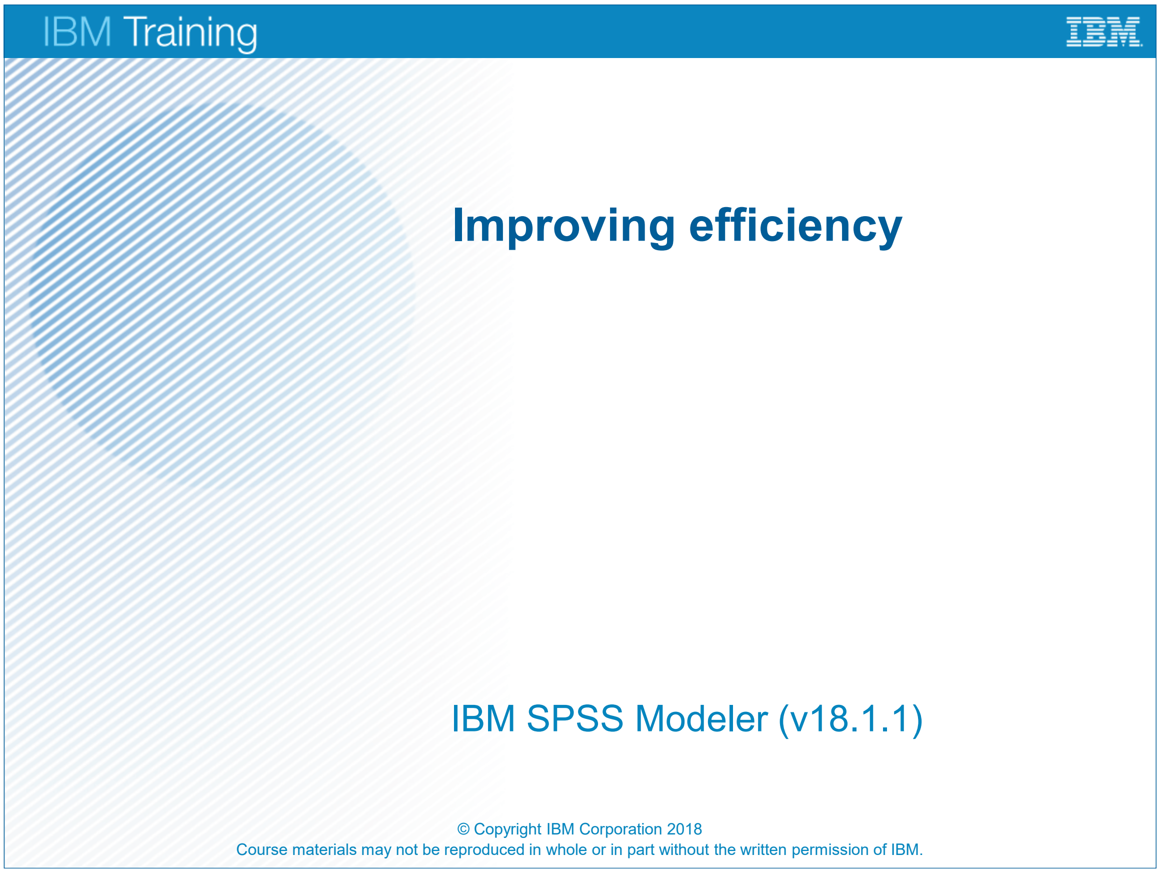
  Note: Your results may differ from those presented here because of the random selection of records in the Balance node.

Task 6. Exit IBM SPSS Modeler.

- From the **File** menu, click **Exit**, and then exit **IBM SPSS Modeler** without saving.

You will find the solution results in **C:\Training\0A058\Solutions** folder.

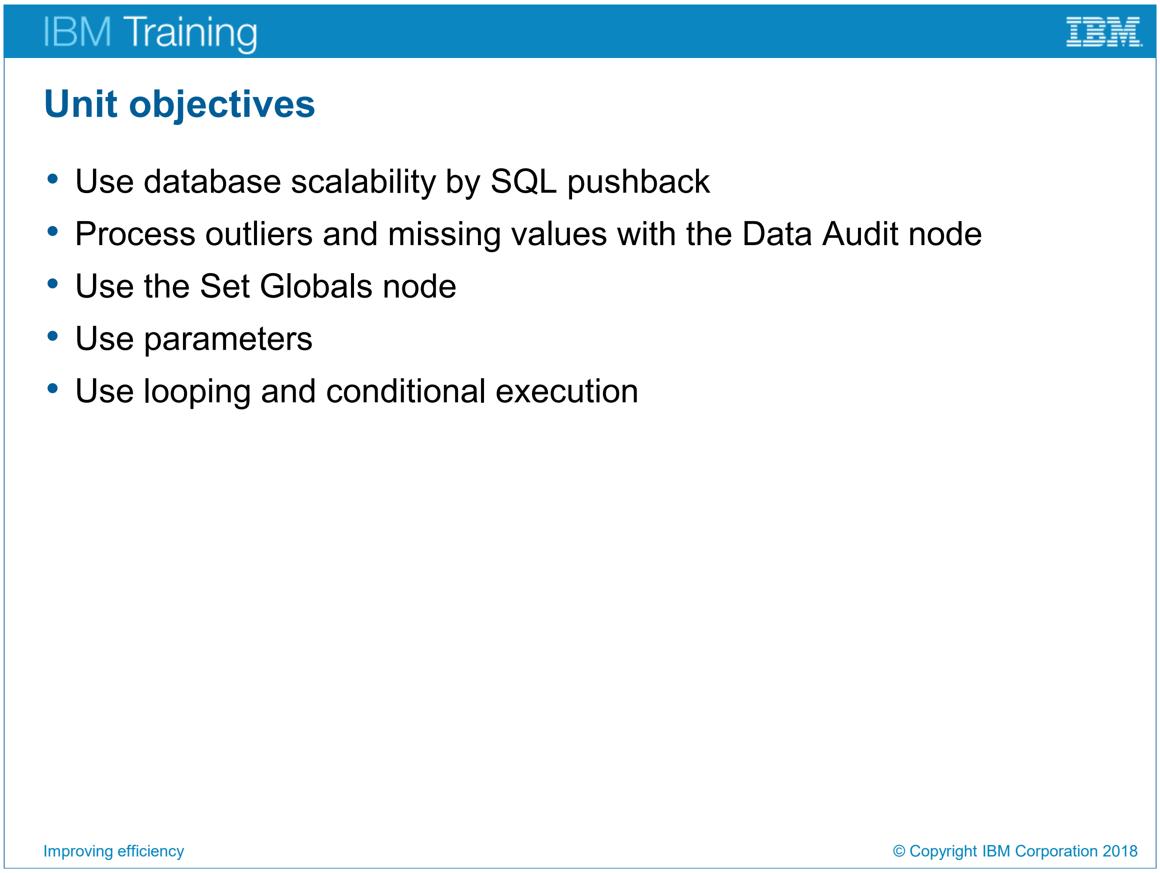
Unit 5 Improving efficiency



IBM Training

Improving efficiency

IBM SPSS Modeler (v18.1.1)



IBM Training

Unit objectives

- Use database scalability by SQL pushback
- Process outliers and missing values with the Data Audit node
- Use the Set Globals node
- Use parameters
- Use looping and conditional execution

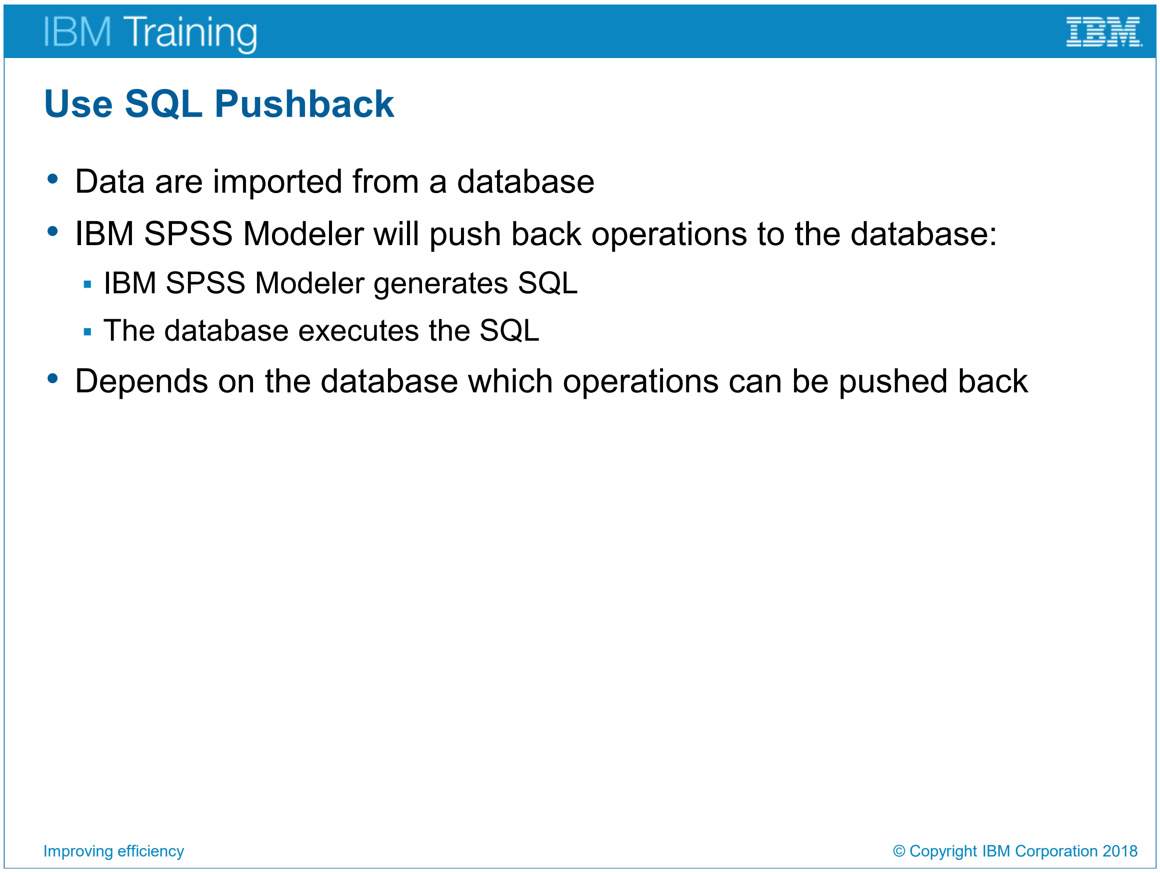
Improving efficiency

© Copyright IBM Corporation 2018

Unit objectives

Before reviewing this unit, you should be familiar with:

- working with IBM SPSS Modeler (streams, nodes, SuperNodes, palettes)
- importing data (Var. File node)
- defining measurement levels, roles, blanks, and instantiating data (Type node)
- selecting fields and records (Filter node, Select node)
- CLEM and the Expression Builder
- deriving and filling fields (Derive node, Filler node)
- running a model and examining the generated model nugget
- examining the data with graphs (Distribution node, Histogram node) and tables (Table node, Data Audit node, Statistics node, Matrix node, Means node)



IBM Training

Use SQL Pushback

- Data are imported from a database
- IBM SPSS Modeler will push back operations to the database:
  - IBM SPSS Modeler generates SQL
  - The database executes the SQL
- Depends on the database which operations can be pushed back

Improving efficiency

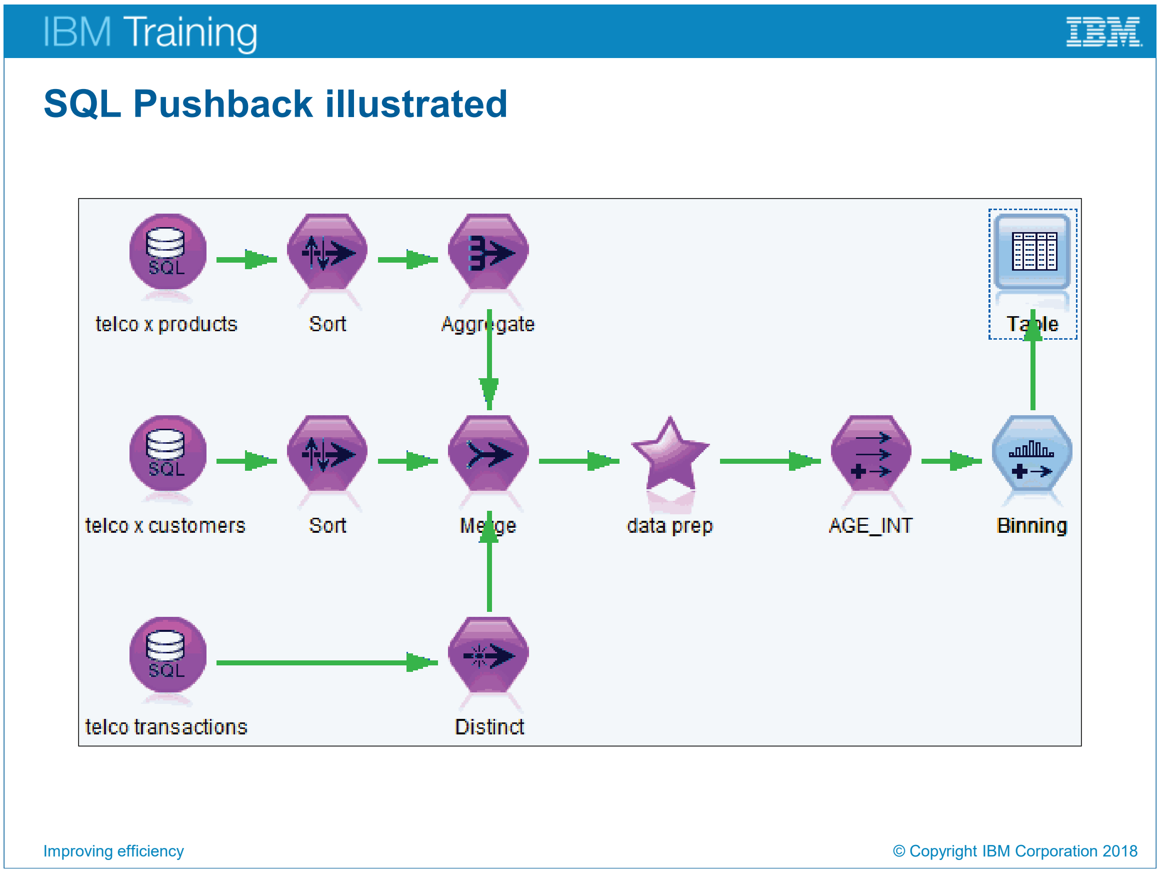
© Copyright IBM Corporation 2018

Use SQL Pushback

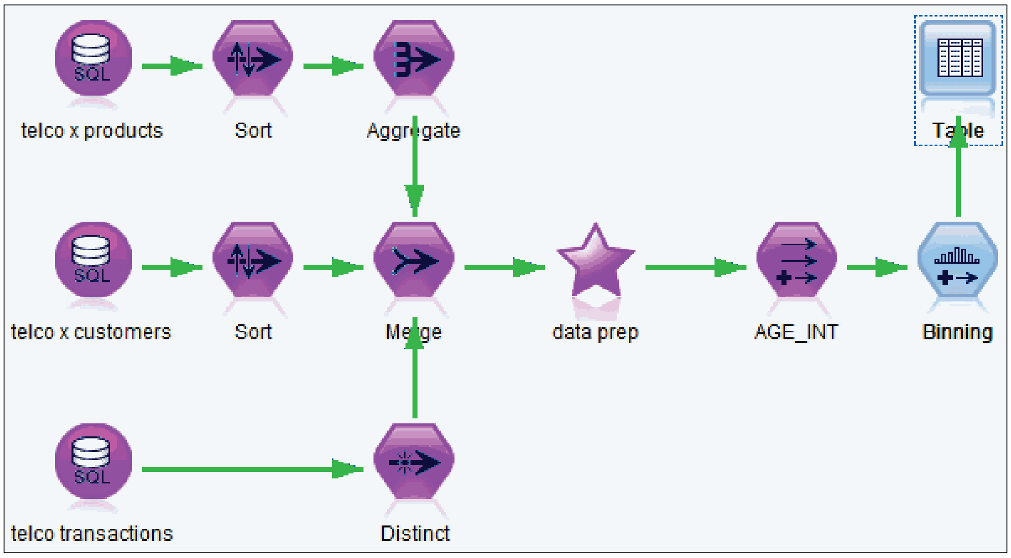
Many data scientists will access data directly from a database rather than work on extracts that have been written out to separate files. When you work with database sources, one of the most powerful capabilities of IBM SPSS Modeler is the ability to perform data preparation operations directly in the database. By generating SQL code that is pushed back to the database for execution, many operations, such as deriving fields, merging, aggregating, sampling, and sorting can be performed in the database itself rather than that they are executed by IBM SPSS Modeler. This functionality is called SQL pushback.

Not all operations can be pushed back to the database and it depends on the database that is in use which operations are supported. When IBM SPSS Modeler encounters a node that cannot be compiled to SQL, IBM SPSS Modeler will extract the data from the database and process the data from there.

When you import data from a database, you can make use of various database aggregate functions in the Derive, Select node and Aggregate node. For example, you can use these functions to compute a moving average using the appropriate database function to ensure SQL pushback.


IBM Training

SQL Pushback illustrated



Improving efficiency

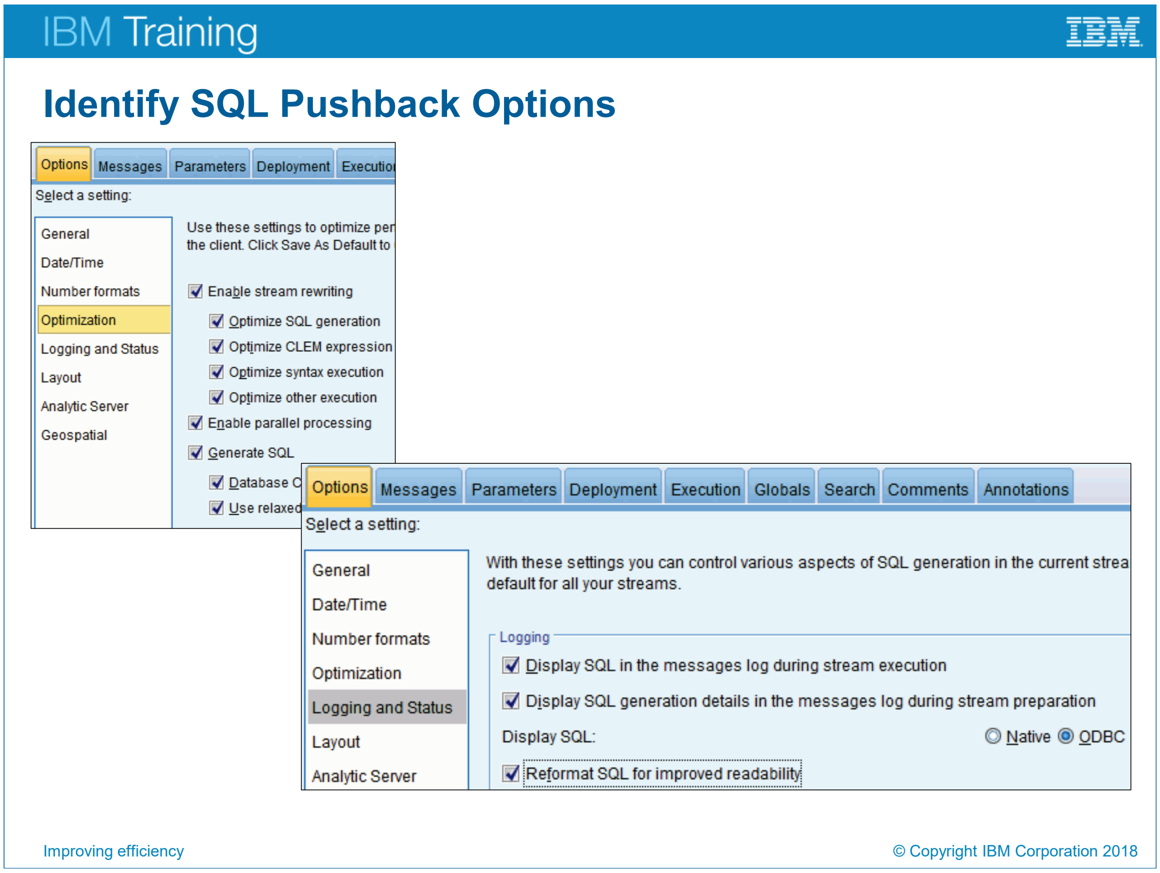
© Copyright IBM Corporation 2018

SQL Pushback illustrated

When a stream is executed and there is SQL pushback, the nodes whose functionality is being pushed back to the database will be highlighted in purple.

The stream depicted on this slide reads data from three database sources, sorts the data, merges the data, runs some data preparation nodes (encapsulated in the SuperNode named data prep), derives a field AGE_INT, runs a Binning node, and ends with a Table. When the stream is executed, all the nodes that can be pushed back to the database turn purple. Here, all nodes turn purple, except for the Binning node and the Table node. For example, IBM SPSS Modeler outsourced creating AGE_INT to the database (by generating SQL code and having that code executed by the database). By comparison, binning fields was not supported by this database. Thus, IBM SPSS Modeler has extracted the data from the database after the AGE_INT field was created in the database, and will proceed with the Binning operation, using IBM SPSS Modeler's native binning algorithms.

When this stream is executed using another database, it is not guaranteed that the same nodes turn purple. The operations that are pushed back depend on the database in use.



IBM Training

Identify SQL Pushback Options

Options | Messages | Parameters | Deployment | Executio

Select a setting:

- General
- Date/Time
- Number formats
- Optimization
- Logging and Status
- Layout
- Analytic Server
- Geospatial

Use these settings to optimize per
the client. Click Save As Default to

- ✓ Enable stream rewriting
  - ✓ Optimize SQL generation
  - ✓ Optimize CLEM expression
  - ✓ Optimize syntax execution
  - ✓ Optimize other execution
- ✓ Enable parallel processing
- ✓ Generate SQL
  - ✓ Database C
  - ✓ Use relaxed

Options | Messages | Parameters | Deployment | Execution | Globals | Search | Comments | Annotations

Select a setting:

- General
- Date/Time
- Number formats
- Optimization
- Logging and Status
- Layout
- Analytic Server

With these settings you can control various aspects of SQL generation in the current strea
default for all your streams.

Logging

- ✓ Display SQL in the messages log during stream execution
- ✓ Display SQL generation details in the messages log during stream preparation

Display SQL: Native [✓ ODBC]

- ✓ Reformat SQL for improved readability

Improving efficiency

© Copyright IBM Corporation 2018

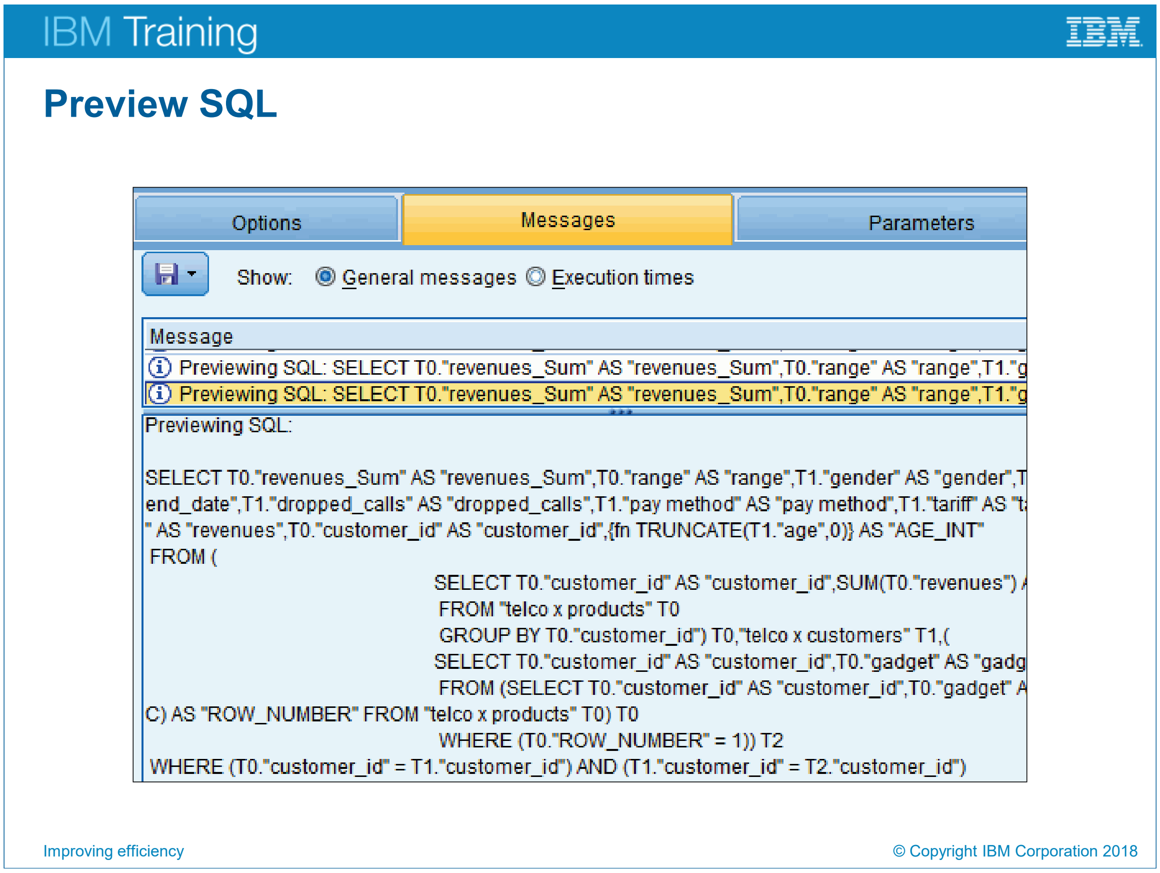
Identify SQL Pushback Options

Options for SQL Pushback can be set on the Options tab in the Stream Properties dialog box (accessed via Tools\Stream Properties\Options in IBM SPSS Modeler's main menu).

The Generate SQL option must be enabled to facilitate SQL pushback. With the Optimize SQL generation option enabled, IBM SPSS Modeler will look for a node that cannot be compiled into SQL code and then look downstream from that node to check for nodes that can be rendered into SQL code. IBM SPSS Modeler will then reorder the nodes, provided that that does not affect the results.

To further improve efficiency, you can cache the data on a midstream node, in which case IBM SPSS Modeler will try to store the data in a temporary database table rather than in an IBM SPSS Statistics.sav file (the regular format for the cache).

The SQL code that was generated can be displayed in the messages log. It is recommended to let IBM SPSS Modeler reformat the code to improve the readability. Also, you can choose whether the SQL code is displayed in the database's native SQL functions, or in standard ODBC functions.



Preview SQL

```
Previewing SQL: SELECT T0."revenues_Sum" AS "revenues_Sum",T0."range" AS "range",T1."g
Previewing SQL: SELECT T0."revenues_Sum" AS "revenues_Sum",T0."range" AS "range",T1."g
Previewing SQL:

SELECT T0."revenues_Sum" AS "revenues_Sum",T0."range" AS "range",T1."gender" AS "gender",T
end_date",T1."dropped_calls" AS "dropped_calls",T1."pay method" AS "pay method",T1."tariff" AS "t
" AS "revenues",T0."customer_id" AS "customer_id",{fn TRUNCATE(T1."age",0)} AS "AGE_INT"
 FROM (
                                SELECT T0."customer_id" AS "customer_id",SUM(T0."revenues") A
                                 FROM "telco x products" T0
                                 GROUP BY T0."customer_id") T0,"telco x customers" T1,(
                                SELECT T0."customer_id" AS "customer_id",T0."gadget" AS "gadg
                                 FROM (SELECT T0."customer_id" AS "customer_id",T0."gadget" A
C) AS "ROW_NUMBER" FROM "telco x products" T0) T0
                                 WHERE (T0."ROW_NUMBER" = 1)) T2
 WHERE (T0."customer_id" = T1."customer_id") AND (T1."customer_id" = T2."customer_id")
```

Improving efficiency © Copyright IBM Corporation 2018

Preview SQL

It is useful to preview the SQL prior to execution. When you click the Preview Run the stream will not be executed, but the SQL code will be generated and you can examine it in the Messages log.

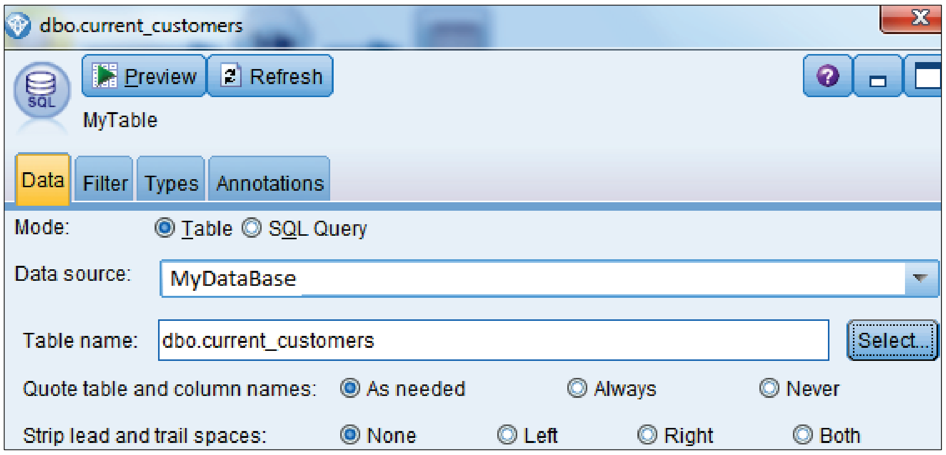
This slide shows a section of the Messages log when the Preview Run button was clicked. The SQL code that was generated by IBM SPSS Modeler is in the lower part of the window.

You can also execute the stream, and then examine the SQL code that was actually fired at the database. Again, you will find this code in the Messages log.

Note: You can access the Messages log via Tools in the main menu, Stream Properties, Options, Messages tab.

IBM Training

SQL Pushback illustrated: The Database source node


dbo.current_customers

Preview Refresh

MyTable

Data | Filter | Types | Annotations

Mode: [✓ Table] SQL Query

Data source: MyDataBase

Table name: dbo.current_customers Select...

Quote table and column names: [✓ As needed] Always Never

Strip lead and trail spaces: [✓ None] Left Right Both

Improving efficiency

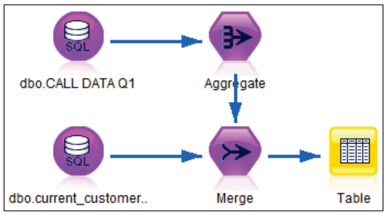
SQL Pushback illustrated: The Database source node

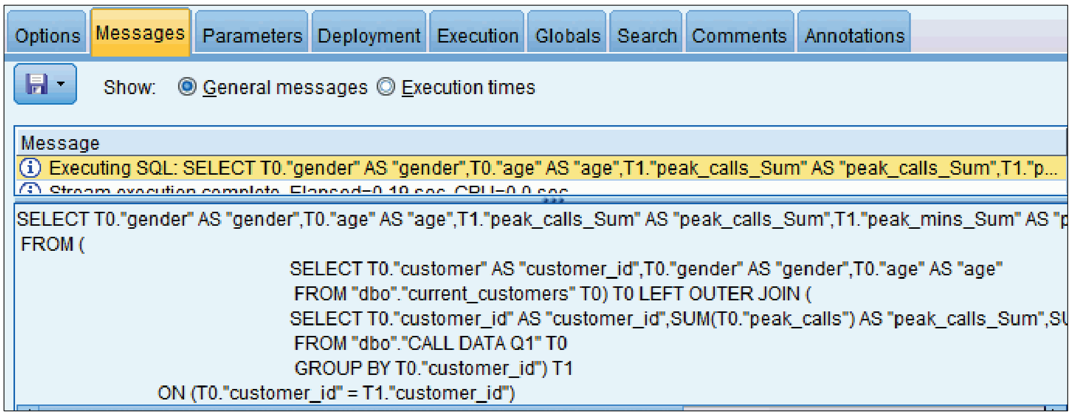
This slide shows the Database source node dialog box. Only two specifications are needed: the Data source name and the table that you want to select from that database.

You can use the Filter, Types, and Annotations tab as with any other data source node.

IBM Training IBM

SQL Pushback illustrated: Executed stream and messages





```
Message
ⓘ Executing SQL: SELECT T0."gender" AS "gender",T0."age" AS "age",T1."peak_calls_Sum" AS "peak_calls_Sum",T1."p...
ⓘ Stream execution complete. Elapsed=0.10 sec, CPU=0.0 sec
```

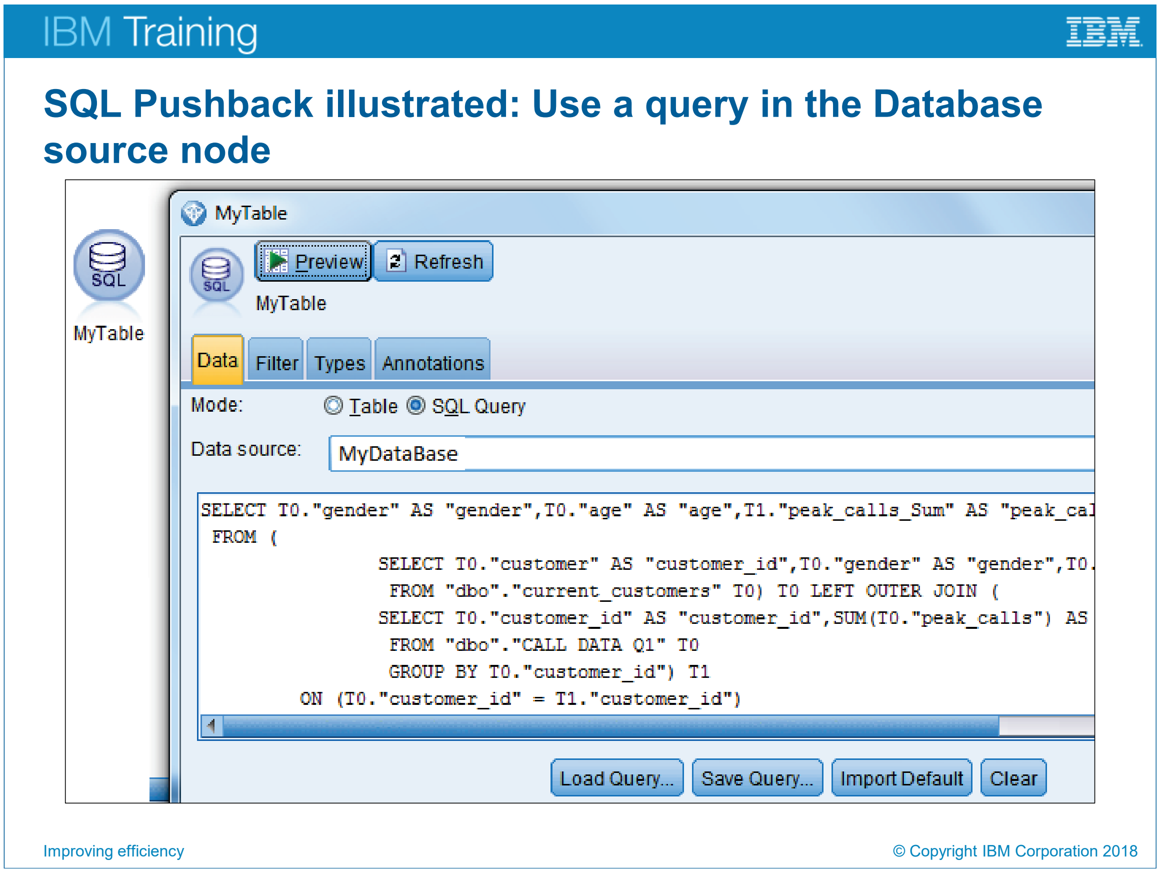
```
SELECT T0."gender" AS "gender",T0."age" AS "age",T1."peak_calls_Sum" AS "peak_calls_Sum",T1."peak_mins_Sum" AS "p
 FROM (
                                SELECT T0."customer" AS "customer_id",T0."gender" AS "gender",T0."age" AS "age"
                                 FROM "dbo"."current_customers" T0) T0 LEFT OUTER JOIN (
                                SELECT T0."customer_id" AS "customer_id",SUM(T0."peak_calls") AS "peak_calls_Sum",SU
                                 FROM "dbo"."CALL DATA Q1" T0
                                 GROUP BY T0."customer_id") T1
                ON (T0."customer_id" = T1."customer_id")
```

Improving efficiency © Copyright IBM Corporation 2018

SQL Pushback illustrated: Executed stream and messages

This slide presents a stream with only a few nodes. Because aggregating and merging is supported by the database in use, SQL code is generated for the aggregate and merge operations and IBM SPSS Modeler runs the generated SQL code against the database. The Messages log will display the generated SQL.

This example illustrates that even a simple stream, with a few nodes and processing only a few fields, will generate many lines of SQL. For those familiar with SQL, however, the code will have no secrets.


IBM Training

SQL Pushback illustrated: Use a query in the Database source node

Mode: Table [✓ SQL Query]

Data source: MyDataBase

```
SELECT T0."gender" AS "gender",T0."age" AS "age",T1."peak_calls_Sum" AS "peak_cal
 FROM (
               SELECT T0."customer" AS "customer_id",T0."gender" AS "gender",T0.
                FROM "dbo"."current_customers" T0) T0 LEFT OUTER JOIN (
               SELECT T0."customer_id" AS "customer_id",SUM(T0."peak_calls") AS
                FROM "dbo"."CALL DATA Q1" T0
                GROUP BY T0."customer_id") T1
        ON (T0."customer_id" = T1."customer_id")
```

Improving efficiency

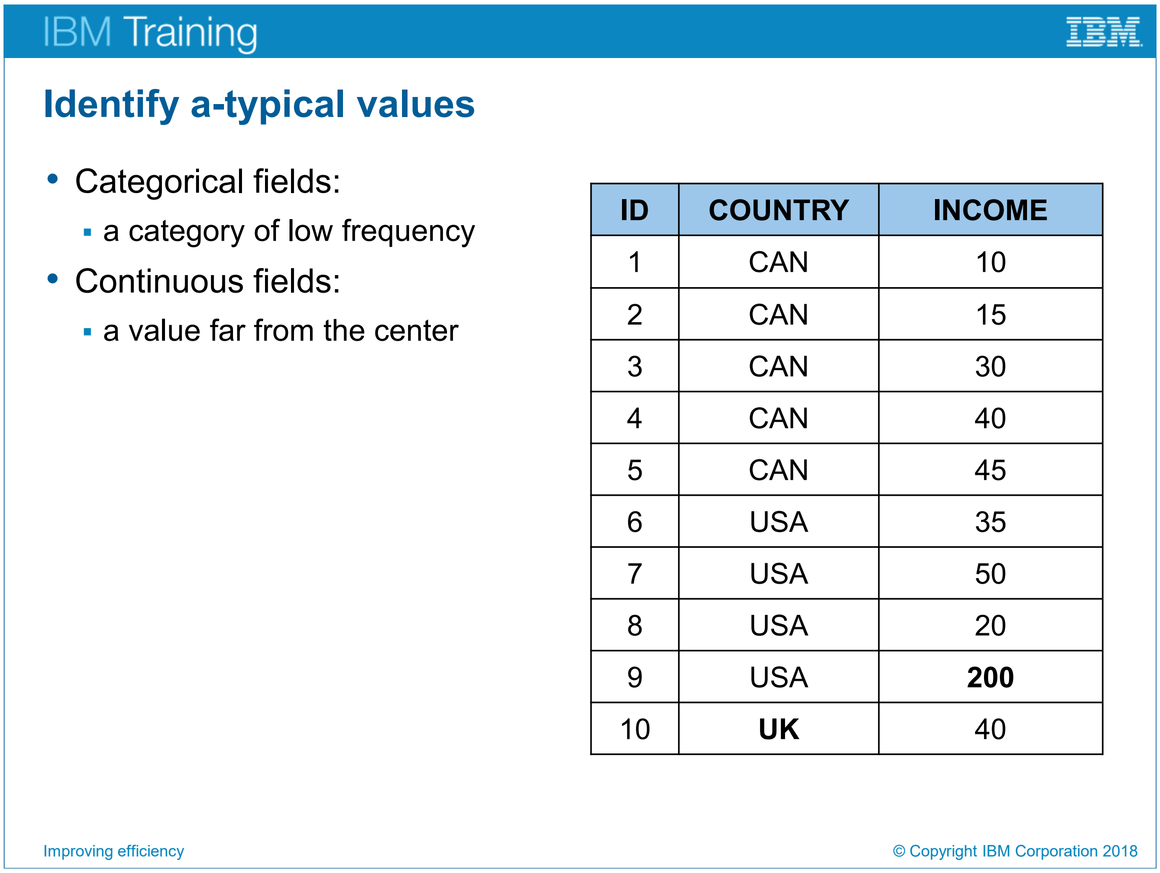
© Copyright IBM Corporation 2018

SQL Pushback illustrated: Use a query in the Database source node

You can copy and paste the generated SQL code to other applications. You can also copy it to IBM SPSS Modeler itself, using the SQL query option in the Database source node. In this case you can dismiss the nodes from where the SQL code was generated, making the stream simpler. The disadvantage, however, is that changes (for example, including other statistics in the aggregate operation) are not so easily made to the SQL code as they are modified in the original stream.

The Database source node lets you also import a query from elsewhere, by clicking Load Query.

Note: SQL pushback is not demonstrated in this course because it depends on the database in use. For the latest information on which databases are supported and tested for use with IBM SPSS Modeler 18.1, refer to the corporate Support site at http://www.ibm.com/support.



IBM Training

Identify a-typical values

- Categorical fields:
  - a category of low frequency
- Continuous fields:
  - a value far from the center

ID	COUNTRY	INCOME
1	CAN	10
2	CAN	15
3	CAN	30
4	CAN	40
5	CAN	45
6	USA	35
7	USA	50
8	USA	20
9	USA	**200**
10	**UK**	40

Improving efficiency

© Copyright IBM Corporation 2018

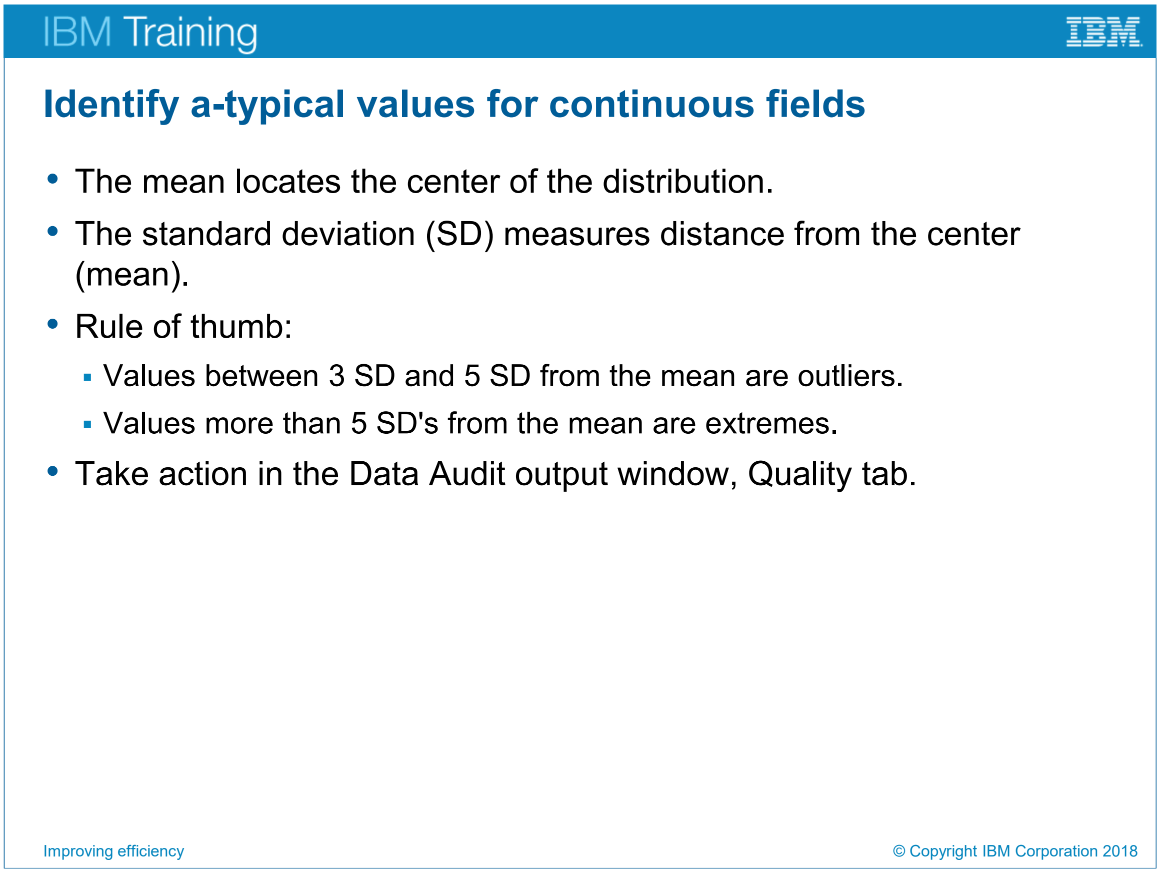
Identify a-typical values

In any data file there may be values on one or more fields that are unusual, or a-typical. What a-typical values are depends on the measurement level of the fields. For a categorical field, a category with only a few records in it might be considered as a-typical. For a continuous field, an a-typical value would be a value far of the center of the distribution. This slide gives an example for a nominal and continuous field.

A-typical values might make it more difficult to build models. For example, classical statistical techniques like Linear Regression are sensitive to a-typical values and the smaller the number of records, the higher the impact these values. Also, outliers can affect data science techniques such as neural networks. Some types of models are less affected by extreme data values, such as CHAID.

There are several ways to define a-typical values. They can be defined by looking at a single field or by looking at several fields simultaneously. In the latter case you can use the Anomaly node. Refer to the *Clustering and Association Modeling Using IBM SPSS Modeler* course for a presentation of the Anomaly node.

In this unit, a-typical values are defined by looking at a single field, in particular a-typical values of a continuous field.



IBM Training

Identify a-typical values for continuous fields

- The mean locates the center of the distribution.
- The standard deviation (SD) measures distance from the center (mean).
- Rule of thumb:
  - Values between 3 SD and 5 SD from the mean are outliers.
  - Values more than 5 SD's from the mean are extremes.
- Take action in the Data Audit output window, Quality tab.

Improving efficiency

© Copyright IBM Corporation 2018

Identify a-typical values for continuous fields

The center of a distribution is usually measured by the mean. How far away a value is from the mean is typically expressed in terms of standard deviations. For example, if a record has an income of 3,000, and the mean income and standard deviation are 2,000 and 500 respectively, then that income is (3,000 - 2,000)/500= 2 standard deviations above the mean income. An income of 1,500 is 1 standard deviation below the mean. Scores computed in this way are called standardized scores or z-scores in the statistical literature. A standardized score expressed how many standard deviations the score is away from the mean.

Typically, a threshold of 3 is used to mark outliers. An outlier is said to be extreme if it is more than 5 standard deviations away from the mean. The motivation for using these thresholds comes from the normal distribution. Refer to text books on statistics for more information on mean, standard deviation, normal distribution and standardized scores.

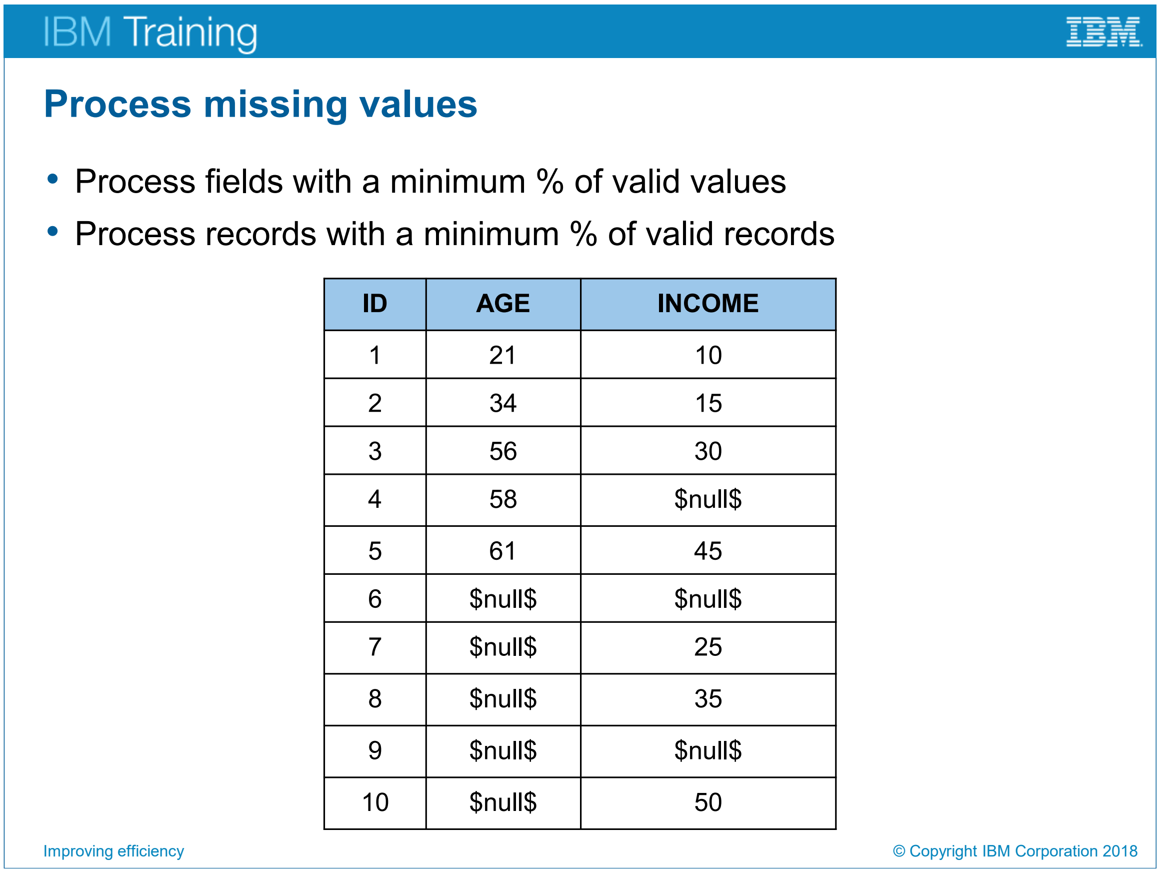
The mean and standard deviation have the important property that when a field's unit of measurement changes by a linear transformation, mean and standard deviation change by the same factor. Consequently, the relative position of a score does not change when the unit of measurement changes.

You can deal with outliers and extremes by running a Data Audit node and then take a specific action on the Quality tab in the Data Audit output window. You can choose to:

- Coerce outliers or extremes to the lower limit (mean - 3 * SD) or upper limit (mean - 3 * SD). The disadvantage of this approach is that all the values for these cases will be the same, causing a pile-up of records at one value.
- Discard records with outliers or extremes from the dataset. If their numbers are very small, this might be an acceptable strategy if one presumes that outliers and extremes values are of no special interest.
- Nullify outliers and extremes. Recode outliers and extremes to $null$; depending on the modeling technique, these records will be included.
- Coerce outliers, and either discard records with extreme values or nullify them.

You can try models with and without the modified fields to examine the effect of the action that was taken.

Note: The mean and the standard deviation themselves are sensitive to extreme values, especially in small datasets. For example, in a series of data of 1, 2, 3, 4, 5, 50, the mean (10.8) and standard deviation (19.2) will be inflated because of the value 50, and consequently the value 50 will not be marked as an outlier. Rather than using mean and standard deviation to measure center and spread, robust statistics such as median and interquartile range are in use to define outliers and extremes. A discussion of these measures is beyond the scope for this course. Refer to the online Help for more information.



IBM Training

Process missing values

- Process fields with a minimum % of valid values
- Process records with a minimum % of valid records

ID	AGE	INCOME
1	21	10
2	34	15
3	56	30
4	58	$null$
5	61	45
6	$null$	$null$
7	$null$	25
8	$null$	35
9	$null$	$null$
10	$null$	50

Improving efficiency

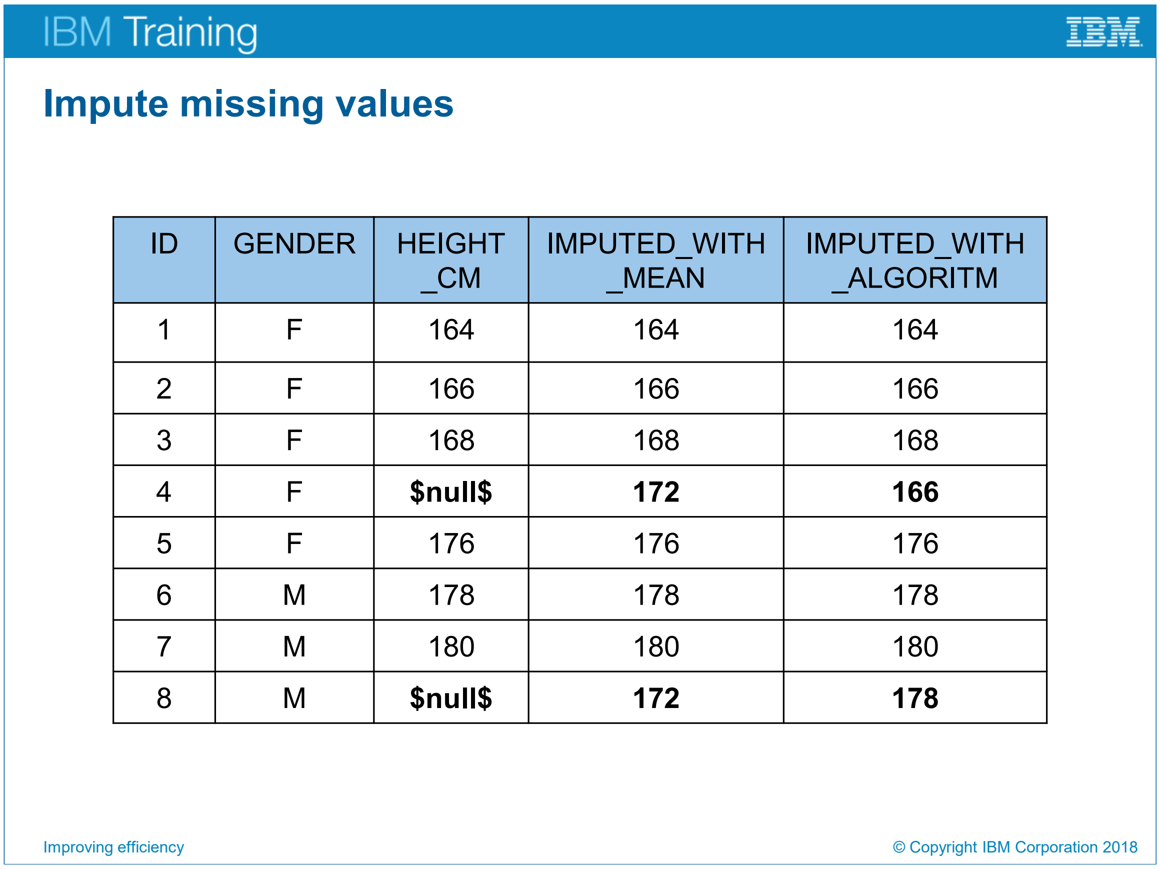
© Copyright IBM Corporation 2018

Process missing values

The Data Audit output window not only enables you to process outliers and extremes, it also provides options to deal with missing values. When you have executed the Data Audit node you can generate a SuperNode from the Data Audit output that processes missing values. The Generate menu in the Data Audit output, Quality tab, lets you filter fields that have a minimum percentage of valid values or select records that have a minimum percentage of valid values.

In the dataset depicted on this slide, when you specify that you want to retain only fields with at least 60% of valid values, you can generate a SuperNode that encapsulates a Filter node that will remove AGE from the dataset. Likewise, you can generate a SuperNode that contains a Select node that will retain records with, for example, at least 50% valid values.

How you handle missing data when you are developing a model has complications. For example, when you use only valid records to develop a model to predict an outcome, you cannot apply the model successfully to new data unless all the fields have non-missing data, which is not very likely. Unless the number of records with missing data is negligible, following this scenario would prevent you from making predictions on an appreciable fraction of records in new data.



IBM Training

Impute missing values

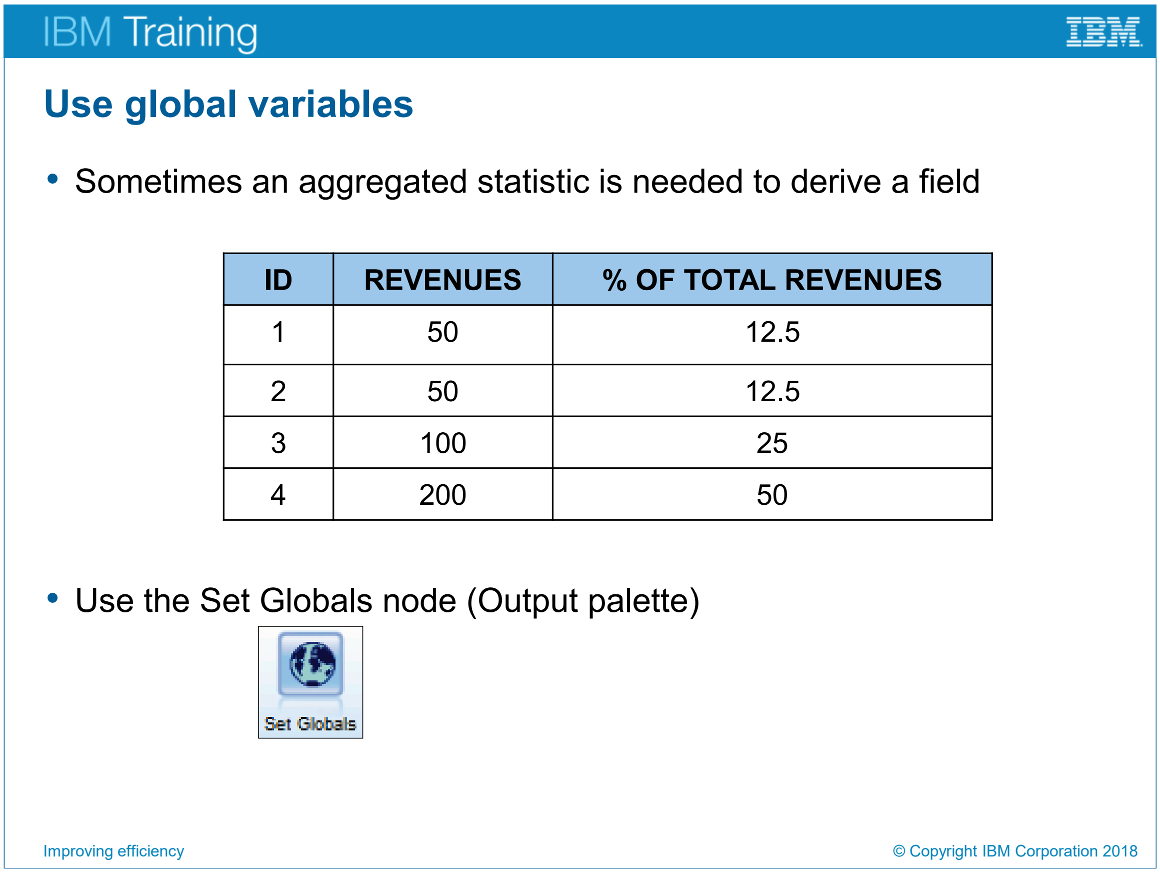
ID	GENDER	HEIGHT _CM	IMPUTED_WITH _MEAN	IMPUTED_WITH _ALGORITM
1	F	164	164	164
2	F	166	166	166
3	F	168	168	168
4	F	**$null$**	**172**	**166**
5	F	176	176	176
6	M	178	178	178
7	M	180	180	180
8	M	**$null$**	**172**	**178**

Improving efficiency © Copyright IBM Corporation 2018

Impute missing values

Rather than discarding fields or records you could decide to replace missing values with valid values, a technique which is called imputation. IBM SPSS Modeler enables you to replace missing values with a fixed value such as the mean, with a value from a certain distribution such as the normal distribution, or, the most sophisticated alternative, you can use the C&R Tree algorithm to replace missing values.

This slide presents two examples of imputation: when HEIGHT is missing it can be replaced with the overall mean or you can use the C&R Tree algorithm to impute values, which will result in different values for men and women.



IBM Training

Use global variables

- Sometimes an aggregated statistic is needed to derive a field

ID	REVENUES	% OF TOTAL REVENUES
1	50	12.5
2	50	12.5
3	100	25
4	200	50

- Use the Set Globals node (Output palette)

Set Globals

Improving efficiency

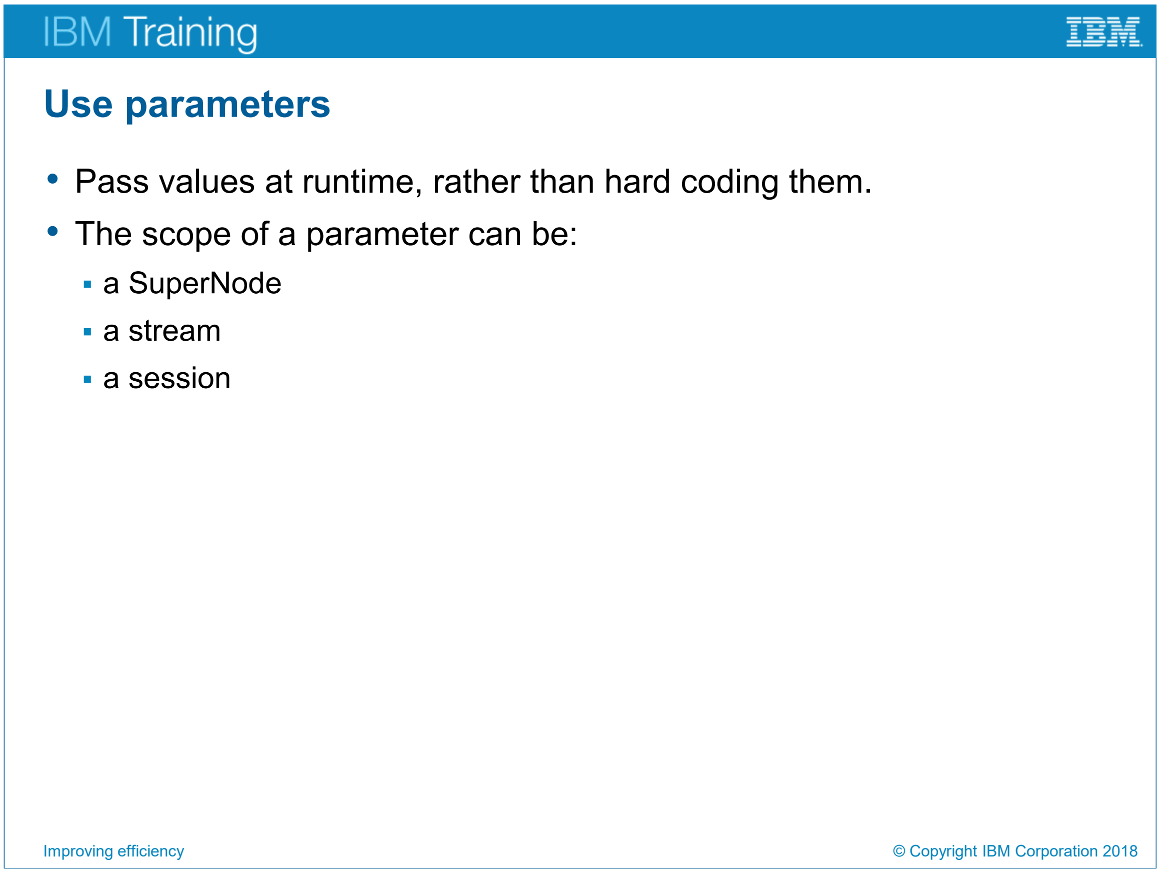
© Copyright IBM Corporation 2018

Use global variables

Sometimes you need an aggregated statistic to compute a field. An example is depicted on this slide. The dataset stores customers and their revenues. Think of a field that stores the share of the customer's revenues in the total revenues, as a percentage. In this example, the total revenues is 400, so for the first customer the result should be (50/400) * 100 = 12.5.

You need the sum of the revenues for this computation. Manually entering the sum is not an option. An alternative is to aggregate the data and to merge the aggregated dataset and the original dataset. This would add a new field to the data, with a value of 400 for each record. Thus, the dataset is expanded needlessly, especially if there are a huge number of records.

It is more efficient to use the Set Globals node, located in the Output palette. This node calculates the mean, standard deviation, minimum, maximum and sum, and stores them in memory. When you have executed the Set Globals node you can use the stored values in CLEM expressions.



IBM Training

Use parameters

- Pass values at runtime, rather than hard coding them.
- The scope of a parameter can be:
  - a SuperNode
  - a stream
  - a session

Improving efficiency

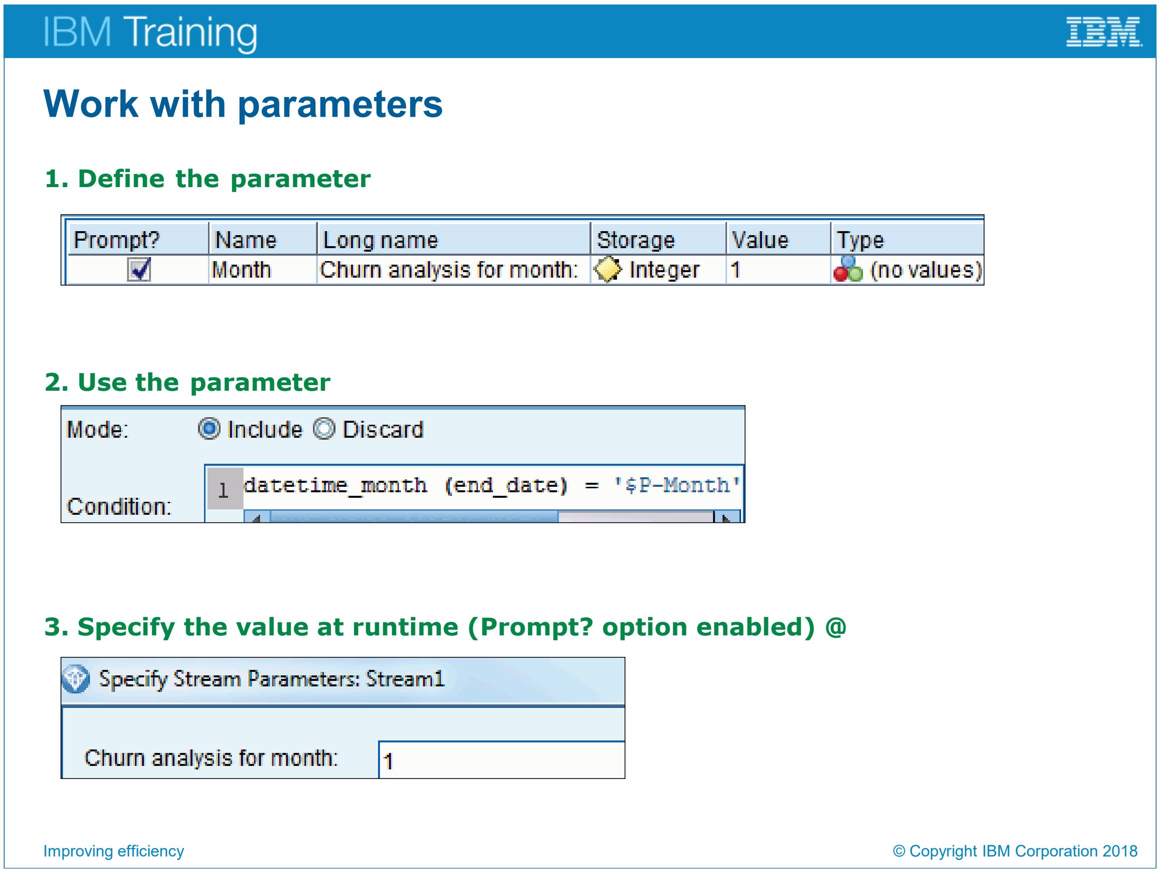
© Copyright IBM Corporation 2018

Use parameters

There are several situations where it is not efficient to enter the values themselves in CLEM expressions. Think of a stream that runs analyses for a certain group, with the name of the group hard coded in a Select node. When you want to run the analyses for another group, you have to edit the Select node and replace the name with the new name. It is more efficient that you will be prompted for the name of the group at run time.

Another situation is when you have several nodes using the same CLEM expression, with the same value(s) hard coded in it. If you need to replace these values you have to be very careful in replacing them all and not to forget one or more. It is more efficient to use a stand-in name for the value and replace that only once, if desired on a prompt at run time.

In these situations you can use parameters that substitute hard coded values. You can specify parameters for SuperNodes, streams, and sessions. SuperNode parameters have the smallest scope and can only be used within a specific SuperNode, whereas session parameters have the largest scope and are available to all streams in an IBM SPSS Modeler session.



IBM Training

Work with parameters

1. Define the parameter

Prompt?	Name	Long name	Storage	Value	Type
✓	Month	Churn analysis for month:	Integer	1	(no values)

2. Use the parameter

Mode: [✓ Include] Discard

Condition:
```
datetime_month (end_date) = '$P-Month'
```

3. Specify the value at runtime (Prompt? option enabled) @

Specify Stream Parameters: Stream1

Churn analysis for month: 1

Improving efficiency

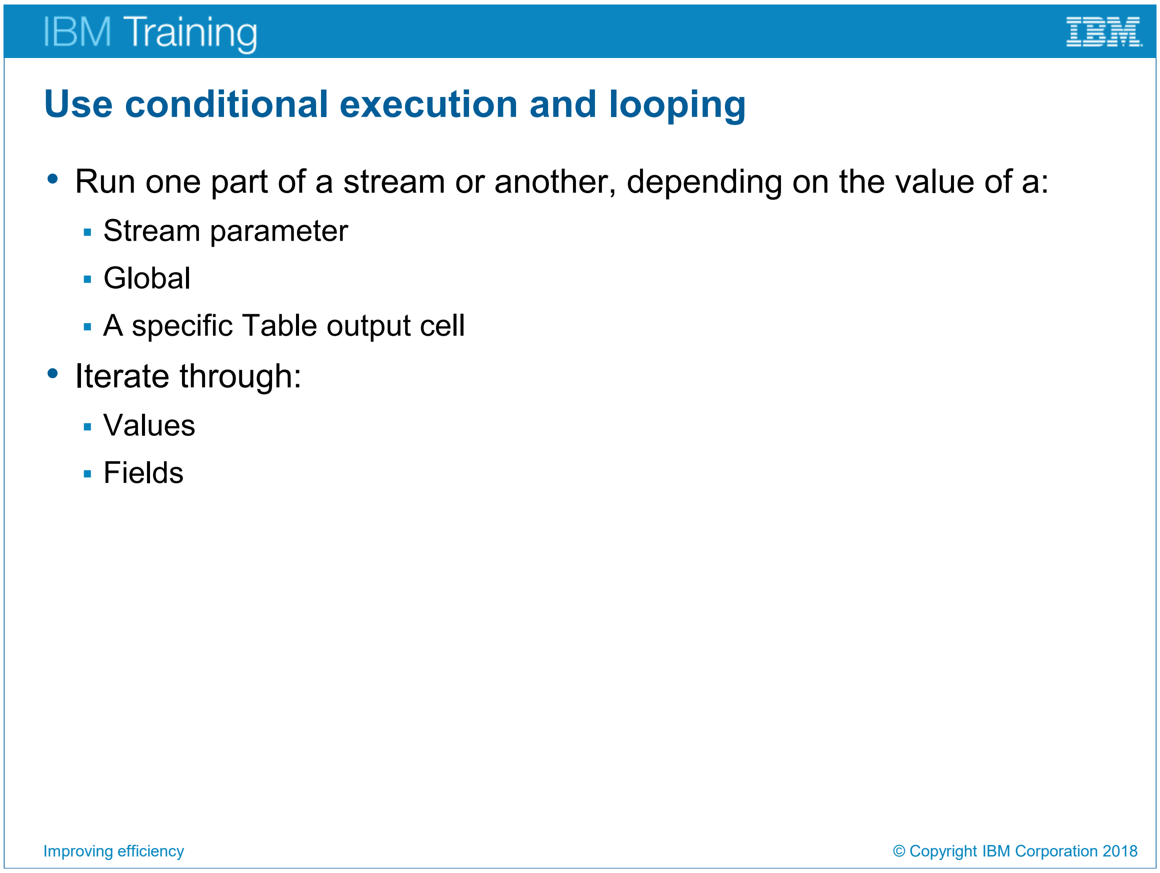
© Copyright IBM Corporation 2018

Work with parameters

SuperNode parameters can be accessed in the SuperNode dialog box by clicking the Define Parameter button. Stream and session parameters are defined by selecting Tools\Stream Properties\Parameters and Tools\Set Session Parameters in IBM SPSS Modeler's main menu, respectively.

Whether it is a SuperNode, stream or session parameter, the way that a parameter is defined is the same in all cases:

- Enable or disable a prompt: When this option is enabled, a prompt will be presented to the user when the stream execution is run.
- Name: The parameter will be represented in CLEM expressions by $P-<pname>, where <pname> is the name specified here.
- Long name: When the Prompt option is enabled, the text specified here will be presented to the user in the prompt.
- Storage, Value, Type: Specify the storage, the default value, and measurement level.When you have defined parameters, the next step is to replace the hard coded values with the parameter name. Finally, run the stream and specify the value at run time (when the Prompt option is enabled).



IBM Training

Use conditional execution and looping

- Run one part of a stream or another, depending on the value of a:
  - Stream parameter
  - Global
  - A specific Table output cell
- Iterate through:
  - Values
  - Fields

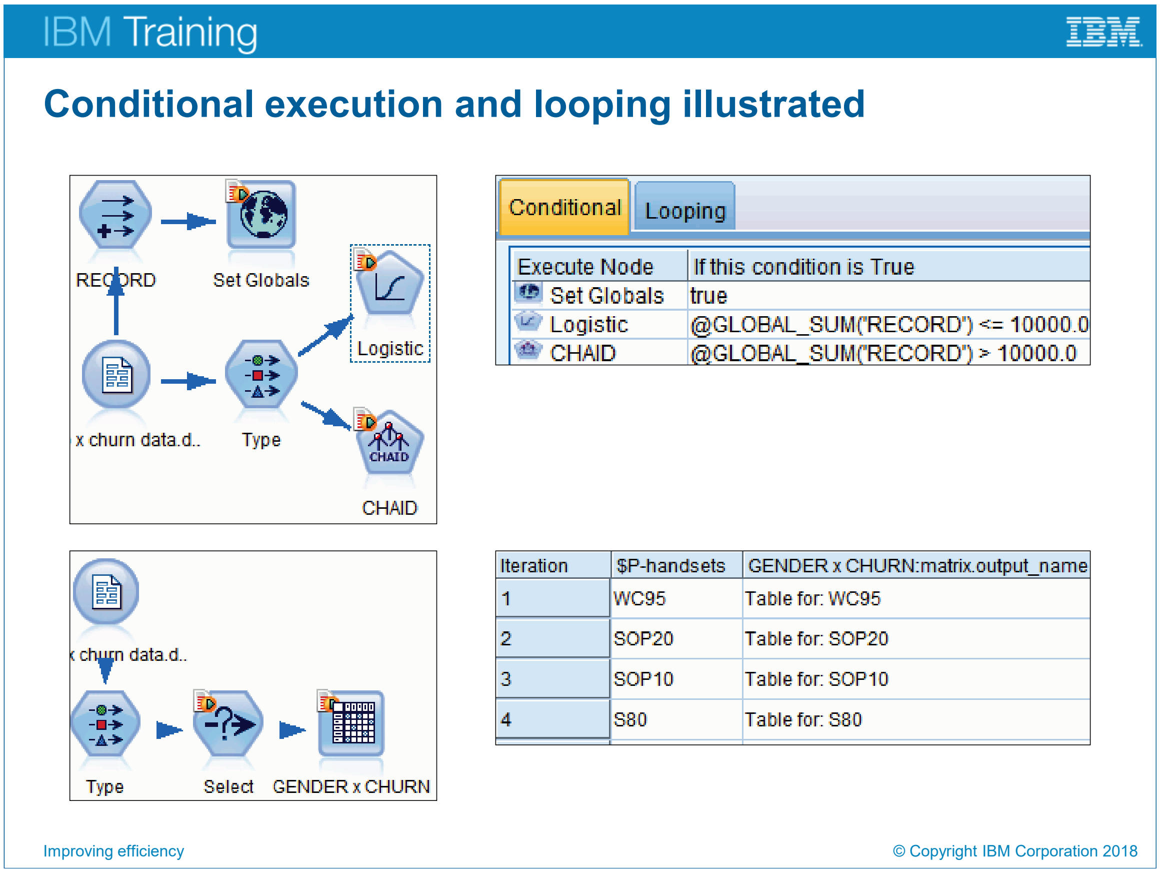
Improving efficiency

© Copyright IBM Corporation 2018

Use conditional execution and looping

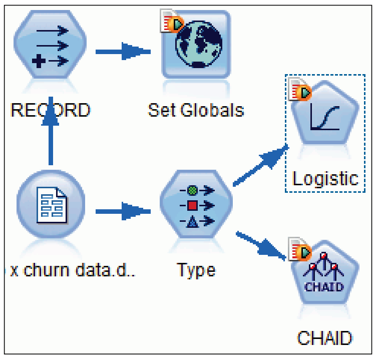
Suppose that you want to execute one branch of a stream when a certain condition is met, and another when the condition is not met. IBM SPSS Modeler provides you with the Conditional Execution feature, where you can base the condition on a stream parameter, a global variable, or a Table output cell.

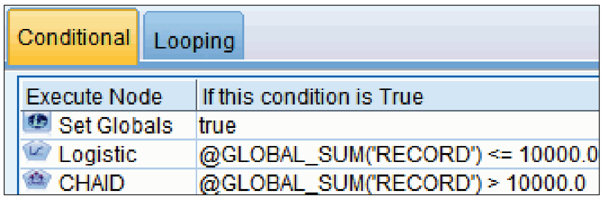
IBM SPSS Modeler also offers the option to automate stream execution by looping through values or fields.



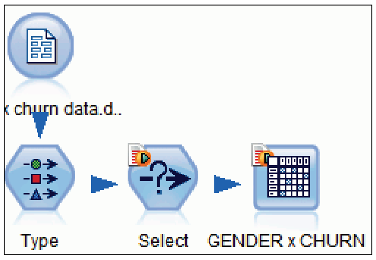
IBM Training

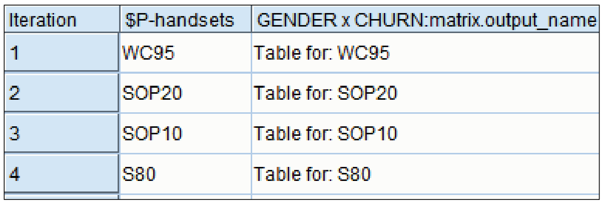
Conditional execution and looping illustrated





Execute Node	If this condition is True
Set Globals	true
Logistic	@GLOBAL_SUM('RECORD') <= 10000.0
CHAID	@GLOBAL_SUM('RECORD') > 10000.0





Iteration	$P-handsets	GENDER x CHURN:matrix.output_name
1	WC95	Table for: WC95
2	SOP20	Table for: SOP20
3	SOP10	Table for: SOP10
4	S80	Table for: S80

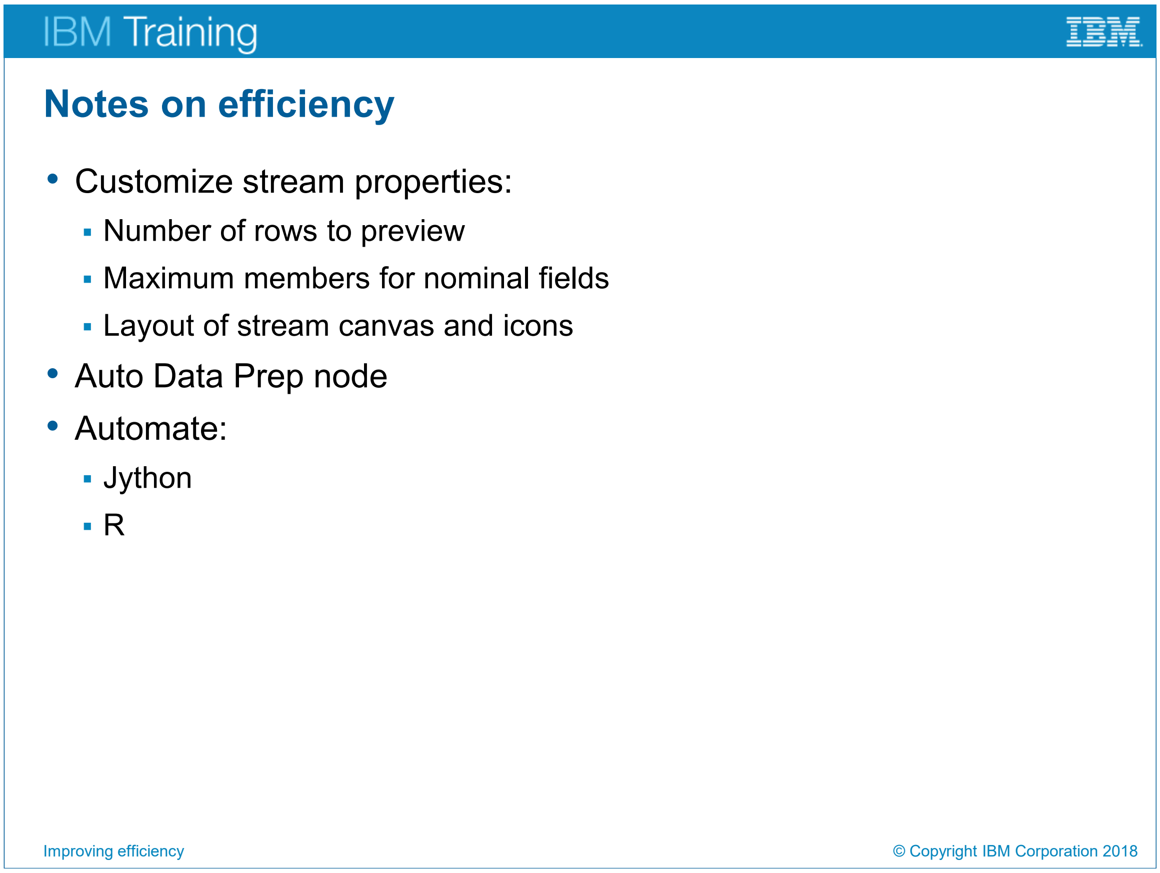
Improving efficiency

© Copyright IBM Corporation 2018

Conditional execution and looping Illustrated

An example of conditional execution is when you want to run one model or the other. Think of running Logistic when the number of records is small and running CHAID when the number of records is large. You can build a stream that stores the number of records in a global variable (using the Set Globals node), and depending on that number you run either the Logistic node or the CHAID node.

As an example of looping, think of cross tabulating GENDER by CHURN in a Matrix node for each type of handset, which you select in a Select node. When there are, for example, 12 handsets you have to edit the Select node and run the Matrix node 12 times. It is more efficiently to loop through the handsets, each time selecting another handset and running the Matrix node.


IBM Training

Notes on efficiency

- Customize stream properties:
  - Number of rows to preview
  - Maximum members for nominal fields
  - Layout of stream canvas and icons
- Auto Data Prep node
- Automate:
  - Jython
  - R

Improving efficiency © Copyright IBM Corporation 2018

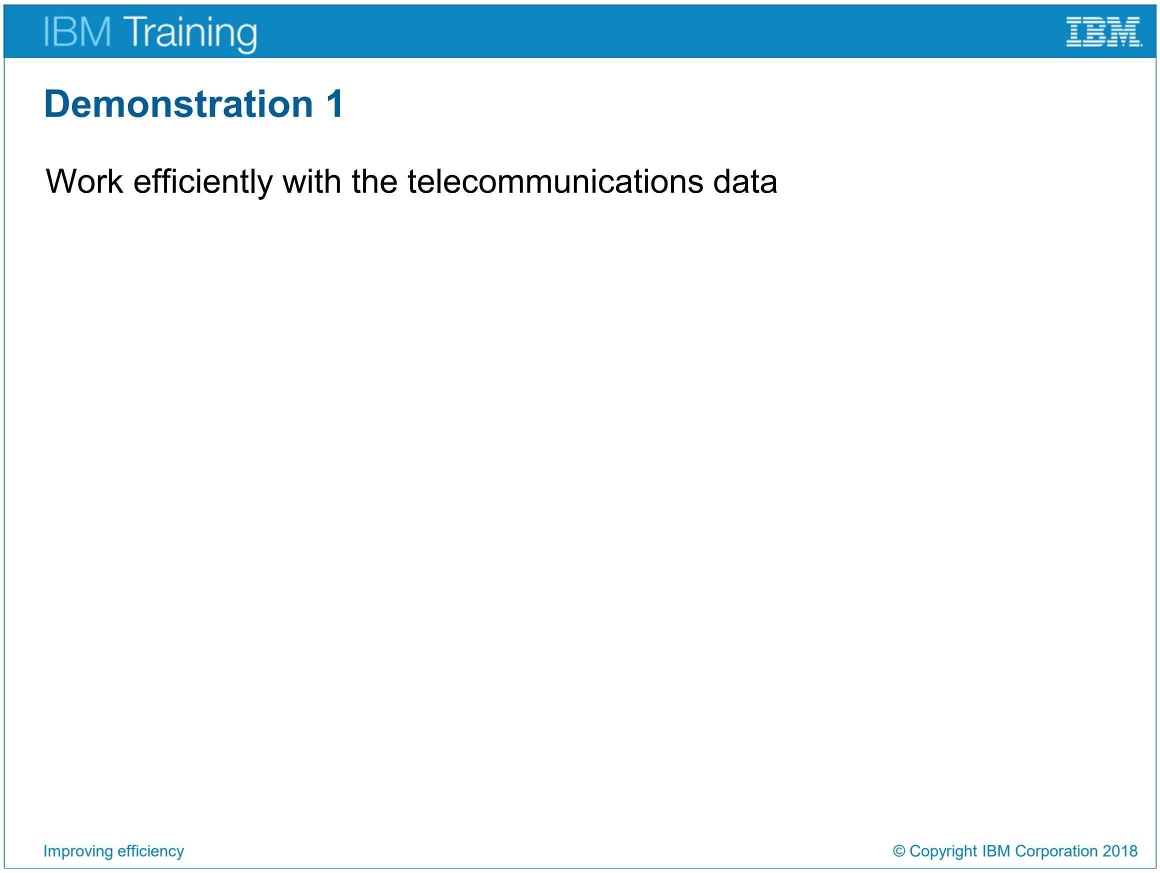
Notes on efficiency

There are more ways to optimize efficiency. The Tools\Stream Properties\Options dialog box enables you to fine tune:

- The number of rows that are displayed when you click the Preview button.
- The Maximum members for nominal fields. This option will automatically set a field's measurement level to typeless when the number of categories exceeds the value specified here. Checking this option will prevent streams from containing long lists of categories to be saved in memory, which speeds loading, saving and processing. Also, it is recommended to set the number to a smaller value, typically in the range from 50 to 100, when you want to use nominal field in modeling.

The Auto Data Prep node screens out fields that are problematic or not likely to be useful, derives new attributes when appropriate, and improves performance through various screening techniques.

The next step in efficiency would be to automate tasks using Jython, R, or Spark. Jython is an implementation of Python, written in Java and running in any environment that supports a Java virtual machine (JVM). To complement IBM SPSS Modeler, the R nodes enable expert R users to input their own R script to carry out data processing, model building and model scoring. The same holds true forSpark. Refer to online Help for more information on Jython, R, and Spark.



IBM Training

Demonstration 1

Work efficiently with the telecommunications data

Improving efficiency

© Copyright IBM Corporation 2018

Demonstration 1: Work efficiently with the telecommunications data

Demonstration 1: Work efficiently with the telecommunications data

Purpose:
You work for a telecommunications firm and want to check your data for outliers and extremes, taking appropriate action if required. You also want to compute standardized scores so that you can use your own cut-off values for outliers and extremes. Finally, you want to explore the relationship between GENDER and CHURN for each region, which you will automate by using parameters and looping.

Data file:	**telco x churn data.csv**
Data folder:	**C:\Training\0A058**
Stream:	**unit_5_demonstration_1_start.str**
Stream folder:	**C:\Training\0A058\Start**

Task 1. Start IBM SPSS Modeler and set the working folder.

1. From the **Start** menu, expand **IBM SPSS Modeler 18.1**, and then click **IBM SPSS Modeler 18.1**. When a splash window displays, click **Cancel**.

   If you have already configured IBM SPSS Modeler in a previous demonstration or exercise, you can skip to Task 2.
2. From the **File** menu, click **Set Directory**.
3. Beside **Look in**, navigate to **C:\Training\0A058**, and then click **Set**.

Task 2. Open the start stream and examine the data.

1. From the **File** menu, click **Open Stream**, navigate to the **C:\Training\0A058\Start** folder, click **unit_5_demonstration_1_start.str**, and then click **Open**.

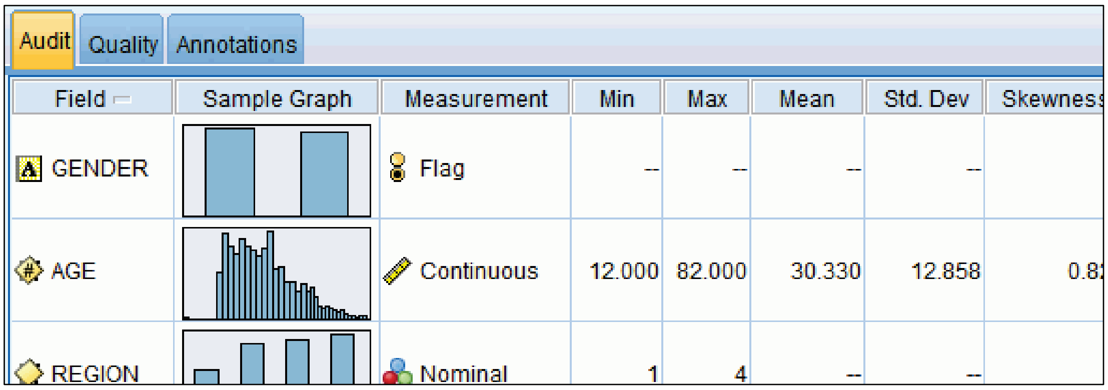
   The stream imports the data, sets measurement levels, declares blanks, and instantiates the data.
2. In the stream, scroll to the branch where the **comment** reads **Outliers and missing values**.
3. Run the **Table** node.

   The dataset stores 31,769 customers of a (fictitious) telecommunications firm.
4. Close the **Table** output window.

Task 3. Use the Data Audit node to process outliers, extremes and missing values.

1. Run the **Data Audit** node named **8 fields**.

   The results appear similar to the following:



Audit | Quality | Annotations

Field	Sample Graph	Measurement	Min	Max	Mean	Std. Dev	Skewness
GENDER		Flag	--	--	--	--	
AGE		Continuous	12.000	82.000	30.330	12.858	0.8
REGION		Nominal	1	4	--	--	

   Mean AGE is 30.330, its standard deviation 12.858. Applying the rule that values less than mean - 3 * standard deviation and values greater than mean - 3 * standard deviation are outliers, there are outliers in the right tail of the distribution (values above 30.330 + 3 * 12.858 ~= 69).

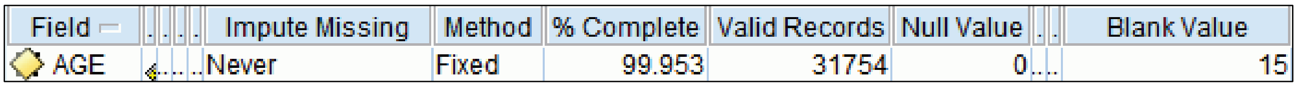
   The question is: Do you want to consider persons older than 69 as outliers? An age higher than 69 is certainly not a-typical, or odd. Deciding which action to take depends on the business question. Maybe the group of retired people is not targeted for any campaign. From this perspective, retired people should be removed from the dataset, to begin with. Maybe the company has special customer loyalty programs in place for the group of 65+ people. In that scenario, it is useful to monitor the behavior of this group.

   All in all, outliers will not be replaced for AGE.

   You will examine missing values for this field.

2. Click the **Quality** tab in the **Data Audit** output window, and then scroll to the last columns in the **Data Audit** output window.

   The results appear similar to the following:



Field	.	.	.	.	Impute Missing	Method	% Complete	Valid Records	Null Value	.	.	Blank Value
AGE		.	.	.	Never	Fixed	99.953	31754	0	.	.	15

   AGE has 15 blank values. You will replace these blank values, using the C&R Tree algorithm.

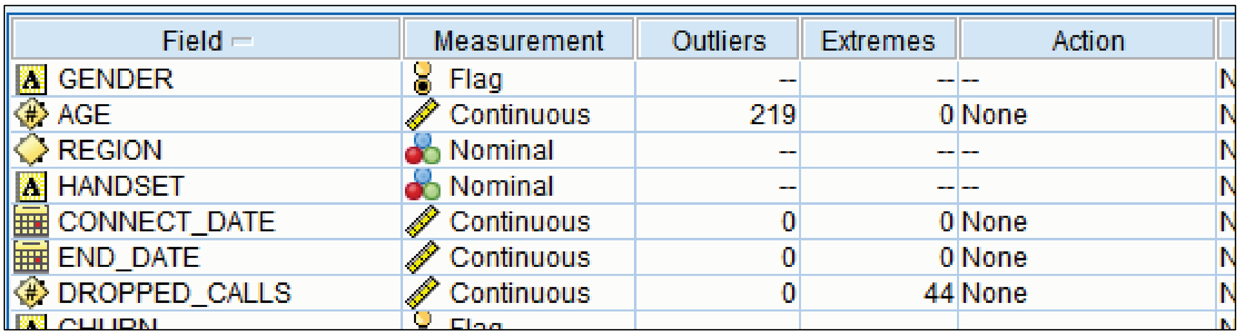
   Note: A discussion of C&R Tree is beyond the scope of this course. Please refer to the *Predictive Modeling for Categorical Targets Using IBM SPSS Modeler* and *Predictive Modeling for Continuous Targets Using IBM SPSS Modeler* course for more information.

3. In the **AGE** row, click the cell in the **Impute Missing** column, and then click **Blank Values**.
4. Click the cell in the **Method** column, and then click **Algorithm.**
5. In the **Fields** column, click **AGE** to ensure that it has focus.
6. From the **Generate** menu, click **Missing Values SuperNode**.
7. In the **Missing Values SuperNode** dialog box that displays, click **Selected fields only** and then click **OK** to close the **Missing Values SuperNode** dialog box.

   A SuperNode named Missing Value Imputation has been generated and is placed in the upper left corner on the stream canvas. You will add this SuperNode to the stream later in this task.

   You will examine outliers and extremes for DROPPED_CALLS. Its mean is 3.190, standard deviation 4.195. Outliers are values above 3.190 + 3 * 4.195 = 15.775, extremes are values above 3.190 + 5 * 4.195 = 24.165. Given the maximum value (45) there is at least one extreme value. The Quality tab gives you the details on outliers and extremes.
8. Scroll to **DROPPED_CALLS**. (Note: If the Data Audit output window is not available any longer, rerun the Data Audit node, and then click the Quality tab.)

   The results appear as similar to the following:



Field	Measurement	Outliers	Extremes	Action
GENDER	Flag	--	--	--
AGE	Continuous	219	0	None
REGION	Nominal	--	--	--
HANDSET	Nominal	--	--	--
CONNECT_DATE	Continuous	0	0	None
END_DATE	Continuous	0	0	None
DROPPED_CALLS	Continuous	0	44	None

   There are no outliers, but 44 extremes. You will coerce these values to the upper limit, which is the mean + 3 * standard deviation. Thus, extreme values will be replaced by 3.190 + 3 * 4.195 = 15.775.
9. In the **DROPPED_CALLS** row, click the cell in the **Action** column, and then click **Coerce**.
10. In the **Fields** column, click **DROPPED_CALLS** to ensure that it has focus.
11. From the **Generate** menu, click **Outlier & Extreme SuperNode**.
12. In the **Outlier SuperNode** dialog box that displays, click **Selected fields only**, and then click **OK** to close the dialog box.

    A SuperNode named Outlier and Extreme has been generated and has been placed in the upper left corner on the stream canvas.

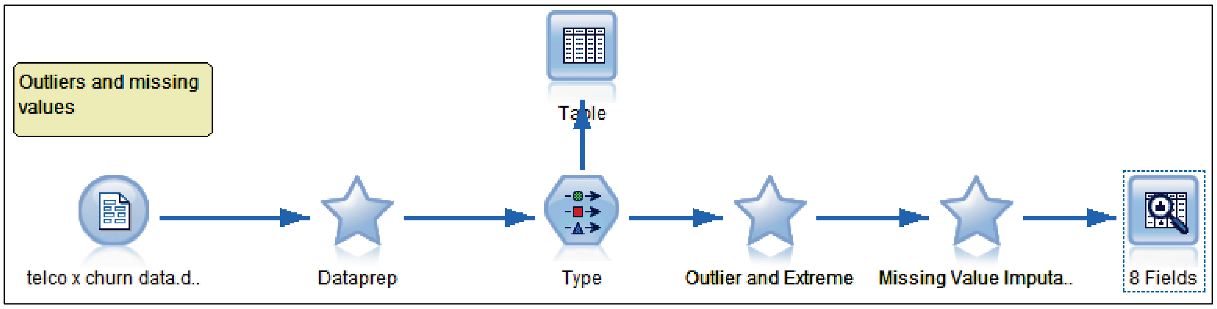
13. Close the **Data Audit** output window.

    You will add the two generated SuperNodes to the stream and rerun the Data Audit node to examine how replacing blanks and coercing extremes affect the statistics.

14. Insert the SuperNode named **Outlier and Extreme** between the **Type** node and the **Data Audit** node named **8 fields**.

15. Insert the SuperNode named **Missing Value Imputation** between the **Outlier and Extreme** SuperNode and the **Data Audit** node named **8 fields**.

    The results appear similar to the following:



16. Run the **Data Audit** node.

    Mean and standard deviation for AGE hardly changed; the number of valid values is 31,769, the size of the dataset. Thus, imputing missing values by the C&R Tree algorithm did not affect the center and spread of the distribution, and at the same time made more records are available for modeling.

    Mean and standard deviation for DROPPED_CALLS is 3.157 and 3.996, respectively (they were 3.190 and 4.195 before coercing extremes). Outliers are now above 3.157 + 3 * 3.996 = 15.145, extremes above 3.157 + 5 * 3.996 = 23.137. This also implies that there will be outliers, the extremes that were just coerced to 15.775.

17. Click the **Quality** tab in the **Data Audit** output window.

    44 outliers are reported for DROPPED_CALLS, as expected.

18. Close the **Data Audit** output window.

Task 4. Compute standardized scores using globals.

To have more control over outliers and extremes you will compute the number of standard deviations that a value is above or below the mean, known as standardized scores or z-scores. Having a field with standardized scores enables you to select outliers and/or extremes with any cut-off value that you want, to sort records on the standardized scores, and so forth.

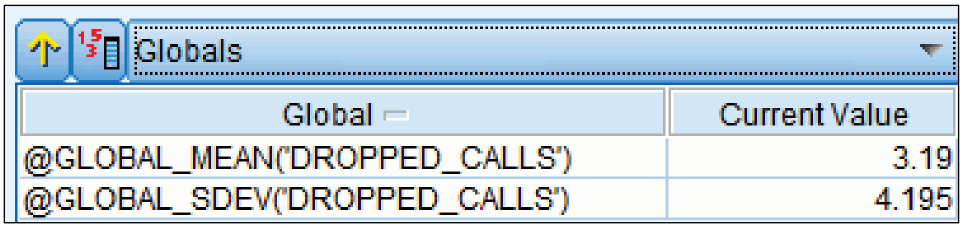
A standardized score for a field X is computed as (value X - mean X) / standard deviation X. You will store mean and standard deviation in global variables, and then compute standardized scores.

1. Scroll to the branch where the **comment** reads **Using globals**.
2. From the **Output** palette, add the **Set Globals** node downstream from the **Type** node.
3. Edit the **Set Globals** node.
4. Beside **Field**, select **DROPPED_CALLS**.
5. Disable all options, except for the **Mean** and **SDev** options.
6. Enable the **Display preview of globals created after execution** option (you can then immediately examine the statistics when the Set Globals node is executed).
7. Click **Run**.

   Mean and standard deviation are in agreement with the output from the Data Audit node for this field in the previous task (prior to coercing extremes).

   You will now use the mean and standard deviation in a Derive node to compute standardized scores, or z-scores, for DROPPED_CALLS.
8. Close the **Globals** window.
9. From the **Field Ops** palette, add a **Derive** node downstream from the **Type** node.
10. Edit the **Derive** node.
11. Under **Derive field**, type **Z_SCORES_DROPPED_CALLS**.
12. Click the **Launch expression builder** button**.**
13. Click the **Fields** drop down list, and then click **Globals**.

    The results appear similar to the following:

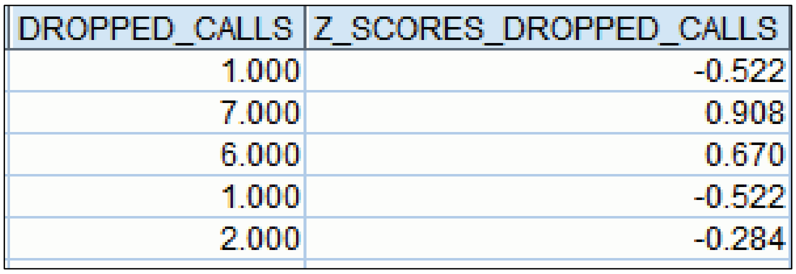


Global	Current Value
@GLOBAL_MEAN('DROPPED_CALLS')	3.19
@GLOBAL_SDEV('DROPPED_CALLS')	4.195

The Globals field is populated with the values.

14. Create the expression **(DROPPED_CALLS - @GLOBAL_MEAN ('DROPPED_CALLS')) / @GLOBAL_SDEV('DROPPED_CALLS')**.
15. Close the **Expression** builder.
16. Click **Preview**, and then move **Z_SCORES_DROPPED_CALLS** next to **DROPPED_CALLS** in the **Preview** output window.

    The results appear similar to the following:

DROPPED_CALLS	Z_SCORES_DROPPED_CALLS
1.000	-0.522
7.000	0.908
6.000	0.670
1.000	-0.522
2.000	-0.284

    You could now select or discard the records using your own cut-off value. For example, if you want to select records that have a value for DROPPED_CALLS within 4 standard deviations from the mean, the expression would be: abs (Z_SCORES_DROPPED_CALLS) < 4.

    Note: the abs function takes the absolute value.
17. Close the **Preview** output window.
18. Close the **Derive** dialog box.

Task 5. Use parameters.

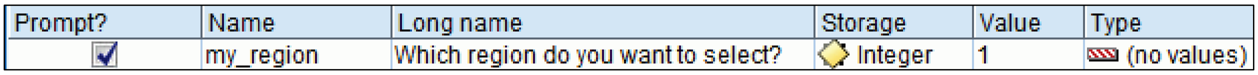
In this task you will replace a hard coded selection of records with a parameterized selection.

1. Scroll to the branch where the **comment** reads **Using parameters**.

   The Select node selects records from a certain region (REGION = 1 at this moment), and a Matrix node is run downstream from the Select node. Rather than hard coding the region, you want to be prompted for the region that you want to run the Matrix node for. You will accomplish this by defining a parameter.
2. From the **Tools** menu, click **Stream Properties**, and then click **Parameters**.
3. Enable the **Prompt?** option.
4. Under **Name**, type **my_region**.
5. Under **Long name**, type **Which region do you want to select?**.
6. Under **storage**, select **Integer** (since REGION is integer valued, the parameter values will be integers also).

7. Under **Value**, type **1** (this will be the default value).

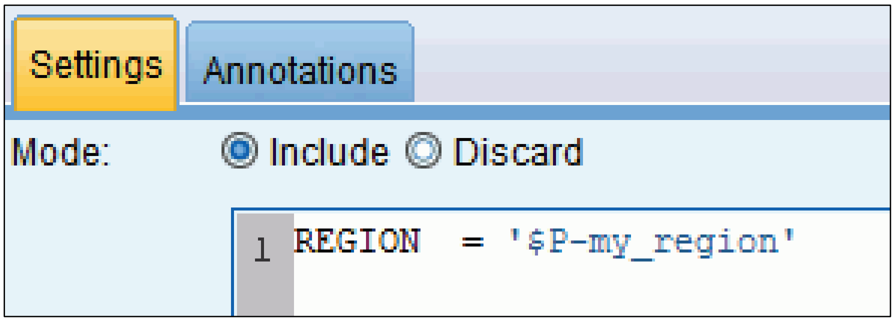
   The results appear as follows:



Prompt?	Name	Long name	Storage	Value	Type
✓	my_region	Which region do you want to select?	Integer	1	(no values)

8. Close the dialog box.
9. Edit the **select region** node.
10. Delete the value **1** in the condition.
11. Click the **Expression Builder** button, select **Parameters** in the drop down list on the right side, select **'$P-my_region'** from there, paste it into the text box, and then click **OK** return to the main dialog box.

    The results appear similar to the following:

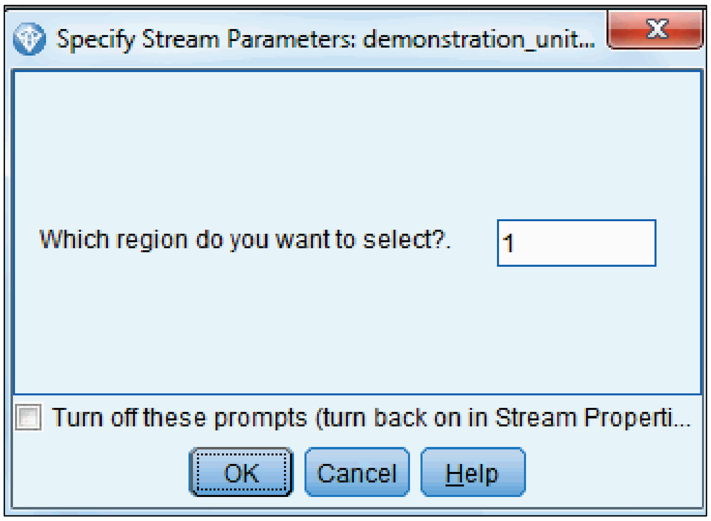


Mode: [✓ Include] Discard

```
REGION  = '$P-my_region'
```

12. Close the **Select** dialog box.
13. Run the **Matrix** node named **GENDER x CHURN**.

    The results appear as follows:



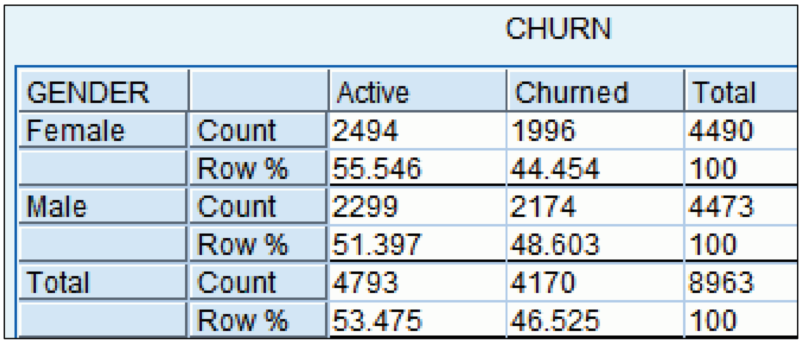
Which region do you want to select?. 1

Turn off these prompts (turn back on in Stream Properti...

You will run the analysis for region 4.

14. Replace **1** by **4**, and then click **OK**.

    The results appear as follows:



		CHURN		
GENDER		Active	Churned	Total
Female	Count	2494	1996	4490
	Row %	55.546	44.454	100
Male	Count	2299	2174	4473
	Row %	51.397	48.603	100
Total	Count	4793	4170	8963
	Row %	53.475	46.525	100

    These are the results for region 4.

15. Close the **Matrix** output window.

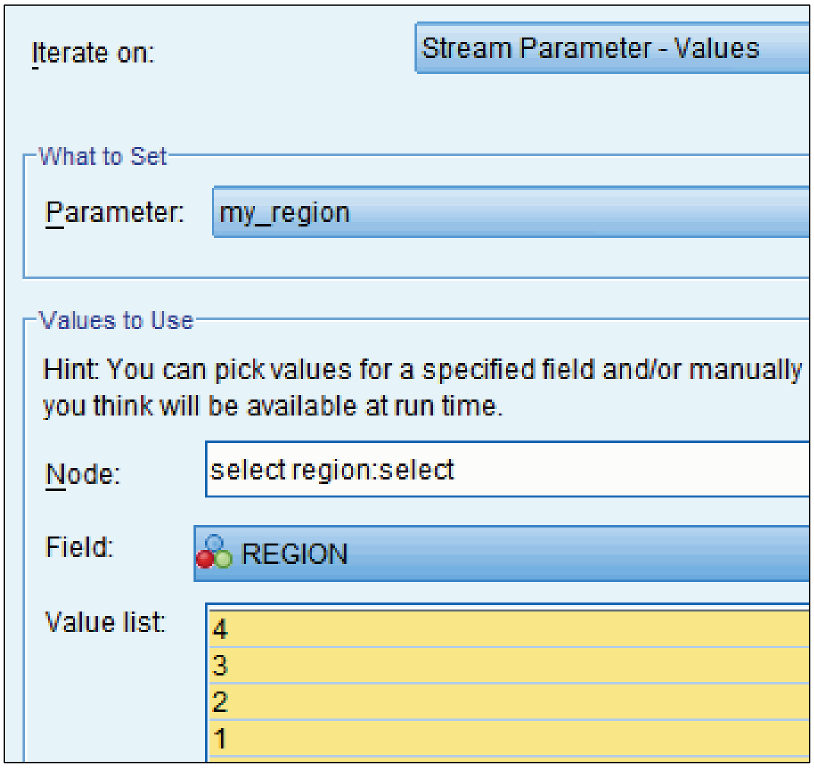
Task 6. Loop through values.

In the previous task you have created a parameter that lets you select the region at the prompt. In this task you will loop through the regions, and run the Matrix node for each region. In order to loop through the values of REGION you must have a parameter in place that serves as the placeholder for the values. You already have a parameter in place, as per the previous task. In this case, however, you do not want to have a prompt, but just loop through the values.

1. From the **Tools** menu, click **Stream Properties**, click **Parameters**, disable the **Prompt?** option, and then close the dialog box.
2. Right-click the s**elect region** node, click **Looping/Conditional Execution** from the context menu, and then click **Define Iteration Key (Values)**.
3. Beside **Iterate on**, select **Stream Parameter - Values** from the list.
4. Beside **Parameter**, click **my_region** from the list.
5. Beside **Node**, click **select region** from the list. It will appear as **select region: select** after you click **OK**.
6. Beside **Field**, select **REGION**.

7. Beside **Value list**, click the **Specify values** button, click the **Select All** button, and then click the **Select** button.

   The results appear as follows:



Iterate on: Stream Parameter - Values

What to Set

Parameter: my_region

Values to Use

Hint: You can pick values for a specified field and/or manually
you think will be available at run time.

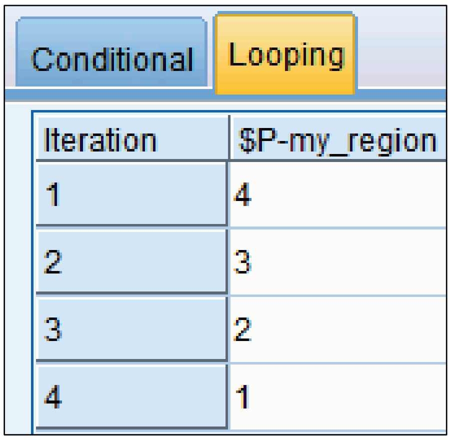
Node: select region:select

Field: REGION

Value list:
4
3
2
1

8. Click **OK** to close the **Define Iteration Key** dialog box.

   The results appear as follows:



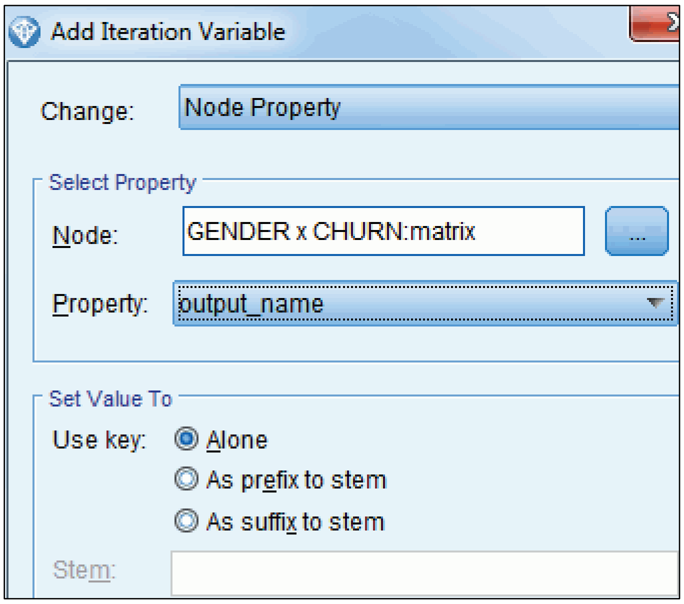
Iteration	$P-my_region
1	4
2	3
3	2
4	1

A loop has been created which iterates through the values 1 to 4 of the parameter $P-my_region.

Running the stream will generate a Matrix table for each value of the loop. Thus, you will generate 4 Matrix tables. You will assign a unique output name to each Matrix table, reflecting the region that was selected (if not, the Matrix table that is produced will be overwritten in each iteration).

9. Click the **Add Variable** button.
10. Beside **Change**, select **Node Property**.
11. Beside **Node**, select the **GENDER x CHURN** matrix node.
12. For **Property**, select **output_name**.

    The results appear as follows:



Add Iteration Variable

Change: Node Property

Select Property

Node: GENDER x CHURN:matrix

Property: output_name

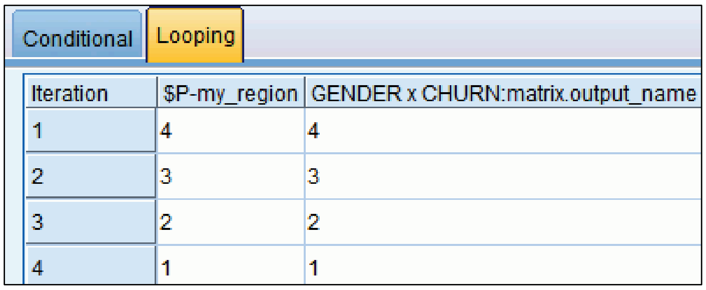
Set Value To

Use key: [✓ Alone] As prefix to stem / As suffix to stem

Stem:

13. Click **OK** to close the **Add Iteration Variable** dialog box.

    The results appear as follows:



Iteration	$P-my_region	GENDER x CHURN:matrix.output_name
1	4	4
2	3	3
3	2	2
4	1	1

14. Close the dialog box.

    The Select node and Matrix node show a little icon indicating that looping has been set on these nodes.

    To enable looping, ensure that you execute the entire stream, and not just the Matrix node.

15. Click the **Run the current stream** ▶ button.

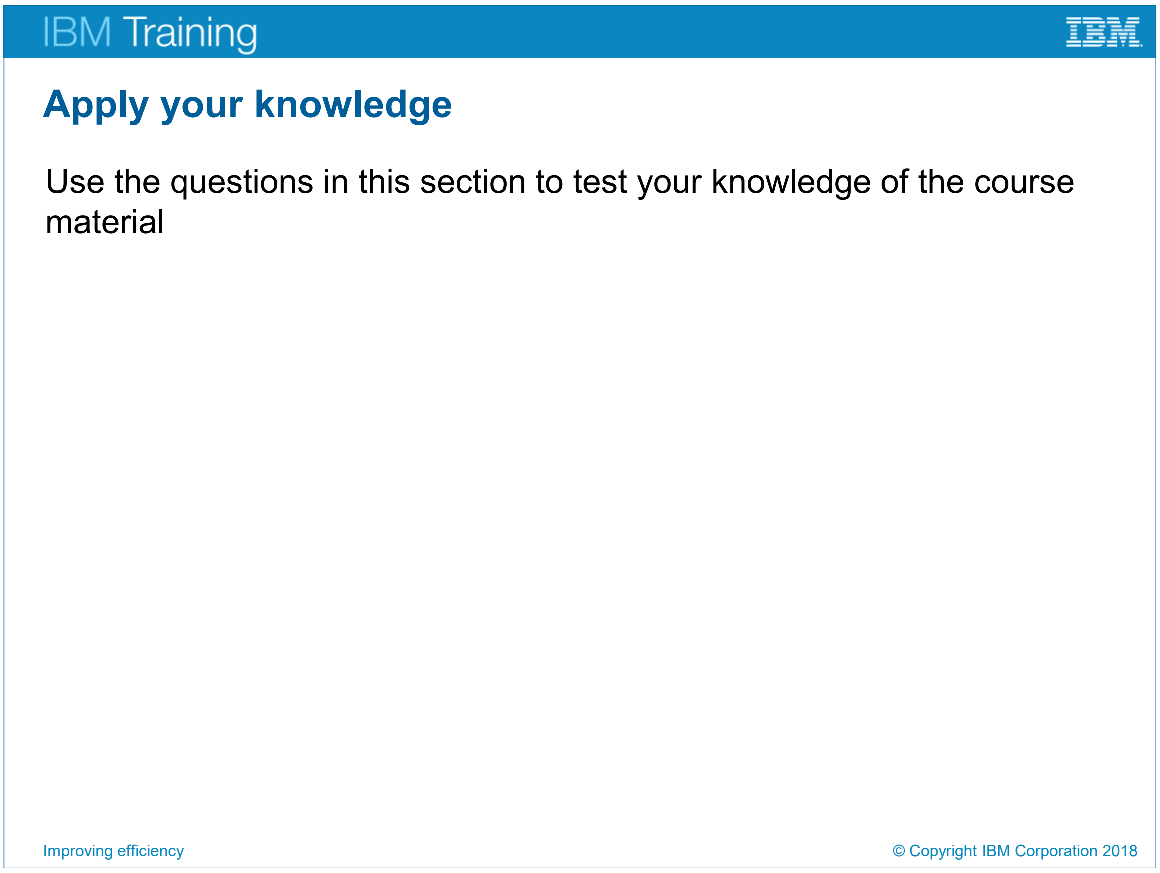
    The Outputs tab in the upper right pane will contain the four tables (you will have some additional output because all terminal nodes were executed).

    This completes the demonstration You will create a clean state for the exercises.

16. From the **File** menu, click **Close Stream**. Click **No** when asked to save the stream.
17. From the **File** menu, click **New Stream**.
18. Leave **IBM SPSS Modeler** open for the exercise.

Results:
You have checked your data for outliers and extremes, and you have taken appropriate action. You also have computed standardized scores so that you could use your own cut-off values to process outliers and extremes. Finally, you have used parameters and looping to automate tasks.

You will find the completed stream in the **C:\Training\0A058\Solutions** folder.



IBM Training

Apply your knowledge

Use the questions in this section to test your knowledge of the course material

Improving efficiency

© Copyright IBM Corporation 2018

Apply your knowledge

Questions

Question 1: Which of the following statements are correct? (Select all that apply.)

A. The standard deviation cannot be computed for a string field.

B. If all records have the same value for the (continuous) field X, the standard deviation for X equals 0.

C. The standard deviation for a field X will not change if we add 5 to each score of X.

D. If two fields X and Y have the same unit of measurement (say, inches), and X has a standard deviation of 10, and Y of 100, the histogram for Y will show more spread than the histogram for X.

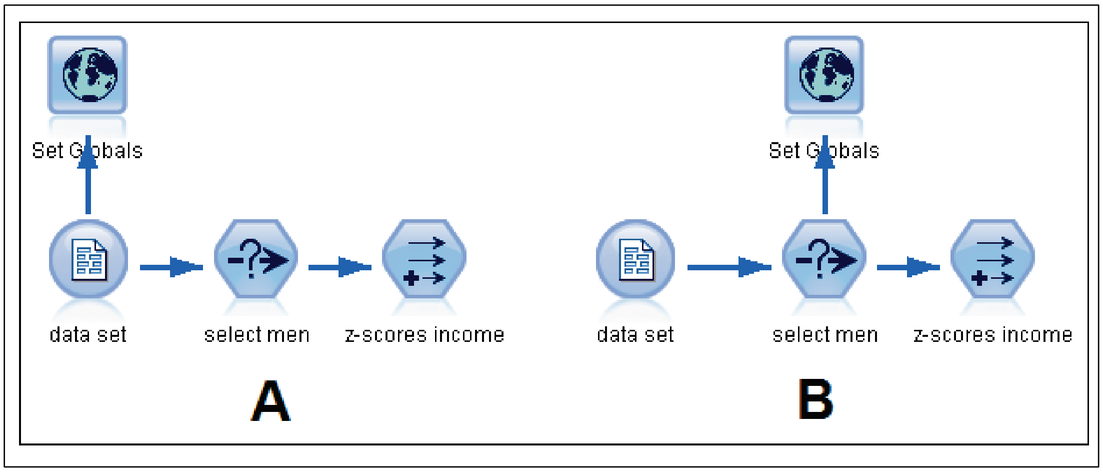
Question 2: True or false: The median is one of the statistics that can be requested in the Set Globals node.

A. True

B. False

Question 3: Refer to the figures A and B below. The Set Globals node in both streams computes the mean and standard deviation of income.

Needed are standardized scores for income, for men only, using the mean income and standard deviation income for men. Stream _____ (A/B) accomplishes this.



Question 4: SQL pushback is only relevant for:

A. Text files

B. Databases

C. IBM SPSS Statistics (*.sav) files

D. XML files

Question 5: True or false: The operations that can be pushed back to the database depend on the database in use.

A. True

B. False

Question 6: True or false: A parameter that is defined for a particular SuperNode will automatically be available for all SuperNodes in the IBM SPSS Modeler session.

A. True

B. False

Question 7: You have a field named HANDSET in your dataset with 24 categories. You want to export your data to 24 data files, one data file for each handset. Which automation method will you use?

A. Conditional execution

B. Looping

C. Auto Cluster

Question 8: True or false: You can run a Distribution graph for each (categorical) field in the dataset with a single Distribution node by setting a loop on the node.

A. True

B. False

Answers to questions

Answer 1: A, B, C, and D. Computing the standard deviation is only relevant for numeric fields. When all the scores are the same, there is no spread and the standard deviation will be 0. The spread will not change when you add 5 to the scores. When two fields X and Y have the same unit of measurement, the field with the larger standard deviation will show more spread.

Answer 2: B. False. The median cannot be requested as a global variable. If you want the median you can use an Aggregate node and merge the aggregated dataset back to the original dataset.

Answer 3: B. The mean needs to be computed for men only, so downstream from the node where men are selected.

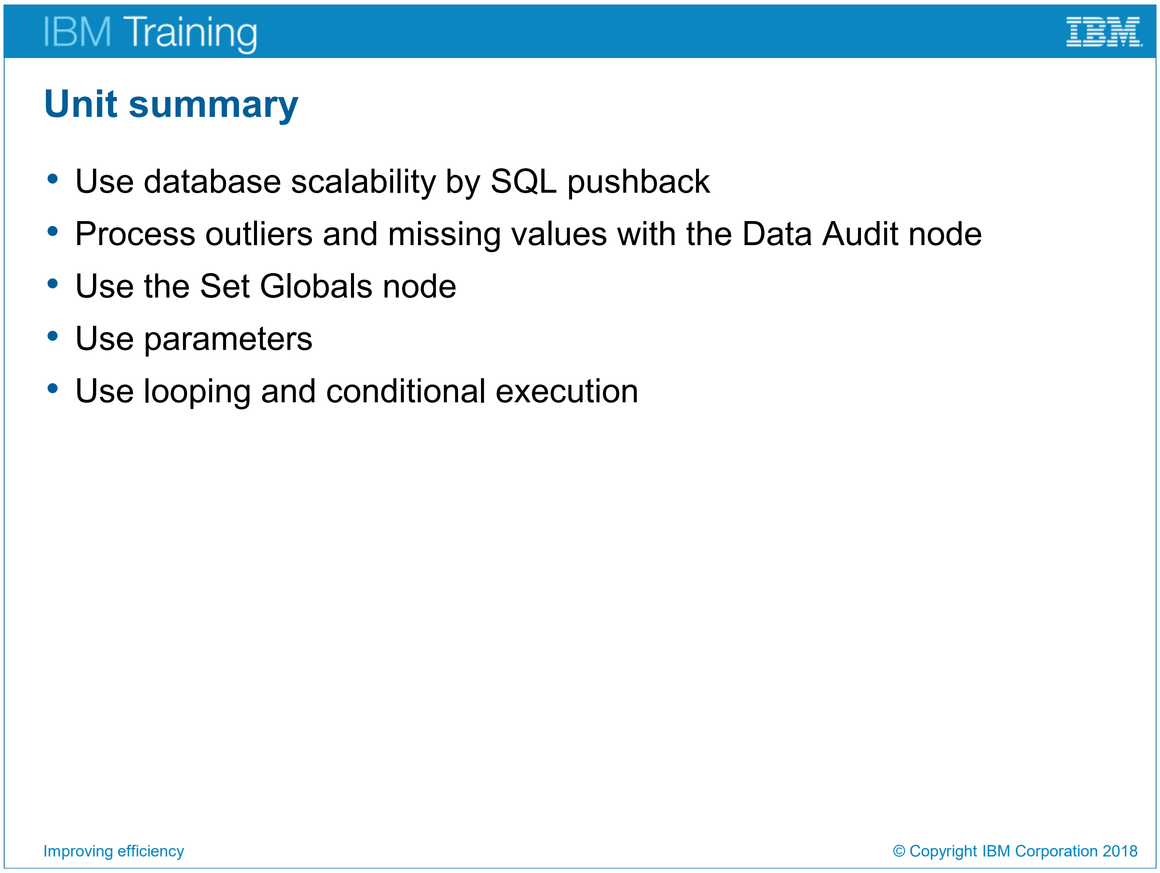
Answer 4: B. SQL pushback is only relevant when you import data from a database.

Answer 5: A. True. It is possible that an operation can be pushed back to database A, but not to database B.

Answer 6: B. False. A SuperNode parameter is available only in the SuperNode where you defined the parameter.

Answer 7: B. You will loop through the categories of HANDSET.

Answer 8: A. True. You can create a loop to go through fields in a Distribution node.



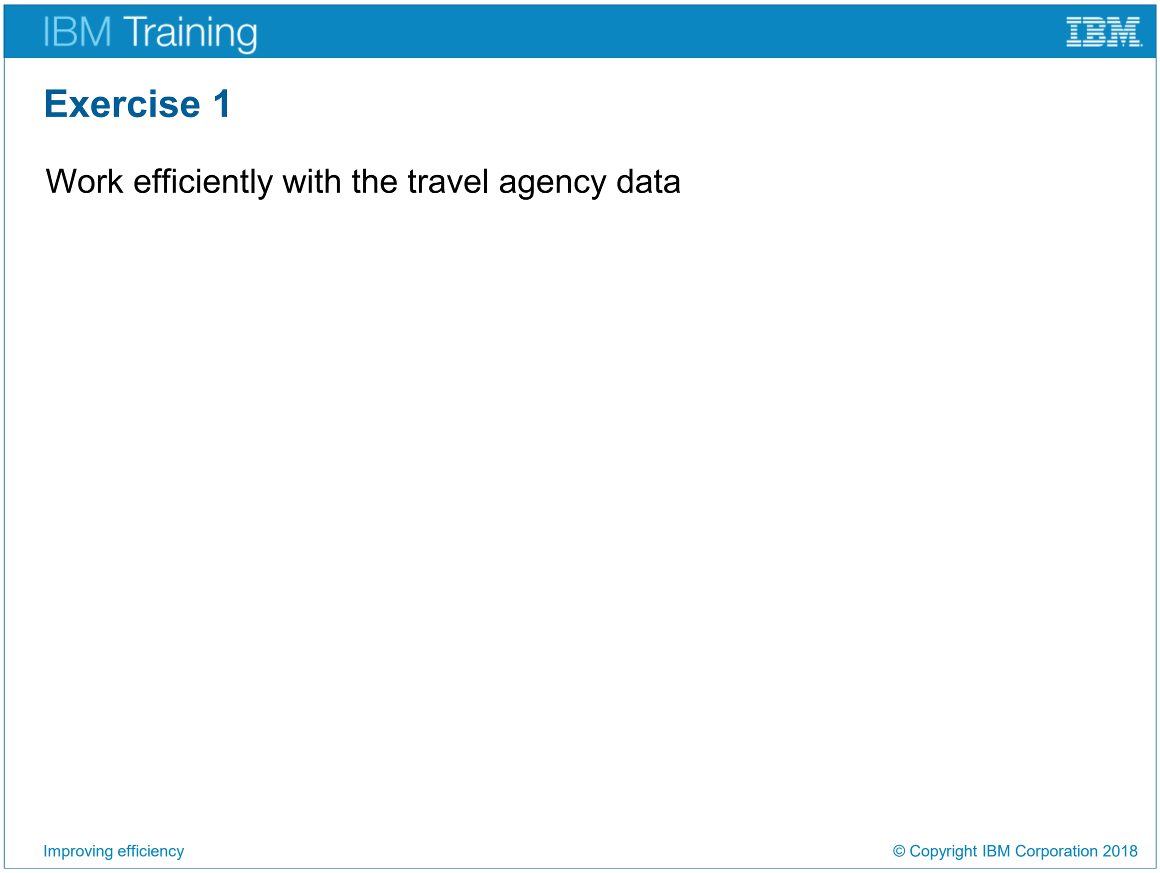
IBM Training

Unit summary

- Use database scalability by SQL pushback
- Process outliers and missing values with the Data Audit node
- Use the Set Globals node
- Use parameters
- Use looping and conditional execution

Improving efficiency

Unit summary



IBM Training

Exercise 1

Work efficiently with the travel agency data

Improving efficiency

© Copyright IBM Corporation 2018

Exercise 1: Work efficiently with the travel agency data

Exercise 1: Work efficiently with the travel agency data

Data file:	**customers_and_holidays.csv**
Data folder:	**C:\Training\0A058**
Stream file:	**unit_5_exercise_1_start.str**
Stream folder:	**C:\Training\0A058\Start**

In this exercise you will process outliers, extremes and missing values, using the Data Audit node. You will use the Set Globals node to replace missing values, and you will be introduced to automation by using parameters and looping.

- The stream **unit_5_exercise_1_start.str**, located in the **C:\Training\0A058\Start** folder, imports the data, cleanses data, declares -999 as blank for DISTANCE_TO_BEACH, and instantiates the data.
  - Open the stream.
  - Run the **Data Audit** node.
- What is the minimum, maximum, mean and standard deviation for **DISTANCE_TO_BEACH**? Do you expect outliers (values more than 3 standard deviations away from the mean) or extremes (values more than 5 standard deviations away from the mean)?
- Check your expectations by examining the number of outliers and extremes for **DISTANCE_TO_BEACH** on the **Quality** tab of the **Data Audit** output.
- On the **Quality** tab of the **Data Audit** output window, coerce outliers and nullify extremes for **DISTANCE_TO_BEACH**, and then run a **Data Audit** node to examine the effect of coercing outliers and nullifying extremes.

  Are there still outliers and extremes after you coerced outliers and nullified extremes?
- Starting from the stream in the previous task (so outliers coerced and extremes nullified for **DISTANCE_TO_BEACH**), replace undefined ($null$) values of **DISTANCE_TO_BEACH** with the **mean DISTANCE_TO_BEACH**, by using the **Set Globals** node, in conjunction with a **Filler** node.

  Note: This can also be accomplished by generating a Missing Value SuperNode from the Quality tab in the Data Audit output. However, there is a subtle difference because the Missing value SuperNode will have the mean hard coded, whereas the global stores a value which can updated easily (by re-executing the Set Globals node).

- Select all customers that went to **SPAIN**, and request a cross tabulation (**Matrix** node, located in the **Output** palette) of **GENDER** in the row by **SATISFIED** in the column for customers that went to Spain. Request row percentages on the **Appearance** tab in the **Matrix** node.

  What is the percentage of men that is satisfied? And what is the percentage of women that is satisfied?
- Define a parameter defined named **my_country**, long name **What was your country destination?**, default **SPAIN**, and use this parameter so that you are prompted for the country name when you run the **Matrix** node of **GENDER** by **SATISFIED**.

  Test whether the parameterization is working correctly, using the default value SPAIN.
- Create a loop so that the **Matrix** node is executed for row fields **GENDER**, **REGION**, **HOLCODE**, **POOL** and **ACCOM** by column field **SATISFIED**.

  Hint: create a loop on the **Matrix** node, with a loop through fields, in particular a loop through the row fields in the Matrix node.
  - Right-click the Matrix node, click Looping/Conditional Execution, and then click Define Iteration Key (Fields).
  - For Iterate on, ensure that Node Property - Fields is selected.
  - For Node, ensure that GENDER x SATISFIED:matrix is selected.
  - For Property, select row (this refers to the row field in the cross tabulation).
  - In the Fields to use section, select GENDER, REGION, HOLCODE, POOL, and ACCOM.
  - Close the dialog boxes and then click the Run the current stream ▶ button (take the default SPAIN as value for the country destination parameter; you can also turn the prompt off).
- Exit IBM SPSS Modeler without saving anything.

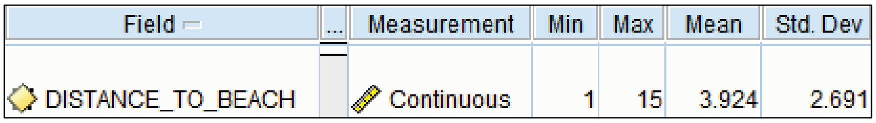
For more information about where to work and the exercise results, refer to the Task and Results section that follows. If you need more information to complete a task, refer to earlier demonstrations for detailed steps.

Exercise 1: Tasks and Results

Task 1. Open the start stream and examine the data.

- From the **File** menu, click **Open Stream**, navigate to **C:\Training\0A058\Start**, and then open **unit_5_exercise_1_start.str**.
- Run the **Data Audit** node.

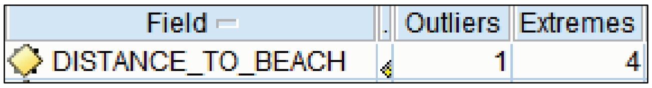
  The results appear as follows:



Field	...	Measurement	Min	Max	Mean	Std. Dev
DISTANCE_TO_BEACH		Continuous	1	15	3.924	2.691

- The scores range from 1 to 15. The mean and standard deviation are 3.92 and 2.69, respectively. It is expected that there are no outliers or extremes in the left tail of the distribution. It is expected that there are outliers (values above 3.92 + 3 * 2.69 ~= 12) and no extremes (values above 3.92 + 5 * 2.69 ~= 17.3) in the right tail of the distribution.
- Click the **Quality** tab in the **Data Audit** output window.

  The results appear as follows:



Field	.	Outliers	Extremes
DISTANCE_TO_BEACH		1	4

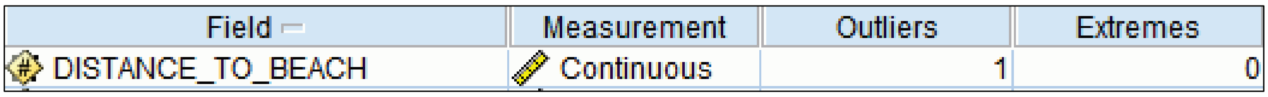
- One outlier is reported, which must be the score of 15. Contrary to the expectations there are 4 extremes. Apparently, the value -999, which was declared as blank value upstream and has a frequency of 4, is reported as an extreme value.

Task 2. Process outliers and extremes for DISTANCE_TO_BEACH.

- With still the **Data Audit** window, **Quality** tab open:
  - In the **DISTANCE_TO_BEACH** row, click the cell in the **Action** column.
  - Click **Coerce outliers / nullify extremes**.
- Ensure that **DISTANCE_TO_BEACH** has focus (click on the field name so that it is highlighted).
- From the **Generate** menu, click **Outlier & Extreme SuperNode**, choose **Selected fields only**, and then click **OK**.

- Add the generated **SuperNode** named **Outlier and Extreme** downstream from the **Type** node.
- From the **Output** palette, add a **Data Audit** node downstream from the **SuperNode** named **Outlier and Extreme**.
- Run the **Data Audit** node.
- Click the **Quality** tab.

  The results appear similar to the following:



Field	Measurement	Outliers	Extremes
DISTANCE_TO_BEACH	Continuous	1	0

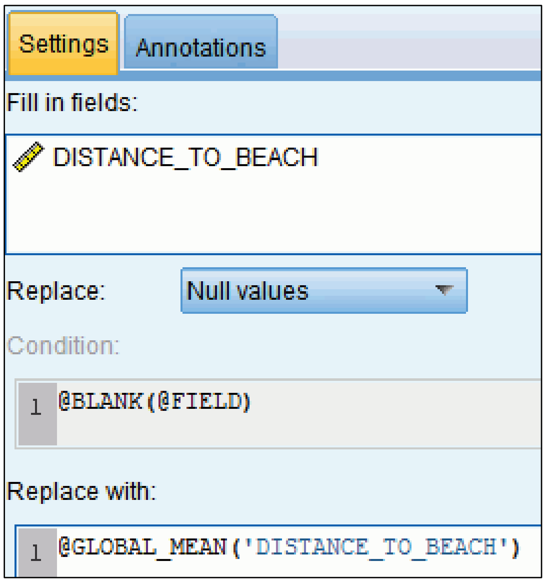
- There is one outlier, which is, when you examine the Audit tab, just above mean + 3 * SD, so there is no reason to take further action.

Task 3. Use globals to replace undefined values with the mean.

- From the **Output** palette, add a **Set Globals** node downstream from the **SuperNode** named **Outlier and Extreme**.
- Edit the **Set Globals** node:
  - For **Fields**, select **DISTANCE_TO_BEACH**.
  - Ensure that the **MEAN** option is enabled.
  - Disable all other options.
  - Run the **Set Globals** node.
- From the **Field Ops** palette, add a **Filler** node downstream from the **SuperNode** node named **Outlier and Extreme**.

- Edit the **Filler** node:
  - For **Fill in fields**, select **DISTANCE_TO_BEACH**.
  - Set **Replace:** to **Null values**.
  - Set **Replace with** to **@GLOBAL_MEAN('DISTANCE_TO_BEACH')**. (Use the Expression Builder to create the expression, if preferred.)

  The results appear as follows:



Fill in fields:
DISTANCE_TO_BEACH

Replace: Null values

Condition:
```
@BLANK(@FIELD)
```

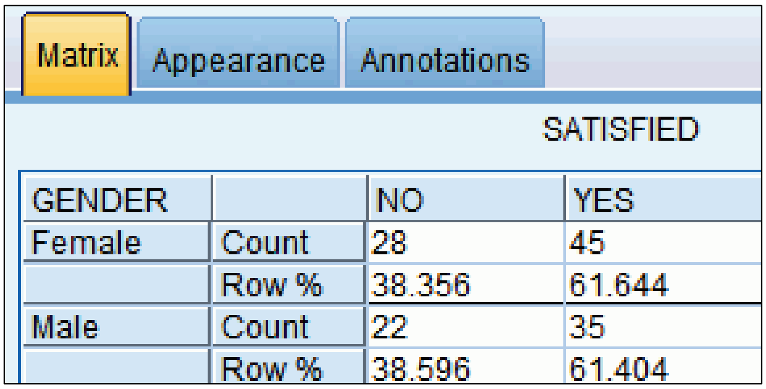
Replace with:
```
@GLOBAL_MEAN('DISTANCE_TO_BEACH')
```

Task 4. Cross tabulate GENDER by SATISFIED, for customers with destination SPAIN.

- From the **Record Ops** palette, add a **Select** node downstream from the **Filler** node.
- Edit the **Select** node, and then enter the condition **COUNTRY = "SPAIN"**.
- From the **Output** palette, add a **Matrix** node downstream from the **Select** node.

- Edit the **Matrix** node, and then:
  - Select **GENDER** in the **Rows**.
  - Select **SATISFIED** in the **Columns**.
  - Click the **Appearance** tab, and then enable the **Percentage of row** option.
  - Run the **Matrix** node.

  The results appear as follows:

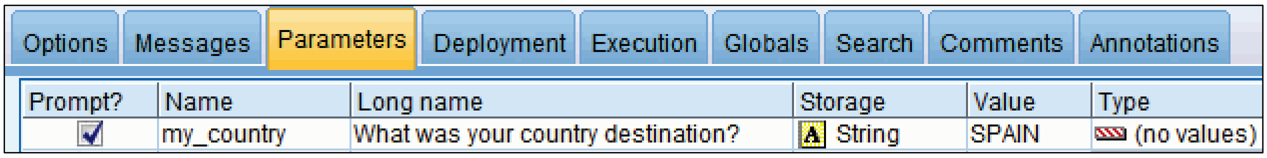


		SATISFIED	
GENDER		NO	YES
Female	Count	28	45
	Row %	38.356	61.644
Male	Count	22	35
	Row %	38.596	61.404

There is almost no difference in satisfaction between men and women who went to Spain.

Task 5. Define a parameter to select the destination.

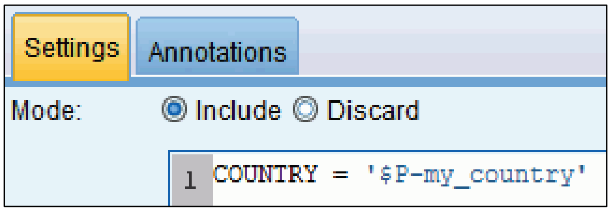
- From the **Tools** menu, click **Stream Properties**, and then click **Parameters**.
- Enable the **Prompt?** option.
- Enter the specifications, as shown in the figure that follows.



Prompt?	Name	Long name	Storage	Value	Type
✓	my_country	What was your country destination?	String	SPAIN	(no values)

- Edit the **Select** node that is already on the stream canvas, and then replace **"SPAIN"** with **'$P-my_country'**. (Use the Expression Builder, if preferred.)

  The results appear as follows:



Mode: [✓ Include] Discard

```
COUNTRY = '$P-my_country'
```

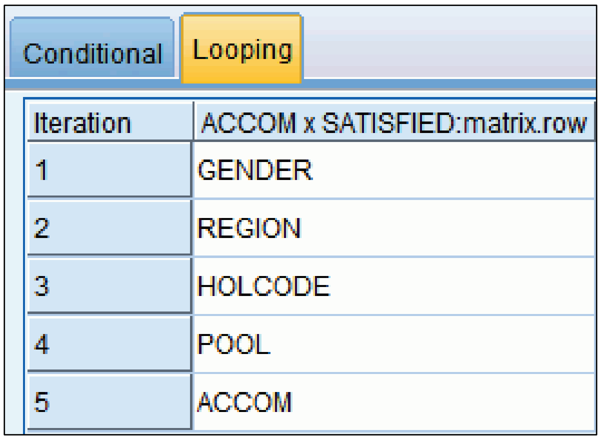
- Run the **Matrix** node.

- You will be prompted for the country. Keep the country that is suggested, SPAIN (the default), and then click **OK** to continue.
- The Matrix node output is the same as in the previous task, so the parameter is defined correctly.

Task 6. Create a loop through row fields in the Matrix node.

- Right-click the **Matrix** node, click **Looping/Conditional Execution**, and then click **Define Iteration Key (Fields)**.
- For **Iterate on**, select **Node Property - Fields**.
- For **Node**, select **GENDER x SATISFIED** matrix.
- For **Property**, select **row** (this refers to the row field in the cross tabulation).
- In the **Field list**, select **GENDER**, **REGION**, **HOLCODE**, **POOL**, and **ACCOM**.
- Close the **Define Iteration Key** dialog box.

  The results appear as follows:



Iteration	ACCOM x SATISFIED:matrix.row
1	GENDER
2	REGION
3	HOLCODE
4	POOL
5	ACCOM

- Close the **Stream Properties** dialog box, and then click the **Run the current stream** ▶ button.

  If a dialog box displays asking you for a country, go ahead with the default country (SPAIN), enable the **Turn off these prompts (…)** option, and then click **OK**.

  The Matrix node is executed 5 times, generating a cross tabulation for each of the row fields (more output is produced because of the various Data Audit nodes). In practice, you would first delete the Data Audit nodes before looping.

Task 7. Exit IBM SPSS Modeler.

- From the **File** menu, click **Exit**, and then exit **IBM SPSS Modeler** without saving.

You will find the solution results in the C:\Training\0A058\Solutions folder.

IBM Training