## Contents

### Chapter 1 • Using Data Grids in SAS Intelligent Decisioning

- Introduction to Data Grids ........................................ 1
- Serialize and Import Data Grids ................................ 3
- Working with Data Grids ........................................... 3
- Using Data Grid Functions ........................................ 5
- Processing Data Grids in a Code File ........................... 6
- Working with Data Grids in SAS Studio ....................... 6

### Chapter 2 • Data Grid Macros

- Using the Data Grid Macros ......................................... 9
- Data Grid Macros Available in SAS Intelligent Decisioning . 9
- Dictionary .................................................................. 10

### Chapter 3 • Data Grid Functions

- Comparison Operators ............................................... 18
- Data Grid Functions Available in SAS Intelligent Decisioning 18
- Dictionary .................................................................. 21
Introduction to Data Grids

A data grid is a table that is the value of a table cell. Data grids enable you to create a table within a table. For example, suppose you have a table that contains the data for all of the insurance policies for all of your customers. This table might look like the table shown in Table 1.1.

Table 1.1  Insurance Policy Table

<table>
<thead>
<tr>
<th>PolicyHolder</th>
<th>PolicyNumber</th>
<th>YearlyPremium</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smyth, Joe</td>
<td>453975R398</td>
<td>439.50</td>
</tr>
<tr>
<td>Smyth, Joe</td>
<td>987348P210</td>
<td>132.90</td>
</tr>
<tr>
<td>Dupree, Marcel</td>
<td>983092B228</td>
<td>334.00</td>
</tr>
<tr>
<td>Dupree, Marcel</td>
<td>274933P412</td>
<td>219.25</td>
</tr>
</tbody>
</table>
Alternatively, you can use a data grid to store the policy information as represented in Figure 1.1.

**Figure 1.1 Insurance Policy Table Using Data Grids**

<table>
<thead>
<tr>
<th>PolicyHolder</th>
<th>Policies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smyth, Joe</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PolicyNumber</td>
</tr>
<tr>
<td></td>
<td>453975R398</td>
</tr>
<tr>
<td></td>
<td>987348P210</td>
</tr>
<tr>
<td></td>
<td>Dupree, Marcel</td>
</tr>
<tr>
<td></td>
<td>PolicyNumber</td>
</tr>
<tr>
<td></td>
<td>983092B228</td>
</tr>
<tr>
<td></td>
<td>274933P412</td>
</tr>
</tbody>
</table>

The information in the Policies column is represented by JavaScript Object Notation (JSON) strings.

A data grid JSON string has the following basic format:

```json
[{
  "metadata": [column-definitions],
  "data": [column-data]
}]
```

The column definitions are name-value pairs separated by commas:

```json
{"column1-name": "data-type"},{"column2-name": "data-type"}, ...
```

The data for each row of the data grid is specified in square brackets with commas between each value:

```json
[column1-data, column2-data ...]
```

For example, if the data grids shown in Figure 1.1 are serialized, the insurance policy table appears as shown in Table 1.2.

**Table 1.2 Serialized Insurance Policy Table**

<table>
<thead>
<tr>
<th>PolicyHolder</th>
<th>Policies</th>
</tr>
</thead>
</table>
| Smyth, Joe   | ["metadata": [{"POLICYNUMBER": "string"},
                  {"YEARLYPREMIUM": "decimal"}],
                 "data": [{"453975R398": 439.50},
                           {"987348P210": 132.90}]} |
|              | Dupree, Marcel |
|              | ["metadata": [{"POLICYNUMBER": "string"},
                  {"YEARLYPREMIUM": "decimal"}],
                 "data": [{"983092B228": 334.00},
                           {"274933P412": 219.35}]} |

If you are processing the data grid column in the insurance policy table, you could define a variable named Policies of type DATAGRID and use the functions described in Chapter 3, “Data Grid Functions,” on page 17 to process the data. For example:

```javascript
DATAGRID_GET(Policies, 'PolicyNumber', 2)
```
For policy holder Joe Smyth, this function call returns 987348P210. In this case, you define variables for only the PolicyHolder and Policies columns. You do not define variables for the PolicyNumber and YearlyPremium columns within the data grid.

If you are processing the data grid column in the insurance policy table by using a rule set or model that iterates through the data grids, you cannot use data grid functions because a data grid cannot contain another data grid. (You specify that a rule set or model iterates through a data grid by selecting Score rows in this data grid. For more information, see “Ways to Work with Data Grids” on page 3.)

### Serialize and Import Data Grids

Note: If you write a serialized data grid variable to a table, the maximum size of the JSON string is based on the engine writes the table. For the Base SAS engine, the limit is 32,767. If you write the serialized data grid variable to a SAS Cloud Analytic Services (CAS) table, the maximum size of the JSON string is determined by the amount of memory that is available. The default size is 10,485,760.

1. Use the %DCM_SERIALIZEGRID macro to serialize your data grid into a JavaScript Object Notation (JSON) string. The data grid must be serialized if it is used in a rule set or in a decision that meets either of the following criteria:
   - The rule set or decision is used in a test or in a publishing validation test.
   - The rule set will be deployed to Hadoop, Teradata, or SAS Cloud Analytic Services (CAS).

   Note: Data grids that are used in decisions that are deployed to SAS Micro Analytic Service are automatically serialized when a request that uses the data grids is sent to the service. If your job is deployed only to SAS Micro Analytic Service, you do not need to use the %DCM_SERIALIZEGRID macro to serialize the data grids.

   Note: The names of data grid column are limited to 255 characters.

2. To combine data from multiple tables into one table, use the %DCM_MERGE_SERIALIZEGRID macro. This macro merges multiple data grids and scalar data into one table based on the values of key columns in each data grid and in the scalar data table.

3. Import the table that contains the serialized data grids as a data source into SAS Intelligent Decisioning. See “Importing Local Files” in SAS Data Explorer: User’s Guide for more information.

### Working with Data Grids

You can publish rule sets and decisions that use data grids to any destination. However, it is unlikely that objects that use data grids can be published to Teradata because of limitations on row sizes.

**Ways to Work with Data Grids**

In a rule set or model, you can work with data grids in two ways:
use data grid functions. For more information, see “Using Data Grid Functions” on page 5.

execute the model or rule set against each row in the data grid. When you are mapping the output variables of a rule set or model, you can specify that the rule set or model executes against the data grid by selecting **Score rows in this data grid** when you map the input and output variables for the rule set or model. For more information, see “Mapping Data Grid Variables in a Rule Set or Model” on page 4.

In custom code files, you use data grid functions to process the data in a data grid. For more information, see “Processing Data Grids in a Code File” on page 6.

**Note:** You cannot use data grids in condition expressions. If you need to use a value calculated from a data grid in a decision condition expression, calculate the value in a rule set, assign the value to an action variable, and then use the action variable in the decision condition expression.

---

**Defining Data Grid Variables**

Data grid variables can be imported, exported, created, edited, and added to rule sets in the same way as other variables. When you are creating or importing data grid variables, the following guidelines apply:

- The columns within a data grid can contain only character or numeric data.

- In the SAS Intelligent Decisioning interface, which variables you define depend on how you are working with the data grid.
  
  - If you are using data grid functions, you define variables for only the table columns that contain the data grids. When you use a data grid function, the function parameter that specifies the name of the data grid column must be either a literal value or a variable that evaluates to the data grid column name.
  
  - If you are executing a model or rule set against each row in the data grid, you define variables for the columns within the data grid.

**Mapping Data Grid Variables in a Rule Set or Model**

By default, each node in a decision is executed for the first record in the input data source, then for the second input record, and so on. However, if you select **Score rows in this data grid** in the **Input Variables** or **Output Variables** property pane for a rule set or model, then the rule set or model is executed against each row of the data grid before execution moves to the next node in the decision. In this case, the rule set or model processes only the data grid. All other output variables are passed through to the next node in the decision.

When you add an object to a decision and the object contains a data grid variable, SAS Intelligent Decisioning creates a decision variable for the data grid in the same way that it creates decision variables for object variables of other data types. When you select **Score rows in this data grid** for a rule set or model that uses a data grid, you can choose to map the columns in the object’s data grid variable either to columns in the decision’s data grid variable or to other decision variables. In the lists of variables in the **Input Variables** and **Output Variables** property panes, the decision’s scalar variables are identified by the icon, and the decision’s data grid columns are identified by the icon.

For more information, see “Map Object Variables to Decision Variables” in **SAS Intelligent Decisioning: User’s Guide**.
**Editing Data Grid Variable Metadata**

For existing data grid variables, you can add or delete columns in the data grid variable.

**Add Columns to a Data Grid**

1. On the **Variables** tab of a rule set or decision, click on the data grid variable that you want to edit. The Edit Variable window appears.
2. Click 📋 to open the Edit Columns window.
3. Add custom columns or add columns from a data source.
   - To add custom columns to the data grid, complete these steps for each column:
     1. Select **Enter a single value to add to the list**, and enter the name of the new column.
     2. Select the data type of the column, and click **Add**.
   - To add columns from a data source, complete these steps:
     1. Select **Select columns from a data source to add to the list**, and click **Browse**. The Choose a Data Source window appears.
     2. Select the data source, and click **OK** to close the Choose a Data Source window.
     3. In the Edit Columns window, click **Add**. SAS Intelligent Decisioning adds all of the columns in the data source to your data grid.
     4. (Optional) Remove any columns from the list that you do not want to include in your data grid.
     4. (Optional) For character string values, enter a length if you do not want to use the default length.
4. Click **OK**.

**Delete Columns from a Data Grid**

1. On the **Variables** tab of a decision, click on the data grid variable that you want to edit. The Edit Variable window appears.
2. Click 📋 to open the Edit Columns window.
3. Click ✕ for each column that you want to delete.
4. Click **OK**.

---

**Using Data Grid Functions**

For rule sets and models that iterate over a data grid, you do not need to use data grid functions. For more information, see “Working with Data Grids” on page 3.

In all other cases, you must use SAS Intelligent Decisioning data grid functions to process data grid variables. SAS Intelligent Decisioning supplies several functions for use with data grids. These functions are described in Chapter 3, “Data Grid Functions,” on page 17.
Note: You can nest function calls. For example:

datagrid_count(datagrid_subsetByValue(assets, 'Asset_Type', 'Savings'))

---

### Processing Data Grids in a Code File

You can create custom DS2 code files that process the values in a data grid and include the code file in a decision. For example, the following DS2 code uses the DATAGRID_COUNT function to determine how many rows are in the data grid, then loops through the data grid to process each row. It uses the DATAGRID_GET function to retrieve the values for the data grid variables. The values of the data grid variables are passed to the assessLoanRequest method.

```ds2
/* Use the DATAGRID_COUNT function to determine the */
/* number of rows that are in the data grid.         */

"numRows" = DATAGRID_COUNT("loanrequests");
if "numRows" > 0 then do;
  do "loopIndex" = 1 to "numRows";
    /* Use the DATAGRID_GET function to retrieve the values */
    /* in the data grid. The assessLoanRequest method invokes */
    /* a published rule set to evaluate each loan request.   */
    assessLoanRequest(DATAGRID_GET("loanrequests","annualSalary","loopIndex"),
                      DATAGRID_GET("loanrequests","incomeThreshold","loopIndex"));
  end;
end;

/* Continue processing the data. */
```


---

### Working with Data Grids in SAS Studio

When you are using data grids in a DS2 custom code file in SAS Intelligent Decisioning, the data grid package DCM_DATAGRID and the data grid functions are predefined. However, these packages and functions are not automatically available when you are using SAS Studio to develop and test code. You can use the %DCM_DATAGRID_IMPL and %DCM_DATAGRID_INTERFACE macros to make data grid functionality available in SAS Studio. For more information, see “%DCM_DATAGRID_INTERFACE” on page 10 and “%DCM_DATAGRID_IMPL” on page 10.

In addition, if your test thread or data program uses any data grid functions, then you need to invoke %DCM_DATAGRID_INTERFACE within the thread or data block, after any global declaration statements. For example:

```sas
data work.cars;
  set sashelp.cars;
run;
```
%dcm_serializegrid(gridSourceTable=work.cars,
   classVars=make,
   outputTable=work.carsByMakeGrid,
   gridColName=carsGrid)

proc ds2;
  /* define data grid DS2 packages */
  %dcm_datagrid_impl()
  package "testCustomCode" /inline;
  /* define data grid interface methods */
  %dcm_datagrid_interface()
  method execute(varchar(32) whichColumn,
      in_out double meanValue,
      in_out package dcm_datagrid thisGrid);
    meanValue = DataGrid_Mean(thisGrid,whichColumn);
  end;
endpackage;

data work.profileByMake(keep=(rowCount make meanMSRP meanWeight carsGrid))
   / overwrite=yes;
  dcl package testCustomCode myCustomCode();
  dcl double meanMSRP;
  dcl double meanWeight;
  dcl varchar(32767) carsGrid;  /* serialize in and out */
  dcl package dcm_datagrid _carsGrid ();
  dcl integer rowCount;
  /* define data grid interface methods */
  %dcm_datagrid_interface()
  method run();
    set work.carsByMakeGrid;
    meanMSRP = .;
    meanWeight = .;
    rowCount = DataGrid_Create(_carsGrid,carsGrid);
    myCustomCode.execute('msrp', meanMSRP, _carsGrid);
    myCustomCode.execute('weight', meanWeight, _carsGrid);
    carsGrid = DataGrid_toString(_carsGrid);
    output work.profileByMake;
  end;
enddata;
run;
quit;
Chapter 2
Data Grid Macros

Using the Data Grid Macros
The SAS Intelligent Decisioning macros are for use in SAS Studio only.

Data Grid Macros Available in SAS Intelligent Decisioning
You can prototype and test custom code that uses data grids in SAS Studio before you add your code to a decision. You can use the following macros to make data grid functionality available in SAS Studio:

- %DCM_DATAGRID_INTERFACE
  makes the data grid functions available in SAS Studio.

- %DCM_DATAGRID_IMPL
  compiles the DCM_DATAGRID package as well as several underlying packages.

  For more information, see “Working with Data Grids in SAS Studio” on page 6.

Data grids must be serialized into JavaScript Object Notation (JSON) strings before you can use the data in SAS Intelligent Decisioning. You can use the following macros to prepare grid data to be used in rules sets:

- %DCM_SERIALIZEGRID
  creates a table in which one of the columns contains a JSON string that represents a data grid.
merges tables that contain data grids that have been serialized with the %DCM_SERIALIZEGRIDS macro with a table that contains scalar data.

## Dictionary

### %DCM_DATAGRID_INTERFACE

Makes the data grid functions available in SAS Studio.

**Syntax**

```
%DCM_DATAGRID_INTERFACE()
```

**Details**

Use the %DCM_DATAGRID_INTERFACE macro together with the %DCM_DATAGRID_IMPL macro to make data grid functions available in SAS Studio. You must invoke this macro within the custom code file package, after any global package declaration statements and before any methods in the package that use data grid functions. In addition, if your test thread or data program uses any data grid functions, then you must invoke the %DCM_DATAGRID_INTERFACE macro within the thread or data block, after any global declaration statements. For more information, see “Working with Data Grids in SAS Studio” on page 6.

```sas
proc ds2;
   %dcm_datagrid_impl()
   data _null_;
      dcl package dcm_datagrid myPackage();
      %dcm_datagrid_interface()
      /* your DS2 code */
   enddata;
run;
quit;
```

### %DCM_DATAGRID_IMPL

Compiles the code necessary to enable you to create instances of the DCM_DATAGRID package.

**Syntax**

```
%DCM_DATAGRID_IMPL()
```
Details

Use the %DCM_DATAGRID_IMPL macro with the %DCM_DATAGRID_INTERFACE macro to make data grid packages available for use in SAS Studio. You must invoke this macro within the DS2 procedure, before the definition of the custom code file package. For more information, see “Working with Data Grids in SAS Studio” on page 6.

```sas
proc ds2;
    %dcm_datagrid_impl()
    package myPackage / overwrite=yes;
    %dcm_datagrid_interface()
    /* your DS2 code */
    endpackage;
run;
quit;
```

%DCM_SERIALIZEGRID

Creates a table in which one of the columns contains a JSON string that represents a data grid.

**Requirement:** If you are working with SAS Cloud Analytic Services (CAS) tables, all of the tables must be accessed in the same CAS session.

**Syntax**

```sas
%DCM_SERIALIZEGRID (
    GRIDCOLNAME=\texttt{data\_grid\_column\_name},
    GRIDSOURCE\_TABLE=\texttt{table\_name},
    OUTPUT\_TABLE=\texttt{results\_table},
    \langle CLASSVARS=class\_variable1<class\_variable2…>\rangle
    \langle GRID\_COLS=column1<column2…>\rangle
    \langle GRID\_COL\_LENGTH=length\rangle
)
```

**Required Arguments**

- **GRIDCOLNAME=\texttt{data\_grid\_column\_name}**
  specifies the name for the serialized data grid column in the results table.

- **GRIDSOURCE\_TABLE=\texttt{table\_name}**
  specifies the table that contains the columns that are to be serialized into a data grid.

- **OUTPUT\_TABLE=\texttt{results\_table}**
  specifies the name of the results table that contains the data grid column.
Optional Arguments

CLASSVARS=class_variable1 <class_variable2...>
specifies the class variables that control how the data is grouped into data grids. If you specify one class variable, the results table contains one data grid for each value of the class variable. If you specify more than one class variable, a separate data grid is created for each combination of values for the class variables. The class variables are written to the output results table.

GRIDCOLS=column1 <column2...>
specifies the names of the columns that are to be serialized into the data grid. If you do not specify this option, all of the columns in the table are serialized except for the columns specified by the CLASSVARS= option.

GRIDCOLLEN=length
specifies the length for the column specified by the GRIDCOLUMN= option. If the results table is a Base SAS table, the length must be less than or equal to 32,767. If the results table is a SAS Cloud Analytic Services (CAS) table, the length must be less than or equal to 10,485,760.

Default 32,767

Example: Serializing the Assets Table

The following example serializes (creates JSON strings) the assets data for each customer in the mylib.assets table. The output table is named assetsGrid, and the data grid column in the output table is named Assets. The data in the mylib.assets table is grouped by the class variable custName, so the output table contains one row for each value of custName.

```
%dcm_serializeGrid(
    gridSourceTable=mylib.assets,
    gridColName=Assets,
    outputTable=assetsGrid,
    classvars=custName);
```

<table>
<thead>
<tr>
<th>CustName</th>
<th>AssetType</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MacKelly, Sara</td>
<td>Property, Primary</td>
<td>212000</td>
</tr>
<tr>
<td>MacKelly, Sara</td>
<td>Property, Investment</td>
<td>125000</td>
</tr>
<tr>
<td>Smyth, Joe</td>
<td>Property, Primary</td>
<td>234500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CustName</th>
<th>Assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>MacKelly, Sara</td>
<td></td>
</tr>
</tbody>
</table>
| ["metadata":{"ASSETTYPE":"string"},"VALUE":"decimal"},
  
  ["Property, Primary",212000],["Property, Investment",125000]]} |

Table 2.1  mylib.assets Source Table

Table 2.2  assetsGrid Results Table
%%DCM_MERGESERIALIZEDGRIDS

Merges tables that contain data grids that have been serialized with the %DCM_SERIALIZEGRIDS macro with a table that contains scalar data.

**Requirement:** If you are working with SAS Cloud Analytic Services (CAS) tables, all of the tables must be accessed in the same CAS session.

**Syntax**

```sas
%DCM_MERGESERIALIZEDGRIDS ( 
   MERGETABLE=table_name, 
   MERGEKEY=merged_table_key, 
   OUTPUTTABLE=results_table, 
   GRIDTABLES=data_grid1 <data_grid2>..., 
   GRIDMERGEKEYS=data_grid_key1 <data_grid_key2...>, 
   GRIDCOLUMNS=data_grid_column1 <data_grid_column2>... )
```

**Required Arguments**

**MERGETABLE=table_name**

specifies the name of the table that contains the scalar data.

**MERGEKEY=merge_table_key**

specifies the name of the key column in the table specified by the MERGETABLE= argument.

*Note:* You can specify only one merge column key. The %DCM_MERGESERIALIZEDGRIDS macro does not support merges based on multiple key values.

**OUTPUTTABLE=results_table**

specifies the name of the table that contains the results of the merge.

**GRIDTABLES=data_grid1 <data_grid2> ...**

specifies the names of the tables that contain the data grid columns that are to be merged into the results table. The data grid columns must contain a serialized data grid produced by the %DCM_SERIALIZEGRID macro.

*Interaction* For each data grid that is to be merged into the results table, the name of the data grid table, merge key, and data grid column must be specified in the same order for each of the GRIDTABLES=, GRIDMERGEKEYS=, and GRIDCOLUMNS= arguments.
GRIDMERGEKEYS=\texttt{data\_grid\_key1 <data\_grid\_key2>} ... 
specifies the names of the key columns in the tables specified by the 
GRIDTABLES= argument.

\textbf{Interaction} The first key specified must be the key for the first table specified in the 
GRIDTABLES= argument, the second key must be the key for the second table, and so on.

GRIDCOLUMNS=\texttt{data\_grid\_column1 <data\_grid\_column2>} ... 
specifies the names of the data grid columns in the tables specified by the 
GRIDTABLES= argument.

\textbf{Interaction} The first column specified must be the name of the data grid column in 
the first table specified in the GRIDTABLES= argument, the second key must be the name of the data grid column in the second table, and so on.

\textbf{Example: Merging Debts and Assets Data Grids with Loan Request Information}

The following example merges the scalar data in the mylib.loanRequests table with the 
data grid columns in the tables \texttt{debtsGrid} and assetGrid. The key column for the scalar table and the data grid tables is custName. The data grid columns in the resulting output table, mylib.loadRequestData, are named Debts and Assets.

```bash
%dcm_mergeSerializedGrids(
    mergeTable=mylib.loanRequests,
    mergekey=Customer,
    outputTable=mylib.loadRequestData,
    gridTables=debtsGrid assetsGrid,
    gridMergeKeys=custName custName
    gridColumns=Debts Assets);
```

\textbf{Table 2.3} \texttt{mylib.loanrequests} Merge Table

<table>
<thead>
<tr>
<th>RequestID</th>
<th>RequestedAmt</th>
<th>Customer</th>
</tr>
</thead>
<tbody>
<tr>
<td>ME4922Mac01</td>
<td>20000</td>
<td>MacKelly, Sara</td>
</tr>
<tr>
<td>NC2W497Smy03</td>
<td>80495</td>
<td>Smyth, Joe</td>
</tr>
</tbody>
</table>

\textbf{Table 2.4} \texttt{debtsGrid} Data Grid Table

<table>
<thead>
<tr>
<th>CustName</th>
<th>Debts</th>
</tr>
</thead>
<tbody>
<tr>
<td>MacKelly, Sara</td>
<td>{&quot;metadata&quot;:{&quot;DEBTTYPE&quot;:&quot;string&quot;},&quot;BALANCE&quot;:decimal}, {&quot;data&quot;:{&quot;Mortgage&quot;,80053.16,&quot;CreditCard&quot;,2143.68}}}</td>
</tr>
</tbody>
</table>
### Table 2.5  assetsGrid Data Grid Table

<table>
<thead>
<tr>
<th>CustName</th>
<th>Assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>MacKelly, Sara</td>
<td>[&quot;metadata&quot;: [{&quot;ASSETTYPE&quot;: &quot;string&quot;}, {&quot;VALUE&quot;: &quot;decimal&quot;}]], &quot;data&quot;: [&quot;Property, Primary&quot;, 212000], [&quot;Property, Investment&quot;, 125000]]</td>
</tr>
<tr>
<td>Smyth, Joe</td>
<td>[&quot;metadata&quot;: [{&quot;ASSETTYPE&quot;: &quot;string&quot;}, {&quot;VALUE&quot;: &quot;decimal&quot;}]], &quot;data&quot;: [&quot;Property, Primary&quot;, 234500]]</td>
</tr>
</tbody>
</table>

### Table 2.6  mylib.loanRequestData Results Table

<table>
<thead>
<tr>
<th>RequestID</th>
<th>RequestAmt</th>
<th>Customer</th>
<th>Debts</th>
<th>Assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>ME4922Mac01</td>
<td>20000</td>
<td>MacKelly, Sara</td>
<td>[&quot;metadata&quot;: [{&quot;DEBTTYPE&quot;: &quot;string&quot;}, {&quot;BALANCE&quot;: &quot;decimal&quot;}]], &quot;data&quot;: [&quot;Mortgage&quot;, 80053.16], [&quot;CreditCard&quot;, 2143.68]]</td>
<td>[&quot;metadata&quot;: [{&quot;ASSETTYPE&quot;: &quot;string&quot;}, {&quot;VALUE&quot;: &quot;decimal&quot;}]], &quot;data&quot;: [&quot;Property, Primary&quot;, 212000], [&quot;Property, Investment&quot;, 125000]]</td>
</tr>
<tr>
<td>NC2497Smy03</td>
<td>80495</td>
<td>Smyth, Joe</td>
<td>.</td>
<td>[&quot;metadata&quot;: [{&quot;ASSETTYPE&quot;: &quot;string&quot;}, {&quot;VALUE&quot;: &quot;decimal&quot;}]], &quot;data&quot;: [&quot;Property, Primary&quot;, 234500]]</td>
</tr>
</tbody>
</table>
# Chapter 3

## Data Grid Functions

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATAGRID_ADDCHARACTERCOLUMN</td>
<td>Add a character column to the data grid</td>
</tr>
<tr>
<td>DATAGRID_ADDNUMERICCOLUMN</td>
<td>Add a numeric column to the data grid</td>
</tr>
<tr>
<td>DATAGRID_ADDROW</td>
<td>Add a row to the data grid</td>
</tr>
<tr>
<td>DATAGRID_APPEND</td>
<td>Append a data grid to another data grid</td>
</tr>
<tr>
<td>DATAGRID_BOTTOMN</td>
<td>Get the bottom N rows of the data grid</td>
</tr>
<tr>
<td>DATAGRID_CLEAR</td>
<td>Clear the data grid</td>
</tr>
<tr>
<td>DATAGRID_CLEARDATA</td>
<td>Clear the data in the data grid</td>
</tr>
<tr>
<td>DATAGRID_COLUMNCOUNT</td>
<td>Get the number of columns in the data grid</td>
</tr>
<tr>
<td>DATAGRID_COLUMNNAME</td>
<td>Get the name of a column in the data grid</td>
</tr>
<tr>
<td>DATAGRID_CONFORM</td>
<td>Conform the data grid to a specified format</td>
</tr>
<tr>
<td>DATAGRID_COPY</td>
<td>Copy the data grid</td>
</tr>
<tr>
<td>DATAGRID_CORR</td>
<td>Calculate the correlation of columns in the data grid</td>
</tr>
<tr>
<td>DATAGRID_COUNT</td>
<td>Count the number of rows in the data grid</td>
</tr>
<tr>
<td>DATAGRID_CREATE</td>
<td>Create a data grid</td>
</tr>
<tr>
<td>DATAGRID_DELETECOLUMN</td>
<td>Delete a column from the data grid</td>
</tr>
<tr>
<td>DATAGRID_DELETEROW</td>
<td>Delete a row from the data grid</td>
</tr>
<tr>
<td>DATAGRID_DISTINCTROWCOUNT</td>
<td>Get the number of distinct rows in the data grid</td>
</tr>
<tr>
<td>DATAGRID_FILTEREDGET</td>
<td>Get the filtered data grid</td>
</tr>
<tr>
<td>DATAGRID_FILTEREDSET</td>
<td>Set a data grid based on a filter</td>
</tr>
<tr>
<td>DATAGRID_FILTEREDSETALL</td>
<td>Set all data grids based on a filter</td>
</tr>
<tr>
<td>DATAGRID_FREQ</td>
<td>Calculate the frequency of values in the data grid</td>
</tr>
<tr>
<td>DATAGRID_FULLJOIN</td>
<td>Perform a full join on the data grids</td>
</tr>
<tr>
<td>DATAGRID_GET</td>
<td>Get the data grid</td>
</tr>
<tr>
<td>DATAGRID_GRIDMATCHCOUNT</td>
<td>Get the number of matches between two data grids</td>
</tr>
<tr>
<td>DATAGRID_INNERJOIN</td>
<td>Perform an inner join on the data grids</td>
</tr>
<tr>
<td>DATAGRID_LEFTJOIN</td>
<td>Perform a left join on the data grids</td>
</tr>
<tr>
<td>DATAGRID_MATCHCOUNT</td>
<td>Get the number of matches between a column and a pattern</td>
</tr>
<tr>
<td>DATAGRID_MAX</td>
<td>Get the maximum value in the data grid</td>
</tr>
<tr>
<td>DATAGRID_MEAN</td>
<td>Calculate the mean of the values in the data grid</td>
</tr>
<tr>
<td>DATAGRID_MEDIAN</td>
<td>Calculate the median of the values in the data grid</td>
</tr>
<tr>
<td>DATAGRID_MIN</td>
<td>Get the minimum value in the data grid</td>
</tr>
<tr>
<td>DATAGRID_NMISS</td>
<td>Get the number of missing values in the data grid</td>
</tr>
<tr>
<td>DATAGRID_NVALID</td>
<td>Get the number of valid values in the data grid</td>
</tr>
<tr>
<td>DATAGRID_RENAMECOLUMN</td>
<td>Rename a column in the data grid</td>
</tr>
<tr>
<td>DATAGRID_RIGHTJOIN</td>
<td>Perform a right join on the data grids</td>
</tr>
<tr>
<td>DATAGRID_SET</td>
<td>Set a data grid</td>
</tr>
<tr>
<td>DATAGRID_SETALL</td>
<td>Set all data grids</td>
</tr>
</tbody>
</table>

**Comparison Operators**

Comparison operators allow you to compare values in the data grid. They are crucial for filtering and sorting data based on specific conditions.
Comparison Operators

You can use any of the following operators with functions that compare values such as the DATAGRID_FILTEREDGET and DATAGRID_MATCHCOUNT functions.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQ, =, ==</td>
<td>Equals</td>
</tr>
<tr>
<td>NE, !=, ^=, &lt;&gt;</td>
<td>Not equal to</td>
</tr>
<tr>
<td>GT, &gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>LT, &lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>LE, &lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>GE, &gt;=</td>
<td>Greater than or equal to</td>
</tr>
</tbody>
</table>

Data Grid Functions Available in SAS Intelligent Decisioning

Functions are categorized by the types of values that they return or type of operation they perform. Each function belongs to one of the following categories:

Create, Update, or Delete
Create, copy, update, or clear the data in a data grid.

Information
Return information about an entire data grid or about a column or row in the data grid.

Join or Append
Join or append two data grids.

Rename
Rename columns in a data grid.

Retrieve Values
Retrieve values from a data grid.
Serialize
Serialize a data grid into a JSON string.

Set Values
Set the value of cells in a data grid.

Statistical
Perform statistical calculations on values in the data grid.

Subset and Sort
Subset or sort a data grid.

The following table provides brief descriptions of the data grid functions.

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create, Update, or Delete</td>
<td>DATAGRID_ADDCHARACTERCOLUMN (p. 21)</td>
<td>Adds a character column to the specified data grid.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_ADDNUMERICCOLUMN (p. 22)</td>
<td>Adds a numeric column to the specified data grid.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_ADDROW (p. 22)</td>
<td>Appends a new row to the specified data grid.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_CLEAR (p. 24)</td>
<td>Deletes all rows and all column metadata from the specified data grid.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_CLEARDATA (p. 24)</td>
<td>Deletes all rows from the specified data grid but does not remove the column metadata.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_CONFORM (p. 25)</td>
<td>Adds the columns that are exclusive to dataGrid1 to dataGrid2, and adds the columns that are exclusive to dataGrid2 to dataGrid1.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_COPY (p. 26)</td>
<td>Copies the source data grid into the target data grid.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_CREATE (p. 27)</td>
<td>Creates a data grid from the specified JSON string.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_DELETECOLUMN (p. 28)</td>
<td>Deletes the specified column from the specified data grid.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_DELETEROW (p. 29)</td>
<td>Deletes the specified row from the specified data grid.</td>
</tr>
<tr>
<td>Information</td>
<td>DATAGRID_COLUMNCOUNT (p. 25)</td>
<td>Returns the number of columns in the specified data grid.</td>
</tr>
<tr>
<td></td>
<td>DATAGRIDCOLUMNNAME (p. 25)</td>
<td>Returns the name of the column that is in the specified ordinal position in the specified data grid.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_COUNT (p. 27)</td>
<td>Returns the number of rows in the specified data grid.</td>
</tr>
<tr>
<td></td>
<td>DATAGRIDDISTINCTROWCOUNT (p. 29)</td>
<td>Returns the number of unique rows in the specified data grid.</td>
</tr>
<tr>
<td>Category</td>
<td>Language Elements</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_GRIDMATCHCOUNT(p. 35)</td>
<td>Returns the number of rows for which the value in the specified column in one data grid matches the value in the specified column in another data grid.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_MATCHCOUNT(p. 37)</td>
<td>Returns the number of rows in the specified column for which the specified comparison evaluates to true.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_NVALID(p. 40)</td>
<td>Returns the number of valid nonmissing values in the specified column of the specified data grid.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_ADDROW(p. 22)</td>
<td>Appends a new row to the specified data grid.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_APPEND(p. 23)</td>
<td>Appends dataGrid2 to dataGrid1.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_FULLJOIN(p. 34)</td>
<td>Performs a full join of two data grids and populates the target data grid with the results of the join.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_INNERJOIN(p. 36)</td>
<td>Performs an inner join of two data grids and populates the target data grid with the results of the join.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_LEFTJOIN(p. 36)</td>
<td>Performs a left join of two data grids, returns the resulting data grid, and populates the target data grid with the results of the join.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_RIGHTJOIN(p. 41)</td>
<td>Performs a right join of two data grids, returns the resulting data grid, and populates the target data grid with the results of the join.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_RENAMECOLUMN(p. 40)</td>
<td>Renames the specified column in the specified data grid.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_FILTEREDGET(p. 30)</td>
<td>Returns the value in the first row in the specified column for which the specified comparison evaluates to true.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_GET(p. 34)</td>
<td>Returns the value of the cell in the specified row and column.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_TOSTRING(p. 46)</td>
<td>Returns the JSON string for the specified data grid.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_FILTEREDSET(p. 31)</td>
<td>Sets the value in the specified row and column to the specified value if the specified comparison evaluates to true.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_FILTEREDSETALL (p. 32)</td>
<td>Sets the cell in the specified column to the specified value for each row for which the specified comparison evaluates to true.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_SET(p. 42)</td>
<td>Assigns the specified value to the specified row and column.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_SETALL(p. 42)</td>
<td>Assigns the specified value to all rows in the specified column.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_CORR(p. 26)</td>
<td>Returns the Pearson product-moment correlation coefficient for the specified columns in the specified data grid.</td>
</tr>
<tr>
<td></td>
<td>DATAGRID_FREQ(p. 33)</td>
<td>Returns the number of distinct values for the specified column.</td>
</tr>
</tbody>
</table>
**DATAGRID_ADDCHARACTERCOLUMN**

Adds a character column to the specified data grid.

<table>
<thead>
<tr>
<th>Category</th>
<th>Language Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>DATAGRID_MAX</strong> (p. 38)</td>
<td>Returns the maximum value for the specified column.</td>
</tr>
<tr>
<td></td>
<td><strong>DATAGRID_MEAN</strong> (p. 38)</td>
<td>Returns the mean value for the specified column.</td>
</tr>
<tr>
<td></td>
<td><strong>DATAGRID_MEDIAN</strong> (p. 39)</td>
<td>Returns the median value for the specified column.</td>
</tr>
<tr>
<td></td>
<td><strong>DATAGRID_MIN</strong> (p. 39)</td>
<td>Returns the minimum value that appears in the specified column.</td>
</tr>
<tr>
<td></td>
<td><strong>DATAGRID_NMISS</strong> (p. 40)</td>
<td>Returns the number of missing values for the specified column of the specified data grid.</td>
</tr>
<tr>
<td></td>
<td><strong>DATAGRID_STDDEV</strong> (p. 43)</td>
<td>Returns the standard deviation of the values in the specified column.</td>
</tr>
<tr>
<td></td>
<td><strong>DATAGRID_SUM</strong> (p. 45)</td>
<td>Returns the sum of the values in the specified column.</td>
</tr>
<tr>
<td>Subset and Sort</td>
<td><strong>DATAGRID_BOTTOMN</strong> (p. 23)</td>
<td>Populates the target data grid with the rows from the source data grid that contain the lowest <em>number</em> values in the specified column.</td>
</tr>
<tr>
<td></td>
<td><strong>DATAGRID_SORT</strong> (p. 43)</td>
<td>Populates the target data grid with a sorted copy of the source data grid.</td>
</tr>
<tr>
<td></td>
<td><strong>DATAGRID_SUBSETBYVALUE</strong> (p. 44)</td>
<td>Populates the target data grid with the rows from the source data grid for which the specified comparison evaluates to true.</td>
</tr>
<tr>
<td></td>
<td><strong>DATAGRID_TOPN</strong> (p. 45)</td>
<td>Populates the target data grid with the rows from the source data grid that contain the highest <em>number</em> values in the specified column.</td>
</tr>
</tbody>
</table>

**Dictionary**

**DATAGRID_ADDCHARACTERCOLUMN**

Adds a character column to the specified data grid.

- **Category:** Create, Update, or Delete
- **Returned data type:** INTEGER
- **Note:** This function returns the number of columns that are in the data grid after the new column has been added.

**Syntax**

```
DATAGRID_ADDCHARACTERCOLUMN (dataGrid, column)
```
**Required Arguments**

*dataGrid*

specifies the name of the data grid. This argument must be a variable of type DATAGRID.

*column*

specifies the name of the column that you want to create. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

---

**DATAGRID_ADDNUMERICCOLUMN**

Adds a numeric column to the specified data grid.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Create, Update, or Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returned data type:</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Note:</td>
<td>This function returns the number of columns that are in the data grid after the new column has been added.</td>
</tr>
</tbody>
</table>

**Syntax**

```
DATAGRID_ADDNUMERICCOLUMN (dataGrid, column)
```

**Required Arguments**

*dataGrid*

specifies the name of the data grid. This argument must be a variable of type DATAGRID.

*column*

specifies the name of the column that you want to create. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

---

**DATAGRID_ADDROW**

Appends a new row to the specified data grid.

<table>
<thead>
<tr>
<th>Categories:</th>
<th>Create, Update, or Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returned data type:</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Note:</td>
<td>This function returns the number of rows that are in the data grid after the new row has been added.</td>
</tr>
</tbody>
</table>

**Syntax**

```
DATAGRID_ADDROW (dataGrid)
```
**DATAGRID_APPEND**

Appends `dataGrid2` to `dataGrid1`.

**Category:** Join or Append

**Requirement:** The two data grids must contain the same columns. You can use the `DATAGRID_CONFORM` function to make the two data grids have the same columns.

**Returned data type:** INTEGER

**Note:** This function returns the number of rows that are in the data grid after the append.

**Syntax**

```
DATAGRID_APPEND (dataGrid1, dataGrid2)
```

**Required Argument**

- `dataGrid1`
- `dataGrid2`

specifies the names of the two data grids.

---

**DATAGRID_BOTTOMN**

Populates the target data grid with the rows from the source data grid that contain the lowest `number` values in the specified column.

**Category:** Subset and Sort

**Returned data type:** INTEGER

**Note:** This function returns the number of rows in the target data grid. The target data grid is sorted in ascending order of the specified column.

**Syntax**

```
DATAGRID_BOTTOMN (source-dataGrid, column, number, target-dataGrid)
```

**Required Arguments**

- `source-dataGrid`

specifies the name of the source data grid. This argument must be a variable of type DATAGRID.
column
specifies the name of the column in the data grid. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

number
specifies the number of rows to return.

target-dataGrid
specifies the name of the target data grid. This argument must be a variable of type DATAGRID. Any existing data in the target data grid is overwritten.

DATAGRID_CLEAR
Deletes all rows and all column metadata from the specified data grid.

Category: Create, Update, or Delete
Returned data type: INTEGER
Note: This function always returns a zero.

Syntax
DATAGRID_CLEAR (dataGrid)

Required Argument
dataGrid
  specifies the name of the data grid. This argument must be a variable of type DATAGRID.

DATAGRID_CLEARDATA
Deletes all rows from the specified data grid but does not remove the column metadata.

Category: Create, Update, or Delete
Returned data type: INTEGER
Note: This function always returns a zero.

Syntax
DATAGRID_CLEARDATA (dataGrid)

Required Argument
dataGrid
  specifies the name of the data grid. This argument must be a variable of type DATAGRID.
DATAGRID_COLUMNCOUNT

Returns the number of columns in the specified data grid.

Category: Information
Returned data type: INTEGER

Syntax

DATAGRID_COLUMNCOUNT (dataGrid)

Required Argument
dataGrid
specifies the name of the data grid. This argument must be a variable of type DATAGRID.

DATAGRID_COLUMNNAME

Returns the name of the column that is in the specified ordinal position in the specified data grid.

Category: Information
Returned data type: CHARACTER

Syntax

DATAGRID_COLUMNNAME (dataGrid, column_number)

Required Arguments
dataGrid
specifies the name of the data grid. This argument must be a variable of type DATAGRID.
column_number
specifies the column number in the data grid. The columns in a data grid are numbered beginning with 1.

DATAGRID_CONFORM

Adds the columns that are exclusive to dataGrid1 to dataGrid2, and adds the columns that are exclusive to dataGrid2 to dataGrid1.

Category: Create, Update, or Delete
Returned data type: INTEGER
**Note:** This function always returns a zero.

**Syntax**

\[ \text{DATAGRID\_CONFORM} \left( \text{dataGrid1}, \text{dataGrid2} \right) \]

**Required Argument**

- \( \text{dataGrid1} \)
- \( \text{dataGrid2} \)

specifies the names of the two data grids that you want to contain the same columns.

---

**DATAGRID\_COPY**

Copies the source data grid into the target data grid.

**Category:** Create, Update, or Delete

**Returned data type:** INTEGER

**Note:** This function returns the number of rows in the target data grid.

**Syntax**

\[ \text{DATAGRID\_COPY} \left( \text{source-dataGrid}, \text{target-dataGrid} \right) \]

**Required Arguments**

- \( \text{source-dataGrid} \)
- \( \text{target-dataGrid} \)

specifies the name of the source data grid. This argument must be a variable of type DATAGRID.

specifies the name of the target data grid. This argument must be a variable of type DATAGRID. Any existing data in the target data grid is overwritten.

---

**DATAGRID\_CORR**

Returns the Pearson product-moment correlation coefficient for the specified columns in the specified data grid.

**Category:** Statistical

**Returned data type:** DOUBLE

**Syntax**

\[ \text{DATAGRID\_CORR} \left( \text{dataGrid}, \text{column1}, \text{column2} \right) \]
**Required Arguments**

*dataGrid*

specifies the name of the data grid. This argument must be a variable of type DATAGRID.

*column1, column2*

specifies the numeric data grid columns for which you want to compute the correlation coefficient.

**Details**

The Pearson product-moment correlation is a parametric measure of association for two variables. It measures both the strength and the direction of a linear relationship. If one variable X is an exact linear function of another variable Y, a positive relationship exists if the correlation is 1, and a negative relationship exists if the correlation is –1. If there is no linear predictability between the two variables, the correlation is 0. If the two variables jointly have a normal distribution with a zero correlation, the two variables are independent. However, correlation does not imply causality because, in some cases, an underlying causal relationship might not exist.

The formula for the Pearson product-moment correlation, denoted by \( \rho_{xy} \), is as follows:

\[
\rho_{xy} = \frac{\text{Cov}(x, y)}{\sqrt{\text{V}(x)\text{V}(y)}} = \frac{\mathbb{E}(x - \mathbb{E}(x))(y - \mathbb{E}(y))}{\sqrt{\mathbb{E}(x - \mathbb{E}(x))^2\mathbb{E}(y - \mathbb{E}(y))^2}}
\]

---

**DATAGRID_COUNT**

Returns the number of rows in the specified data grid.

**Category:** Information  
**Returned data type:** INTEGER

**Syntax**

**DATAGRID_COUNT** *(dataGrid)*

**Required Argument**

*dataGrid*

specifies the name of the data grid. This argument must be a variable of type DATAGRID.

---

**DATAGRID_CREATE**

Creates a data grid from the specified JSON string.

**Category:** Create, Update, or Delete  
**Returned data type:** DATAGRID
DATAGRID_CREATE (dataGrid, JSON_string)

Required Arguments

$dataGrid$

specifies the name of the data grid. This argument must be a variable of type DATAGRID.

$JSON_string$

specifies a JSON character string that contains the data for the data grid. You can specify a literal value in single quotation marks, or you can specify a variable that evaluates to a JSON character string. You can use the %DCM_SERIALIZEGRID macro to generate the JSON string for a data grid. See “%DCM_SERIALIZEGRID” on page 11 for more information.

The JSON string has the following format:

\[
\{
\text{"metadata":\{"column1":"data_type"},\{"column2":"data_type"\},...\}},
\text{\{"data":\{[row1_data],[row2_data]...\}\}}
\]

column1
column2

specifies the column names of each column in the data grid.

row1_data
row2_data

specifies the data for each row in the data grid. Separate the values for each column in a row with a comma (.). Enclose character values in double quotation marks.

See “Introduction to Data Grids” on page 1

“Example: Serializing the Assets Table” on page 12

DATAGRID_DELETECOLUMN

Deletes the specified column from the specified data grid.

Category: Create, Update, or Delete

Returned data type: INTEGER

Note: This function returns a 1 if the column does not exist and a 0 if the column is deleted.

Syntax

DATAGRID_DELETECOLUMN (dataGrid, column)
**Required Arguments**

*dataGrid*

specifies the name of the data grid. This argument must be a variable of type DATAGRID.

*column*

specifies the name of the column in the data grid. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

---

**DATAGRID_DELETEROW**

Deletes the specified row from the specified data grid.

**Category:** Create, Update, or Delete

**Returned data type:** INTEGER

**Note:** This function returns the number of rows remaining in the data grid.

**Syntax**

`DATAGRID_DELETEROW (dataGrid, row_number)`

**Required Arguments**

*dataGrid*

specifies the name of the data grid. This argument must be a variable of type DATAGRID.

*row_number*

specifies the number of the row to delete.

---

**DATAGRID_DISTINCTROWCOUNT**

Returns the number of unique rows in the specified data grid.

**Category:** Information

**Returned data type:** INTEGER

**Syntax**

`DATAGRID_DISTINCTROWCOUNT (dataGrid)`

**Required Argument**

*dataGrid*

specifies the name of the data grid. This argument must be a variable of type DATAGRID.
DATAGRID_FILTEREDGET

Returns the value in the first row in the specified column for which the specified comparison evaluates to true.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Retrieve Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returned data type:</td>
<td>STRING, DOUBLE</td>
</tr>
</tbody>
</table>

Syntax

DATAGRID_FILTEREDGET(dataGrid, columnReturned, filterColumn, operator, variableOrValue)

**Required Arguments**

*dataGrid*

specifies the name of the data grid. This argument must be a variable of type DATAGRID.

*columnReturned*

specifies the name of the column whose value you want to know. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

*filterColumn*

specifies the name of the column whose value is to be compared to variableOrValue. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

*operator*

specifies one of the following operators shown in “Comparison Operators” on page 18. You can specify the name of a character variable that evaluates to one of these operators, or you can specify the operator enclosed in single quotation marks.

*variableOrValue*

specifies the value to compare to the value of filterColumn. You can specify a number, a character string enclosed in single quotation marks, the name of a variable or of an expression. The value that you specify must be or must evaluate to the same data type as filterColumn.

Details

The DATAGRID_FILTEREDGET function compares the value in filterColumn to the specified variableOrValue by using the comparison operator. The comparison is as follows:

filterColumn operator variableOrValue

For the first row for which the comparison evaluates to true, this function returns the value of columnReturned.
Example

The following example determines whether the value of the riskGroup column is equal to \textit{Low}, and if so, returns the value of the approvalStatus column:

\begin{verbatim}
DATAGRID_FILTEREDGET(DebtEval,'approvalStatus','riskGroup','EQ','Low')
\end{verbatim}

\textbf{DATAGRID_FILTEREDSET}

Sets the value in the specified row and column to the specified value if the specified comparison evaluates to true.

- **Category:** Set Values
- **Returned data type:** INTEGER

\textbf{Syntax}

\begin{verbatim}
DATAGRID_FILTEREDSET(dataGrid, columnToAssign, rowNumber, filterColumn, operator, variableOrValue, valueToAssign)
\end{verbatim}

\textbf{Required Arguments}

- **dataGrid**
  - specifies the name of the data grid. This argument must be a variable of type DATAGRID.

- **columnToAssign**
  - specifies the name of the column to be assigned the specified value. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

- **rowNumber**
  - specifies the row number that contains the cell to be assigned the specified value. You can specify a number or a variable that evaluates to a number.

- **filterColumn**
  - specifies the name of the column whose value is to be compared to \textit{variableOrValue}. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

- **operator**
  - specifies one of the following operators shown in “Comparison Operators” on page 18. You can specify the name of a character variable that evaluates to one of these operators, or you can specify the operator enclosed in single quotation marks.

- **variableOrValue**
  - specifies the value to compare to the value of \textit{filterColumn}. You can specify a number, a character string enclosed in single quotation marks, the name of a variable or of an expression. The value that you specify must be or must evaluate to the same data type as \textit{filterColumn}.

- **valueToAssign**
  - specifies the value to assign to the cell at \textit{rowNumber,columnToAssign}. You can specify a number, a character value enclosed in single quotation marks, or the name of a variable.
Details

The DATAGRID_FILTEREDSET function compares the value of filterColumn to the value of variableOrValue using the comparison operator. The comparison is as follows:

\[
\text{filterColumn \ operator \ variableOrValue}
\]

If the comparison evaluates to true, this function sets the value of rowNumber,columnToAssign to the value specified by valueToAssign.

Example

For the data grid row specified by the value of the customer variable, the DATAGRID_FILTEREDSET function sets the column riskGroup to High if the value of the Debts column is greater than the value of assets variable:

\[
\text{DATAGRID_FILTEREDSET(DebtEval,'riskGroup',customer,'debts','GT',assets,'High')}
\]

---

**DATAGRID_FILTEREDSETALL**

Sets the cell in the specified column to the specified value for each row for which the specified comparison evaluates to true.

- **Category:** Set Values
- **Returned data type:** INTEGER
- **Note:** This function returns the number of cells that were set to the specified value.

Syntax

\[
\text{DATAGRID_FILTEREDSETALL(dataGrid, columnToAssign, filterColumn, operator, variableOrValue, valueToAssign)}
\]

**Required Arguments**

- **dataGrid** specifies the name of the data grid. This argument must be a variable of type DATAGRID.

- **columnToAssign** specifies the name of the column to be assigned the specified value. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

- **filterColumn** specifies the name of the column whose value is to be compared to variableOrValue. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

- **operator** specifies one of the following operators shown in “Comparison Operators” on page 18. You can specify the name of a character variable that evaluates to one of these operators, or you can specify the operator enclosed in single quotation marks.
variableOrValue
specifies the value to compare to the value of filterColumn. You can specify a number, a character string enclosed in single quotation marks, the name of a variable or of an expression. The value that you specify must be or must evaluate to the same data type as filterColumn.

valueToAssign
specifies the value to assign to the appropriate cells in columnToAssign. You can specify a number, a character value enclosed in single quotation marks, the name of a variable, or an expression. The value that you specify must be or must evaluate to the same data type as columnToAssign.

Details
For each row in the data grid, the DATAGRID_FILTEREDSETALL function compares the value of filterColumn to the value of variableOrValue by using the comparison operator. The comparison is as follows:

   filterColumn operator variableOrValue

If the comparison evaluates to true, this function sets the value of the cell in column columnToAssign to the value specified by valueToAssign.

Example
This example sets the value of the cell in column riskGroup to the value of the highGroup variable for all rows in the data grid DebtEval for which the value of the Debts column is greater than the value of the assets variable:

   DATAGRID_FILTEREDSETALL(DebtEval,'riskGroup','debts','GT',assets,highGroup)

---

**DATAGRID_FREQ**

Returns the number of distinct values for the specified column.

- **Category:** Statistical
- **Returned data type:** INTEGER

**Syntax**

   DATAGRID_FREQ(dataGrid, column)

**Required Arguments**

- **dataGrid**
  specifies the name of the data grid. This argument must be a variable of type DATAGRID.

- **column**
  specifies the name of the column in the data grid. You can specify a column name in single quotation marks or a variable that evaluates to a column name.
**DATAGRID_FULLJOIN**

Performs a full join of two data grids and populates the target data grid with the results of the join.

**Category:** Join or Append  
**Returned data type:** INTEGER  
**Note:** This function returns the number of rows in the target data grid.

**Syntax**

`DATAGRID_FULLJOIN(dataGrid1, dataGrid2, keyColumn1, keyColumn2, target-dataGrid)`

**Required Arguments**

- `dataGrid1`  
- `dataGrid2`  
  - specifies the names of the two data grids to be joined.

- `keyColumn1`  
- `keyColumn2`  
  - specifies the name of the key column in `dataGrid1`. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

- `target-dataGrid`  
  - specifies the name of the target data grid. This argument must be a variable of type DATAGRID. Any existing data in the target data grid is overwritten.

**Details**

The DATAGRID_FULLJOIN function performs a full outer join of two data grids for which `dataGrid1.keyColumn1` equals `dataGrid2.keyColumn2` and populates the target data grid with the results of the join. The target data grid includes all rows from both `dataGrid1` and `dataGrid2`. Unmatched rows are preserved.

When the column names in the two data grids are identical, the columns from `dataGrid1` are added to the target data grid.

---

**DATAGRID_GET**

Returns the value of the cell in the specified row and column.

**Category:** Retrieve Values  
**Returned data type:** STRING, DOUBLE
Syntax

DATAGRID_GET \( (dataGrid, column, rowNumber) \)

**Required Arguments**

*dataGrid*

specifies the name of the data grid. This argument must be a variable of type DATAGRID.

*column*

specifies the name of the column in the data grid. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

*rowNumber*

specifies the row number in the data grid. You can specify a number or a variable that evaluates to a number.

---

DATAGRID_GRIDMATCHCOUNT

Returns the number of rows for which the value in the specified column in one data grid matches the value in the specified column in another data grid.

**Category:** Information

**Returned data type:** INTEGER

Syntax

DATAGRID_GRIDMATCHCOUNT \( (dataGrid1, dataGrid2, column1, column2) \)

**Required Arguments**

*dataGrid1*

specifies the names of data grids. These arguments must be variables of type DATAGRID.

*column1*

specifies the name of the column in dataGrid1 that you want to compare to a column in dataGrid2. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

*column2*

specifies the name of the column in dataGrid2. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

**Details**

The DATAGRID_GRIDMATCHCOUNT function returns the number of rows in which the value of dataGrid1, column1 match the value of dataGrid2, column2.
**DATAGRID_INNERJOIN**

Performs an inner join of two data grids and populates the target data grid with the results of the join.

**Category:** Join or Append

**Returned data type:** INTEGER

**Note:** This function returns the number of rows in the target data grid.

**Syntax**

```
DATAGRID_INNERJOIN(dataGrid1, dataGrid2, keyColumn1, keyColumn2, targetDataGrid)
```

**Required Arguments**

- `dataGrid1`
- `dataGrid2`
  - Specifies the names of the two data grids to be joined.
- `keyColumn1`
  - Specifies the name of the key column in `dataGrid1`. You can specify a column name in single quotation marks or a variable that evaluates to a column name.
- `keyColumn2`
  - Specifies the name of the key column in `dataGrid2`. You can specify a column name in single quotation marks or a variable that evaluates to a column name.
- `targetDataGrid`
  - Specifies the name of the target data grid. This argument must be a variable of type DATAGRID. Any existing data in the target data grid is overwritten.

**Details**

The DATAGRID_INNERJOIN function performs an inner join of two data grids for which `dataGrid1.keyColumn1` equals `dataGrid2.keyColumn2` and populates the target data grid with the results of the join. The target table includes all rows from `dataGrid1` that match rows in `dataGrid2`. Unmatched rows from both data grids are discarded.

When the column names in the two data grids are identical, the columns from the left side of the join are added to the resulting data grid.

---

**DATAGRID_LEFTJOIN**

Performs a left join of two data grids, returns the resulting data grid, and populates the target data grid with the results of the join.

**Category:** Join or Append

**Returned data type:** INTEGER

**Note:** This function returns the number of rows in the target data grid.
Syntax

**DATAGRID_LEFTJOIN** (dataGrid1, dataGrid2, keyColumn1, keyColumn2, target-dataGrid)

**Required Arguments**

- **dataGrid1**
  - specifies the names of the two data grids to be joined.

- **dataGrid2**
  - specifies the name of the key column in `dataGrid1`. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

- **keyColumn1**
  - specifies the name of the key column in `dataGrid2`. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

- **keyColumn2**
  - specifies the name of the target data grid. This argument must be a variable of type DATAGRID. Any existing data in the target data grid is overwritten.

**Details**

The **DATAGRID_LEFTJOIN** function performs a left join of two data grids for which `dataGrid1.keyColumn1` equals `dataGrid2.keyColumn2` and populates the target data grid with the results of the join. The target data grid contains all rows from `dataGrid1` plus the matching rows from `dataGrid2`.

When the column names in the two data grids are identical, the columns from the left side of the join are added to the resulting data grid.

---

**DATAGRID_MATCHCOUNT**

Returns the number of rows in the specified column for which the specified comparison evaluates to true.

<table>
<thead>
<tr>
<th><strong>Category:</strong></th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Returned data type:</strong></td>
<td>INTEGER</td>
</tr>
</tbody>
</table>

**Syntax**

**DATAGRID_MATCHCOUNT** (dataGrid, filterColumn, operator, variableOrValue)

**Required Arguments**

- **dataGrid**
  - specifies the name of the data grid. This argument must be a variable of type DATAGRID.

- **filterColumn**
  - specifies the name of the column whose value is to be compared to `variableOrValue`. You can specify a column name in single quotation marks or a variable that evaluates to a column name.
operator
specifies one of the following operators shown in “Comparison Operators” on page 18. You can specify the name of a character variable that evaluates to one of these operators, or you can specify the operator enclosed in single quotation marks.

variableOrValue
specifies the value to compare to the value of filterColumn. You can specify a number, a character string enclosed in single quotation marks, the name of a variable or of an expression. The value that you specify must be or must evaluate to the same data type as filterColumn.

DATAGRID_MAX
Returns the maximum value for the specified column.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Statistical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returned data type:</td>
<td>DOUBLE</td>
</tr>
</tbody>
</table>

**Syntax**

DATAGRID_MAX (dataGrid, column)

**Required Arguments**

dataGrid
specifies the name of the data grid. This argument must be a variable of type DATAGRID.

column
specifies the name of the column in the data grid. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

DATAGRID_MEAN
Returns the mean value for the specified column.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Statistical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returned data type:</td>
<td>DOUBLE</td>
</tr>
</tbody>
</table>

**Syntax**

DATAGRID_MEAN (dataGrid, column)

**Required Arguments**

dataGrid
specifies the name of the data grid. This argument must be a variable of type DATAGRID.
column
specifies the name of the column in the data grid. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

---

**DATAGRID_MEDIAN**

Returns the median value for the specified column.

- **Category:** Statistical
- **Returned data type:** DOUBLE

**Syntax**

`DATAGRID_MEDIAN(dataGrid, column)`

**Required Arguments**

- **dataGrid**
  - specifies the name of the data grid. This argument must be a variable of type DATAGRID.
- **column**
  - specifies the name of the column in the data grid. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

---

**DATAGRID_MIN**

Returns the minimum value that appears in the specified column.

- **Category:** Statistical
- **Returned data type:** DOUBLE

**Syntax**

`DATAGRID_MIN(dataGrid, column)`

**Required Arguments**

- **dataGrid**
  - specifies the name of the data grid. This argument must be a variable of type DATAGRID.
- **column**
  - specifies the name of the column in the data grid. You can specify a column name in single quotation marks or a variable that evaluates to a column name.
**DATAGRID_NMISS**

Returns the number of missing values for the specified column of the specified data grid.

**Syntax**

`DATAGRID_NMISS(dataGrid, column)`

**Required Arguments**

- `dataGrid` specifies the name of the data grid. This argument must be a variable of type `DATAGRID`.
- `column` specifies the name of the column in the data grid. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

---

**DATAGRID_NVALID**

Returns the number of valid nonmissing values in the specified column of the specified data grid.

**Syntax**

`DATAGRID_NVALID(dataGrid, column)`

**Required Arguments**

- `dataGrid` specifies the name of the data grid. This argument must be a variable of type `DATAGRID`.
- `column` specifies the name of the column in the data grid. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

---

**DATAGRID_RENAMECOLUMN**

Renames the specified column in the specified data grid.

**Category:** Rename
**DATAGRID_RENAMECOLUMN**

*Syntax*

\[ \text{DATAGRID_RENAMECOLUMN} \left( \text{dataGrid}, \text{oldName}, \text{newName} \right) \]

*Required Arguments*

- **dataGrid**: specifies the name of the data grid. This argument must be a variable of type DATAGRID.
- **oldName**: specifies the existing name of the column.
- **newName**: specifies the new name of the column.

---

**DATAGRID_RIGHTJOIN**

Performs a right join of two data grids, returns the resulting data grid, and populates the target data grid with the results of the join.

*Category*: Join or Append

*Returned data type*: INTEGER

*Note*: This function returns the number of rows in the target data grid.

*Syntax*

\[ \text{DATAGRID_RIGHTJOIN} \left( \text{dataGrid1}, \text{dataGrid2}, \text{keyColumn1}, \text{keyColumn2}, \text{target-dataGrid} \right) \]

*Required Arguments*

- **dataGrid1**
- **dataGrid2**
- **keyColumn1**: specifies the name of the key column in \( \text{dataGrid1} \). You can specify a column name in single quotation marks or a variable that evaluates to a column name.
- **keyColumn2**: specifies the name of the key column in \( \text{dataGrid2} \). You can specify a column name in single quotation marks or a variable that evaluates to a column name.
- **target-dataGrid**: specifies the name of the target data grid. This argument must be a variable of type DATAGRID. Any existing data in the target data grid is overwritten.
Details

The DATAGRID_RIGHTJOIN function performs a right join of two data grids for which dataGrid1.keyColumn1 equals dataGrid2.keyColumn2. It populates the target data grid with the results of the join. The target data grid contains all rows from dataGrid2 plus all matching rows from dataGrid1.

When the column names in the two data grids are identical, the columns from the right side of the join are added to the resulting data grid.

**DATAGRID_SET**

Assigns the specified value to the specified row and column.

**Category:** Set Values  
**Returned data type:** INTEGER

**Syntax**

DATAGRID_SET (dataGrid, column, rowNumber, variableOrValue)

**Required Arguments**

**dataGrid**  
specifies the name of the data grid. This argument must be a variable of type DATAGRID.

**column**  
specifies the name of the column whose value is to be set. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

**rowNumber**  
specifies the number of the row to be set. You can specify a number or a variable that evaluates to a number.

**variableOrValue**  
specifies the value to be assigned to column. You can specify a number, a character value enclosed in single quotation marks, the name of a variable, or an expression. The value that you specify must be or must evaluate to the same data type as column.

**DATAGRID_SETALL**

Assigns the specified value to all rows in the specified column.

**Category:** Set Values  
**Returned data type:** INTEGER

**Syntax**

DATAGRID_SETALL (dataGrid, column, variableOrValue)
DATAGRID_STDDEV

Returns the standard deviation of the values in the specified column.

**Required Arguments**

- **dataGrid**
  - Specifies the name of the data grid. This argument must be a variable of type DATAGRID.

- **column**
  - Specifies the name of the column whose values are to be set.

- **variableOrValue**
  - Specifies the value to be assigned to column. You can specify a number, a character value enclosed in single quotation marks, the name of a variable, or an expression. The value that you specify must be or must evaluate to the same data type as column.

---

DATAGRID_SORT

Populates the target data grid with a sorted copy of the source data grid.

- **Category**: Subset and Sort
- **Returned data type**: INTEGER
- **Note**: This function returns the number of rows in the target data grid.

**Syntax**

```
DATAGRID_SORT (source-dataGrid, sort_column, sort_order, target-dataGrid)
```

**Required Arguments**

- **source-dataGrid**
  - Specifies the name of the source data grid. This argument must be a variable of type DATAGRID.

- **sort_column**
  - Specifies the column whose values are used to sort the rows in the target data grid.

- **sort_order**
  - Specifies whether the target data grid is sorted in ascending or descending order. Specify A for ascending order or D for descending order. If you specify a variable for the sort order, and the variable value is an empty character string, the function returns a null value.

- **target-dataGrid**
  - Specifies the name of the target data grid. This argument must be a variable of type DATAGRID. Any existing data in the target data grid is overwritten.
Syntax

DATAGRID_STDDEV (dataGrid, column)

Required Arguments

dataGrid
  specifies the name of the data grid. This argument must be a variable of type DATAGRID.

column
  specifies the name of the column in the data grid. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

DATAGRID_SUBSETBYVALUE

Populates the target data grid with the rows from the source data grid for which the specified comparison evaluates to true.

  Category: Subset and Sort
  Returned data type: INTEGER
  Note: This function returns the number of rows in the target data grid.

Syntax

DATAGRID_SUBSETBYVALUE (source-dataGrid, filterColumn, operator, variableOrValue, target-dataGrid)

Required Arguments

dataGrid
  specifies the name of the data grid. This argument must be a variable of type DATAGRID.

filterColumn
  specifies the name of the column whose value is to be compared to variableOrValue. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

operator
  specifies one of the following operators shown in “Comparison Operators” on page 18. You can specify the name of a character variable that evaluates to one of these operators, or you can specify the operator enclosed in single quotation marks.

variableOrValue
  specifies the value to compare to the value of filterColumn. You can specify a number, a character string enclosed in single quotation marks, the name of a variable or of an expression. The value that you specify must be or must evaluate to the same data type as filterColumn.
target-dataGrid
specifies the name of the target data grid. This argument must be a variable of type DATAGRID. Any existing data in the target data grid is overwritten.

Details
The DATAGRID_SUBSETBYVALUE function compares the value of filterColumn in the source data grid to variableOrValue by using the comparison operator. The comparison is as follows:

```
filterColumn operator variableOrValue
```

This function populates the target data grid with all of the rows for which the comparison evaluates to true.

----------

**DATAGRID_SUM**

Returns the sum of the values in the specified column.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Statistical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returned data type:</td>
<td>DOUBLE</td>
</tr>
</tbody>
</table>

**Syntax**

```
DATAGRID_SUM (dataGrid, column)
```

**Required Arguments**

dataGrid
specifies the name of the data grid. This argument must be a variable of type DATAGRID.

column
specifies the name of the column in the data grid. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

----------

**DATAGRID_TOPN**

Populates the target data grid with the rows from the source data grid that contain the highest number values in the specified column.

<table>
<thead>
<tr>
<th>Category:</th>
<th>Subset and Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returned data type:</td>
<td>INTEGER</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>This function returns the number of rows in the target data grid. The returned data grid is sorted in descending order of the specified column.</td>
</tr>
</tbody>
</table>

**Syntax**

```
DATAGRID_TOPN (source-dataGrid, column, number, target-dataGrid)
```
**Required Arguments**

- **source-dataGrid**
  - specifies the name of the source data grid. This argument must be a variable of type DATAGRID.

- **column**
  - specifies the name of the column in the data grid. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

- **number**
  - specifies the number of rows to return.

- **target-dataGrid**
  - specifies the name of the target data grid. This argument must be a variable of type DATAGRID. Any existing data in the target data grid is overwritten.

---

**DATAGRID_TOSTRING**

Returns the JSON string for the specified data grid.

**Category:** Serialize  
**Returned data type:** CHARACTER  
**See:** "Introduction to Data Grids" on page 1

**Syntax**

`DATAGRID_TOSTRING (dataGrid)`

**Required Argument**

- **dataGrid**
  - specifies the name of the data grid. This argument must be a variable of type DATAGRID.
Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore for additional books and resources.