



# SAS<sup>®</sup> Intelligent Decisioning: Using Data Grids

2023.04\*

\* This document might apply to additional versions of the software. Open this document in [SAS Help Center](#) and click on the version in the banner to see all available versions.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2023. *SAS® Intelligent Decisioning: Using Data Grids*. Cary, NC: SAS Institute Inc.

**SAS® Intelligent Decisioning: Using Data Grids**

Copyright © 2023, SAS Institute Inc., Cary, NC, USA

All Rights Reserved. Produced in the United States of America.

**For a hard copy book:** No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**For a web download or e-book:** Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

The scanning, uploading, and distribution of this book via the Internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

**U.S. Government License Rights; Restricted Rights:** The Software and its documentation is commercial computer software developed at private expense and is provided with RESTRICTED RIGHTS to the United States Government. Use, duplication, or disclosure of the Software by the United States Government is subject to the license terms of this Agreement pursuant to, as applicable, FAR 12.212, DFAR 227.7202-1(a), DFAR 227.7202-3(a), and DFAR 227.7202-4, and, to the extent required under U.S. federal law, the minimum restricted rights as set out in FAR 52.227-19 (DEC 2007). If FAR 52.227-19 is applicable, this provision serves as notice under clause (c) thereof and no other notice is required to be affixed to the Software or documentation. The Government's rights in Software and documentation shall be only those set forth in this Agreement.

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

April 2023

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

v\_009-P1:edmdatagrids

---

# Contents

<b>Chapter 1 / Using Data Grids in SAS Intelligent Decisioning</b>	<b>1</b>
Introduction to Data Grids	1
Serialize and Import Data Grids Into a Table Column	4
Working with Data Grids	5
Using Data Grid Functions	9
Using Data Grids in a Code File	9
Working with Data Grids in SAS Studio	10
<b>Chapter 2 / Data Grid Macros</b>	<b>13</b>
Using the Data Grid Macros	13
Data Grid Macros Available with SAS Intelligent Decisioning	13
Dictionary	14
<b>Chapter 3 / Data Grid Functions</b>	<b>29</b>
Comparing Values in Data Grids	30
Data Grid Functions Available in SAS Intelligent Decisioning	31
Dictionary	37



# Using Data Grids in SAS Intelligent Decisioning

---

<b>Introduction to Data Grids</b>	<b>1</b>
What Is a Data Grid?	1
How Are Data Grids Stored?	2
When Are Data Grid Columns Created?	3
Processing Data In A Data Grid	3
<b>Serialize and Import Data Grids Into a Table Column</b>	<b>4</b>
<b>Working with Data Grids</b>	<b>5</b>
Data Grids and Publishing Destinations	5
Ways to Work with Data Grids	5
Defining Data Grid Variables	6
Scoring Rows in a Data Grid	6
Mapping Data Grid Variables in a Decision	7
Editing Data Grid Variable Metadata	7
<b>Using Data Grid Functions</b>	<b>9</b>
<b>Using Data Grids in a Code File</b>	<b>9</b>
<b>Working with Data Grids in SAS Studio</b>	<b>10</b>

---

## Introduction to Data Grids

---

### What Is a Data Grid?

A data grid is a table. A data grid variable is a variable of type DATAGRID whose value is 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

PolicyHolder	PolicyNumber	YearlyPremium
Smyth, Joe	453975R398	439.50
Smyth, Joe	987348P210	132.90
Dupree, Marcel	983092B228	334.00
Dupree, Marcel	274933P412	219.25

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

PolicyHolder	Policies	
Smyth, Joe	PolicyNumber	YearlyPremium
	453975R398	439.50
	987348P210	132.90
Dupree, Marcel	PolicyNumber	YearlyPremium
	983092B228	334.00
	274933P412	219.35

## How Are Data Grids Stored?

The data grid in the Policies column is stored as a JavaScript Object Notation (JSON) string. A data grid JSON string has the following basic format:

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

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

```
{"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:

```
[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

PolicyHolder	Policies
Smyth, Joe	[{"metadata":{"POLICYNUMBER":"string"}, {"YEARLYPREMIUM":"decimal"}}, {"data":["453975R398",439.50], ["987348P210",132.90]}]
Dupree, Marcel	[{"metadata":{"POLICYNUMBER":"string"}, {"YEARLYPREMIUM":"decimal"}}, {"data":["983092B228",324.00], ["274933P412",219.35]}]

## When Are Data Grid Columns Created?

At run time, data grid columns are automatically created for columns that already exist in the input table or that are generated by a treatment group or by a data query node. When you add a new column to the metadata for a data grid in the Edit Columns window, you can choose whether the column is automatically created at run time. For all other data grid columns, you must use a data grid function to create the data grid column at run time. For more information, see [Step 7](#) in “[Add Columns to a Data Grid](#)” and “[Data Grid Functions Available in SAS Intelligent Decisioning](#)” on page 31.

## Processing Data In A Data Grid

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 “[Data Grid Functions](#)” on page 29 to process the data. For example:

```
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.

Alternatively, you can process the data grid column in the insurance policy table by using a rule set, a model, or a decision to process each row in the data grid. To specify that an object processes each row in a data grid, you select **Score rows in this data grid** in the Properties panel for the object. In this case, you do not use the data grid functions. You define decision variables for the individual columns in the data grid, such as PolicyNumber and YearlyPremium. For more information, see “[Ways to Work with Data Grids](#)” on page 5 and “[Scoring Rows in a Data Grid](#)” on page 6.

---

# Serialize and Import Data Grids Into a Table Column

---

**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.

Data grids that are used only as temporary variables do not need to be serialized.

---

- 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 as an input or output variable 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:** If you write a serialized data grid variable to a table column, the maximum size of the JSON string is based on the engine that writes the table. For the Base SAS engine, the limit is 32,767 bytes. If you write the serialized data grid variable to a SAS Cloud Analytic Services (CAS) table, the maximum size of the JSON string is 10,485,760 bytes.

---

**Note:** The names of data grid columns are limited to 32 characters.

---

- 2 To combine data from multiple tables into one table, use the [%DCM\\_MERGESERIALIZEDGRIDS 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 table into SAS Intelligent Decisioning. See [“Importing Local Files” in SAS Data Explorer: User’s Guide](#) for more information.



---

# Working with Data Grids

---

---

## Data Grids and Publishing Destinations

---

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

You must use the %DCM\_SERIALIZEGRID macro to serialize your data grid if it is used in a rule set or in a decision that will be deployed to Apache Hadoop, Teradata, or SAS Cloud Analytic Services (CAS). For more information, see [“Serialize and Import Data Grids Into a Table Column” on page 4](#).

---

## Ways to Work with Data Grids

---

You can work with data grids in two ways:

- Use data grid functions. For more information, see [“Using Data Grid Functions” on page 9](#).
- Execute the model, rule set, or subdecision against each row in a data grid by selecting **Score rows in this data grid** when you map the object's input variables in a decision. For more information, see [“Scoring Rows in a Data Grid” on page 6](#).

In custom code files, you use data grid functions to process the data in a data grid. For more information, see [“Using Data Grids in a Code File” on page 9](#).

In data query files, you can specify whether the query returns a data grid or a single row of scalar variables. You can select what the query returns by specifying the output type on the properties panel of the query file after you add the query file to a decision. See [“Add an Existing Object” in SAS Intelligent Decisioning: User's Guide](#) and [“Create and Add a New Object” in SAS Intelligent Decisioning: User's Guide](#).

---

**Note:** You cannot use data grids in branch condition expressions. If you need to use a value calculated from a data grid in a branch condition expression, calculate the value in a rule set, assign the value to a variable, and then use the variable in a subsequent branch 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 input data grid variables you define depend on how you are working with the data grid.
  - If you are using data grid functions, define input 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 rule set or subdecision against each row in the data grid, define variables for the each column within the data grid.

---

## Scoring Rows in a Data Grid



Data grid variables are processed like any other character variable unless you do one of the following:

- use data grid functions to process the individual data values within the data grid. For more information, see [“Using Data Grid Functions” on page 9](#).
- use the **Score rows in this data grid** option to process each row in the data grid.

To process the rows within a data grid, select the **Score rows in this data grid** option on the **Input Variables** property panel when you are mapping decision variables for the node. If you select this option, you can map the node's input variables to columns in the data grid instead of to columns in the input table. SAS Intelligent Decisioning changes the default variable mappings to columns in the data grid if columns exist that have the same names as the node's input variables. You can customize the variable mappings as needed. When you select this option, the decision node processes all of the rows in the data grid before execution moves to the next node in the decision.

**IMPORTANT** When the node object is a filtering rule set, and you select **Score rows in this data grid**, rows that do not meet the criteria defined by the rules are removed from the data grid.

## Mapping Data Grid Variables in a 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 an object 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** property pane, 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](#) and [“Scoring Rows in a Data Grid” on page 6](#).


## Editing Data Grid Variable Metadata



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

**Note:** Changes to the metadata for an object's data grid affect the data grid only in the latest and subsequent versions of the object. Data grids in earlier versions of the object are not affected.

## Add Columns to a Data Grid

To add a column to a data grid:



- 1 On the **Variables** tab of a rule set, code file, or decision, click on the data grid variable that you want to edit. The Edit Variable window appears.
- 2 Click  on columns field to open the Edit Columns window.
- 3 (Optional) To add new custom columns to the data grid:
  - a Select **Add a new column**, and enter the name of the new column in the columns field.
  - b Select the data type of the column, and click **Add**.
- 4 (Optional) To add columns from a data table:
  - a Select **Add columns from a data table**, and click **Browse**. The Choose Data window appears.
  - b Select the data table, and click **OK**. SAS Intelligent Decisioning closes the Choose Data window, and adds all of the columns in the data table to the columns field in the Edit Columns window.

- c In the Edit Columns window, click **Add**. SAS Intelligent Decisioning adds all of the columns in the data table to the table of columns.
  - d (Optional) Click  for any columns that you do not want to add to your data grid.
  - e (Optional) For character string values, enter a length if you do not want to use the default length.
- 5 (Optional) To add columns from another data grid variable in the same object:
- a Select **Add columns from a data grid**, and click **Browse**. The Edit Columns window appears.
  - b Select the data grid that contains the column that you want to add.
  - c In the **Available items** list, select the columns that you want to add , and click **➤** or **➤➤**.
  - d Click **OK** to return to the Edit Columns window.
- 6 (Optional) Select the **Data does not contain column metadata** check box. The JSON character string for each row in a data grid normally contains metadata that defines the columns in the data grid and the data for each column. Depending on the data types and the number of columns in the data grid, the metadata can increase the size of the data grid considerably.
- For output-only variables, select **Data does not contain column metadata** if you do not want column metadata to be included in the output JSON string for the data grid.
- For input-only variables and input-output variables, select this check box if the input JSON string for the data grid does not contain column metadata. In order to select this option, you must click  and define the columns in the data grid. Do not select this check box if the input JSON contains column metadata. If the column definitions that you enter in the user interface do not match the input data, SAS Intelligent Decisioning does not know what the column names are. The column names are set to placeholder values such as `col_x`. If this checkbox is selected but the JSON for the input data grid contains column metadata, execution might fail or produce inaccurate results.
- 7 (Optional) Select the **Create Column** check box for the new columns if you want SAS Intelligent Decisioning to create the new columns at run time. At run time, if a column already exists in the input data grid, SAS Intelligent Decisioning does not add a new column or overwrite the existing column. For more information, see [“When Are Data Grid Columns Created?” on page 3](#).
- 8 Click **OK** to add the selected columns to your data grid variable and return to the Edit Variable window.
- 9 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 objects that iterate over a data grid, you do not need to use data grid functions. For more information, see [“Ways to Work with Data Grids” on page 5](#) and [“Scoring Rows in a Data Grid” on page 6](#).

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 [“Data Grid Functions” on page 29](#).

---

## Using Data Grids in a Code File

You can create a local data grid variable in a custom DS2 code file. However, the column metadata for the local data grid variable in the DS2 code is not passed to the corresponding decision data grid variable. In this case, you must edit the metadata for the decision variable and add the columns. For instructions, see [“Add Columns to a Data Grid” on page 7](#).

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.

```
/* 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));

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

```
end;
end;
```

See [“Using Custom Code Files”](#) in *SAS Intelligent Decisioning: User’s Guide* for information about code files.

## 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 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\_INTERFACE macro to make data grid functionality available in SAS Studio. For more information, see [“%DCM\\_DATAGRID\\_INTERFACE Macro” on page 16](#).

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:

```
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 */
    package "testCustomCode" /inline;
    /* define data grid interface methods */
    %dcm_datagrid_interface()
    method execute(varchar(32) whichColumn,
                   in_out double meanValue,
                   in_out package 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 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;
```





# Data Grid Macros

---

<i>Using the Data Grid Macros</i> .....	13
<i>Data Grid Macros Available with SAS Intelligent Decisioning</i> .....	13
<i>Dictionary</i> .....	14
%DCM_CONVERT_DATAGRID_TO_TABLE Macro .....	14
%DCM_DATAGRID_INTERFACE Macro .....	16
%DCM_DATAGRID_JSON_TO_TABLE Macro .....	17
%DCM_JSON_TO_DATAGRID Macro .....	18
%DCM_MERGESERIALIZEDGRIDS Macro .....	22
%DCM_SERIALIZEGRID Macro .....	25

---

## Using the Data Grid Macros

The SAS Intelligent Decisioning data grid macros enable you to prototype and test custom code that uses data grids in SAS Studio before you add your code to a decision.

---

**Note:** The SAS Intelligent Decisioning macros are for use in SAS Studio only.

---



---

## Data Grid Macros Available with SAS Intelligent Decisioning

[%DCM\\_CONVERT\\_DATAGRID\\_TO\\_TABLE](#)  
converts a data grid into a series of individual tables.

**%DCM\_DATAGRID\_INTERFACE**

makes the data grid functions available in SAS Studio.

**%DCM\_DATAGRID\_JSON\_TO\_TABLE**

Creates a data grid from the JavaScript Object Notation (JSON) specified in the input file, and writes the data grid to the specified output table.

**%DCM\_JSON\_TO\_DATAGRID**

Converts a standard JSON string to a data grid JSON string, or vice versa.

**%DCM\_SERIALIZEGRID**

creates a table in which one of the columns contains a JSON string that represents a data grid. Data grids must be serialized into JSON strings before you can use the data in SAS Intelligent Decisioning.

**%DCM\_MERGESERIALIZEDGRIDS**

merges tables that contain data grids that have been serialized with the %DCM\_SERIALIZEGRIDS macro with a table that contains scalar data.

---

## Dictionary

---

### %DCM\_CONVERT\_DATAGRID\_TO\_TABLE Macro

Converts each data grid to a separate table, and if the MERGE option is specified, merges the separate tables into a single output table.

Restriction: The maximum length of a data grid JSON string that can be processed by using this macro is 1,048,576 characters.

---

## Syntax

```
%DCM_CONVERT_DATAGRID_TO_TABLE (
  GRIDCOL=data-grid-column-name,
  GRIDTABLE=input-table-base-name,
  <MERGE=YES | NO, >
  <OUTLIB=libref,>
  <PROMOTE=YES | NO>
)
```

## Required Arguments

**GRIDCOL=*data-grid-column-name***

specifies the name of the data grid column whose rows you want to convert to individual tables. When MERGE=YES, the name of this column is used output table name. When MERGE=NO, the name of this column is used as the base name of the output tables. For each data grid that is converted to a separate

table, the macro appends a number to the table name. For example, if you specify Offers, the output tables are named Offers1, Offers2, Offers3, and so on.

**GRIDTABLE=***input-table-base-name*

specifies the name of the table that contains the data grid column.

## Optional Arguments

**MERGE=**YES | NO

specifies whether the tables for each row in the data grid are merged into a single output table.

**Default** NO

**OUTLIB=***libref*

specifies the library to which you want to write the output tables.

**Default** WORK

**PROMOTE=**YES | NO

promotes the merged data grid table from session scope to global scope.

**Default** YES

**Restriction** This option is valid only if the output table is written to a CAS library.

## Example

The following code defines a data set, WORK.INPUT, which contains a data grid column named ASSET. The data set contains two records, and each record contains a serialized data grid. The %DCM\_CONVERT\_DATAGRID\_TO\_TABLE macro creates a table from each serialized data grid. The macro merges the two data grid tables into one table named ASSET, which contains the columns ASSETTYPE and VALUE.

```
data work.input;
  length asset $32000.;
  asset = '{{"metadata":[{"ASSETTYPE":"string"},
    {"VALUE":"decimal"}]},
    {"data":[{"Property, Primary",212000},
    ["Property, Investment",125000]]}}';
  output;
  asset = '{{"metadata":[{"ASSETTYPE":"string"},
    {"VALUE":"decimal"}]},
    {"data":[{"Property, Primary",234500]]}}';
  output;
run;

%DCM_CONVERT_DATAGRID_TO_TABLE(
  GRIDCOL=asset,
  GRIDTABLE=work.input,
  MERGE=YES);
```

The ASSET table appears in SAS Studio as shown in the following figure:

ASSET      Table rows: 3 | Columns: 2 of 2 | Rows 1 to 3 | ↑ ↑ ⋮

🔍 Enter expression 🔍

	⚠ ASSETTYPE	⊕ VALUE
1	Property, Primary	212000
2	Property, Investment	125000
3	Property, Primary	234500

## %DCM\_DATAGRID\_INTERFACE Macro

Makes the data grid functions available in SAS Studio.

### Syntax

```
%DCM_DATAGRID_INTERFACE()
```

### Details

Use the %DCM\_DATAGRID\_INTERFACE macro to make data grid functions available in SAS Studio. You must invoke this macro before any methods 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 10](#).

```
proc ds2;

    data _null_;
        dcl package datagrid myPackage();

        %dcm_datagrid_interface();

        /* your DS2 code */

    enddata;
run;
quit;
```

## %DCM\_DATAGRID\_JSON\_TO\_TABLE Macro

Creates a data grid from the JavaScript Object Notation (JSON) string that is specified in the input file, and writes the data grid to the specified output table.

**Restriction:** The maximum length of a data grid JSON string that can be processed by using this macro is 1,048,576 characters.

### Syntax

```
%DCM_DATAGRID_JSON_TO_TABLE (
JSONFILE=data-grid-JSON-file,
OUTPUTTABLE=<libref.>table,
<PROMOTE=YES | NO>
)
```

### Required Arguments

**JSONFILE=JSON-input-file**

specifies the name of a file that contains a JSON string that defines a data grid. The JSON string must follow the same format as the serialized JSON string that is produced by the %DCM\_SERIALIZEGRID macro.

See [“How Are Data Grids Stored?” on page 2](#)

[“%DCM\\_SERIALIZEGRID Macro” on page 25](#)

**OUTPUTTABLE=<libref.>table**

specifies the libref and name of the output table. The output table can be a CAS table that is stored in the PUBLIC library or in another CAS library, or it can be a BASE table that is stored in the WORK library. The default library is WORK.

### Optional Argument

**PROMOTE=YES | NO**

promotes the merged data grid table from session scope to global scope.

**Default** YES

**Restriction** This option is valid only if the output table is written to a CAS library.

### Example

The following code defines a temporary file named DG\_JSON that contains a JSON string that defines a data grid. The data grid contains two columns: ASSETTYPE

and VALUE. The %DCM\_DATAGRID\_JSON\_TO\_TABLE macro creates a table from the serialized data grid string.

```
cas mycas;
caslib _all_ assign;
filename DG_JSON temp;
data null;
  file DG_JSON;
  put ' [{"metadata":[{"ASSETTYPE":"string"},
    {"ASSETVALUE":"double"}]},
    {"data":[{"CASH",10815},["Investment Account",2162740],
    ["Bank Account",16550],["Real Estate",745504]]}] ' ;
run;

%DCM_DATAGRID_JSON_TO_TABLE(
  JSONFILE=DG_JSON,
  OUTPUTTABLE=PUBLIC.datagrid_out,
  PROMOTE=NO);
```

The DATAGRID\_OUT table appears in SAS Studio as shown in the following figure:

DATAGRID\_OUT      Table rows: 4    Columns: 2 of 2    Rows 1 to 4    ↑ ↓ ↺ ⋮

🔍 Enter expression 🔍

	ASSETTYPE	ASSETVALUE
1	CASH	10815
2	Investment Account	2162740
3	Bank Account	16550
4	Real Estate	745504

## %DCM\_JSON\_TO\_DATAGRID Macro

Converts a standard JSON string to a data grid JSON string, or vice versa.

Restriction:      The maximum length of a data grid JSON string that can be processed by using this macro is 1,048,576 characters.

## Syntax

```
%DCM_JSON_TO_DATAGRID (
  JSONFILE=standard-JSON-file-name,
  GRIDFILE=data-grid-file-name,
  CONVERTTYPE=JSONTOGRID | GRIDTOJSON
  <RETRIEVE_NEST=YES | NO, >
  <GRIDTABLE=table-name,>
  <PROMOTE=YES | NO>
```

)

## Required Arguments

### CONVERTTYPE=JSONTOGRID | GRIDTOJSON

specifies the conversion type. Specify `GRIDTOJSON` to convert the data grid JSON in the file that is specified by the `GRIDFILE=` option to standard JSON. Specify `JSONTOGRID` to convert standard JSON in the file that is specified by the `JSONFILE=` option to data grid JSON.

### JSONFILE=*standard-JSON-file-name*

specifies the name of the file for standard JSON data. If you are converting standard JSON to data grid JSON, this file is the input file. If you are converting data grid JSON to standard JSON, this file is the output file.

### GRIDFILE=*data-grid-JSON-file-name*

specifies the name of a file for the data grid JSON string. If you are converting data grid JSON to standard JSON, this file is the input file. If you are converting standard JSON to data grid JSON, the macro determines the output file as follows:

- If you specify `RETRIEVE_NEST=YES` and the input file does not contain nested JSON, or if you specify `RETRIEVE_NEST=NO`, then the macro writes the output to the file that is specified by the `GRIDFILE=` option.
- If you specify `RETRIEVE_NEST=YES` and the input file contains nested JSON, then the macro writes the output to the table that is specified by the `GRIDTABLE=` option. The `GRIDFILE=` option is ignored.

## Optional Arguments

### RETRIEVE\_NEST=YES | NO

specifies whether to convert nested JSON when you convert standard JSON to data grid JSON. If you specify `NO`, the macro converts only the top-level JSON structure in the input file and writes the output to the file that is specified by the `GRIDFILE=` option. If you specify `YES`, then you must specify the `GRIDTABLE=` option, and if the input JSON contains nested structures, the macro writes the data grid JSON to the table that is specified by the `GRIDTABLE=` option.

**Default** NO

**Restriction** This option is valid only if you also specify the `CONVERTTYPE=JSONTOGRID` option.

### GRIDTABLE=<libref.>*table-name*

specifies the table that contains the data grid JSON that is produced by the macro if you specify `CONVERTTYPE=JSONTOGRID` and `RETRIEVE_NEST=YES`. The output table can be a CAS table that is stored in the `PUBLIC` library or in another CAS library, or it can be a `BASE` table that is stored in the `WORK` library. The default library is `WORK`. If the data grid is very large, specify a CAS table name.

**Note** This option is ignored if you specify the `RETRIEVE_NEST=NO` option or if the input file does not contain nested JSON.

### PROMOTE=YES | NO

promotes the merged data grid table from session scope to global scope.

**Default** YES

**Restriction** This option is valid only if the output table is written to a CAS library.

## Examples

### Example 1: Converting Data Grid JSON to Standard JSON

The following code creates a temporary file that contains a data grid JSON string. The macro converts this string to standard JSON and writes it to the file `/My Folder/stdJSONfile.txt`.

```
filename gridFile temp;
filename jsonFile filesrv folderpath='/Users/userID/My Folder'
filename='stdJSONfile.txt';

data _null_;
  file gridFile;
  put ' [{ "metadata": [{ "ASSETTYPE": "string",
                        { "ASSETVALUE": "double" } } ],
      { "data": [ ["Bank Account", 47465],
                  ["Retirement Fund", 3.29875E+06],
                  ["CASH", 163],
                  ["Investment Account", 2.29051E+06],
                  ["Auto", 2669]] } ] }';

run;

%dcm_json_to_datagrid(jsonFile=jsonfile,
  gridFile=gridfile,
  convertType=GRIDtoJSON);
```

The output file `STDJSONFILE.TXT` contains the following JSON:

```
[{"ASSETTYPE": "Bank Account", "ASSETVALUE": 47465},
 {"ASSETTYPE": "Retirement Fund", "ASSETVALUE": 3298750},
 {"ASSETTYPE": "CASH", "ASSETVALUE": 163},
 {"ASSETTYPE": "Investment Account", "ASSETVALUE": 2290510},
 {"ASSETTYPE": "Auto", "ASSETVALUE": 2669}]
```

### Example 2: Converting Standard JSON to Data Grid JSON

The following program creates a temporary file that contains standard JSON that includes nested JSON structures. It calls the `%DCM_JSON_TO_GRID` macro to convert the standard JSON to data grid JSON. It writes the data grid JSON, including the nested data, to the CAS table named `Public.STDGRID`.

```
cas mycas;
caslib _all_ assign;
```



```

filename stdJSON temp;
filename gridFile temp;

data _null_;
  file jsonFile;
  put '[{
    "Debts": [{
      "REMAININGDEBT": 22144,
      "REMAININGPAYMENTS": 24,
      "DEBTTYPE": "Boat Loan",
      "Bankers": [{ "BANKOFFICER": "Matt" }]
    }],
    "ASSETTYPE": "Bank Account",
    "ASSETVALUE": 47465
  },
  {
    "Debts": [{
      "REMAININGDEBT": 715551,
      "REMAININGPAYMENTS": 84,
      "DEBTTYPE": "Mortgage",
      "Bankers": [{ "BANKOFFICER": "Carl" }]
    }],
    "ASSETTYPE": "Retirement Fund",
    "ASSETVALUE": 3298750
  },
  {
    "Debts": [{
      "REMAININGDEBT": 15577,
      "REMAININGPAYMENTS": 67,
      "DEBTTYPE": "Car Loan",
      "Bankers": [{ "BANKOFFICER": "Ernest" }]
    }],
    "ASSETTYPE": "CASH",
    "ASSETVALUE": 163
  },
  {
    "Debts": [{
      "REMAININGDEBT": 230544,
      "REMAININGPAYMENTS": 45,
      "DEBTTYPE": "Student Loan",
      "Bankers": [{ "BANKOFFICER": "Randy" }]
    }],
    "ASSETTYPE": "Investment Account",
    "ASSETVALUE": 2290510
  }
]';

run;

%DCM_JSON_TO_DATAGRID(
  CONVERTTYPE=JSONTOGRID,
  JSONFILE=stdJSON,
  GRIDFILE=gridFile,
  RETRIEVE_NEST=YES,
  GRIDTABLE=Public.stdToGrid,
  PROMOTE=NO);

```

The STDTOGRID output table appears in SAS Studio as shown in the following figure:

	ROOT	DEBTS	DEBTS_BANKERS
1	[{"metadata":{"ASSETTYPE":"string"},"A...	[{"metadata":{"ORDINAL_ROOT":"deci...	[{"metadata":{"ORDINAL_DEBTS":"decimal"},"ORDINAL_BANKERS":"decim...

The first column in the output table is named ROOT. Subsequent column names indicate the names of the JSON data structure and the nesting level. In this example, the ROOT column contains the data grid JSON for the name-value pairs at the first-level that do not contain any nested name-value pairs: ASSETTYPE and ASSETVALUE.

```
[{"metadata":{"ASSETTYPE":"string"},"ASSETVALUE":"decimal"}],
{"data":["Bank Account",47465],["Retirement Fund",3298750],
["CASH",163],["Investment Account",2290510]]}]}
```

The DEBTS column contains the JSON for the DEBTS data structure, which appears at the same level as ASSETTYPE and ASSETVALUE but contains four nested elements. The parent of the DEBTS structure is ROOT.

```
[{"metadata":{"ORDINAL_ROOT":"decimal"},"ORDINAL_DEBTS":"decimal"},
{"REMAININGDEBT":"decimal"},"REMAININGPAYMENTS":"decimal"},
{"DEBTYPE":"string"}]],{"data":[[1,1,22144,24,"Boat Loan"],
[2,2,71551,84,"Mortgage"],[3,3,15577,67,"Car Loan"],
[4,4,230544,45,"Student Loan"]]}]}
```

The DEBTS\_BANKERS column contains the data grid JSON for the BANKERS data that is nested within the DEBTS data structure.

```
[{"metadata":{"ORDINAL_DEBTS":"decimal"},"ORDINAL_BANKERS":"decimal"},
{"BANKOFFICER":"string"}]],{"data":[[1,1,"Matt"],[2,2,"Carl"],
[3,3,"Ernest"],[4,4,"Randy"]]}]}
```

## %DCM\_MERGESERIALIZEDGRIDS Macro

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.

**Note:** This macro performs an inner join. If any key column value does not appear in one of the data grid tables specified by the GRIDTABLES= option or in the scalar table specified by the MERGETABLE= option, then the table rows for that value of the key column are not included in the output results table.

## Syntax

%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>...,
<PROMOTE=YES | NO>
)

```

## 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=<*libref.*>*table***

specifies the name of the table that contains the results of the merge. The default library is WORK.

### **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=*data\_grid\_key1* <*data\_grid\_key2*>...**

specifies the names of the key columns in the tables specified by the GRIDTABLES= argument.

**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=*data\_grid\_column1* <*data\_grid\_column2*>...**

specifies the names of the data grid columns in the tables specified by the GRIDTABLES= argument.

**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.

### **PROMOTE=YES | NO**

promotes the merged data grid table from session scope to global scope.

**Default** YES

**Restriction** This option is valid only if the output table is written to a CAS library.

## 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 `debtsGrid` and `assetsGrid`. 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`.

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

**Table 2.1** *mylib.loanrequests Merge Table*

RequestID	RequestedAmt	Customer
ME4922Mac01	20000	MacKelly, Sara
NC2W497Smy03	80495	Smyth, Joe

**Table 2.2** *debtsGrid Data Grid Table*

CustName	Debts
MacKelly, Sara	[{"metadata":{"DEBTTYPE":"string"},"BALANCE":"decimal"}], {"data": [{"Mortgage",80053.16}, {"CreditCard",2143.68}]]

**Table 2.3** *assetsGrid Data Grid Table*

CustName	Assets
MacKelly, Sara	[{"metadata":{"ASSETTYPE":"string"},"VALUE":"decimal"}], {"data": [{"Property, Primary",212000}, {"Property, Investment",125000}]]
Smyth, Joe	[{"metadata":{"ASSETTYPE":"string"},"VALUE":"decimal"}], {"data": [{"Property, Primary",234500}]]

Table 2.4 mylib.loanRequestData Results Table

RequestID	RequestAmount	Customer	Debts	Assets
ME4922Mac01	20000	MacKelly, Sara	[{"metadata": [{"DEBTTYPE": "string"}, {"BALANCE": "decimal"}]}, {"data": [{"Mortgage", 80053.16}, {"CreditCard", 2143.68}]]}]	[{"metadata": [{"ASSETTYPE": "string"}, {"VALUE": "decimal"}]}, {"data": [{"Property, Primary", 212000}, {"Property, Investment", 125000}]]}]

## %DCM\_SERIALIZEGRID Macro

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

```
%DCM_SERIALIZEGRID (
  GRIDCOLNAME=data_grid_column_name,
  GRIDSOURCETABLE=table_name,
  OUTPUTTABLE=<libref.>table,
  <CLASSVARS=class_variable1< class_variable2...>,>
  <GRIDCOLS=column1< column2...>,>
  <GRIDCOLLEN=length,>
  <PROMOTE=YES | NO>
)
```

## Required Arguments

**GRIDCOLNAME=***data\_grid\_column\_name*

specifies the name for the serialized data grid column in the results table.

**GRIDSOURCETABLE=***table\_name*

specifies the table that contains the columns that are to be serialized into a data grid.

**OUTPUTTABLE=***<libref.>table*

specifies the name of the output table that contains the data grid column. The default library is WORK.

## 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

### **PROMOTE=**YES | NO

promotes the merged data grid table from session scope to global scope.

**Default** YES

**Restriction** This option is valid only if the output table is written to a CAS library.

## 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 2.5** mylib.assets Source Table

CustName	AssetType	Value
MackKelly, Sara	Property, Primary	212000
MackKelly, Sara	Property, Investment	125000
Smyth, Joe	Property, Primary	234500

Table 2.6 assestGrid Results Table

CustName	Assets
MackKelly, Sara	[{"metadata":{"ASSETTYPE":"string"},"VALUE":"decimal"}], {"data": [["Property, Primary",212000],["Property, Investment",125000]]}]
Smyth, Joe	[{"metadata":{"ASSETTYPE":"string"},"VALUE":"decimal"}], {"data": [["Property, Primary",234500]]}]





# Data Grid Functions

---

<i>Comparing Values in Data Grids</i> .....	<b>30</b>
<i>Data Grid Functions Available in SAS Intelligent Decisioning</i> .....	<b>31</b>
<i>Dictionary</i> .....	<b>37</b>
DATAGRID_ADDCHARACTERCOLUMN Function .....	37
DATAGRID_ADDCOLUMN Function .....	37
DATAGRID_ADDNUMERICCOLUMN Function .....	38
DATAGRID_ADDROW Function .....	39
DATAGRID_APPEND Function .....	39
DATAGRID_BOTTOMN Function .....	40
DATAGRID_CLEAR Function .....	41
DATAGRID_CLEARCOLUMN Function .....	41
DATAGRID_CLEARDATA Function .....	42
DATAGRID_COLUMNCOUNT Function .....	42
DATAGRID_COLUMNINDEX Function .....	43
DATAGRID_COLUMNISNUMERIC Function .....	43
DATAGRID_COLUMNNAME Function .....	44
DATAGRID_COLUMNTYPE Function .....	44
DATAGRID_COLUMNVALESTOLIST Function .....	45
DATAGRID_CONFORM Function .....	46
DATAGRID_COPY Function .....	46
DATAGRID_COPYCOLUMN Function .....	47
DATAGRID_CORR Function .....	47
DATAGRID_COUNT Function .....	48
DATAGRID_CREATE Function .....	49
DATAGRID_DELETECOLUMN Function .....	50
DATAGRID_DELETEROW Function .....	50
DATAGRID_DISTINCTROWCOUNT Function .....	51
DATAGRID_EQUALS Function .....	51
DATAGRID_FILTEREDGET Function .....	52
DATAGRID_FILTEREDGETINDEX Function .....	53
DATAGRID_FILTEREDSET Function .....	54
DATAGRID_FILTEREDSETALL Function .....	56
DATAGRID_FREQ Function .....	57
DATAGRID_GET Function .....	58
DATAGRID_GETBOOL Function .....	58
DATAGRID_GETDOUBLE Function .....	59

DATAGRID_GETFIRSTINDEXBYVALUE Function .....	59
DATAGRID_GETFIRSTMISSINGINDEX Function .....	60
DATAGRID_GETINT Function .....	61
DATAGRID_GETLASTINDEXBYVALUE Function .....	61
DATAGRID_GETLASTMISSINGINDEX Function .....	62
DATAGRID_GETLONG Function .....	62
DATAGRID_GETSTRING Function .....	63
DATAGRID_GRIDMATCHCOUNT Function .....	64
DATAGRID_INITIALIZE Function .....	64
DATAGRID_INNERJOIN Function .....	65
DATAGRID_LEFTJOIN Function .....	66
DATAGRID_MATCHCOUNT Function .....	67
DATAGRID_MAX Function .....	68
DATAGRID_MAXBOOL Function .....	68
DATAGRID_MAXINT Function .....	69
DATAGRID_MAXLONG Function .....	69
DATAGRID_MAXSTRING Function .....	70
DATAGRID_MEAN Function .....	70
DATAGRID_MEDIAN Function .....	71
DATAGRID_MIN Function .....	71
DATAGRID_MINBOOL Function .....	72
DATAGRID_MININT Function .....	72
DATAGRID_MINLONG Function .....	73
DATAGRID_MINSTRING Function .....	73
DATAGRID_MULTISORT Function .....	74
DATAGRID_NMISS Function .....	75
DATAGRID_NVALID Function .....	75
DATAGRID_RENAMECOLUMN Function .....	76
DATAGRID_RIGHTJOIN Function .....	76
DATAGRID_SET Function .....	77
DATAGRID_SETALL Function .....	78
DATAGRID_SETBOOL Function .....	79
DATAGRID_SETDOUBLE Function .....	79
DATAGRID_SETINT Function .....	80
DATAGRID_SETLONG Function .....	81
DATAGRID_SETSTRING Function .....	82
DATAGRID_SORT Function .....	82
DATAGRID_STDDEV Function .....	83
DATAGRID_SUBSETBYVALUE Function .....	84
DATAGRID_SUM Function .....	86
DATAGRID_SUMLONG Function .....	86
DATAGRID_TOPN Function .....	87
DATAGRID_TOSTRING Function .....	87

---

## Comparing Values in Data Grids

You can use any of the following operators with functions that compare values such as the DATAGRID\_FILTEREDGET, DATAGRID\_FILTEREDGETINDEX, and DATAGRID\_MATCHCOUNT functions.

Operator	Description
EQ, =, ==	Equals
NE, !=, ^=, <>	Not equal to
GT, >	Greater than
LT, <	Less than
LE, <=	Less than or equal to
GE, >=	Greater than or equal to

**IMPORTANT** Unless noted otherwise in the description for a specific function, the equals operator (=) trims trailing spaces before comparing values. Therefore, if you compare “hello” to “hello ”, the comparison evaluates to True.

## 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.

**Table 3.1** Data Grid Function Categories

Category	Description
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

Category	Description
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.

Category	Language Elements	Description
Create, Update, or Delete	DATAGRID_ADDCHARACTERCOLUMN Function (p. 37)	Adds a character column to the specified data grid.
	DATAGRID_ADDCOLUMN Function (p. 37)	Adds a column of the specified type to the specified data grid.
	DATAGRID_ADDNUMERICCOLUMN Function (p. 38)	Adds a numeric column to the specified data grid.
	DATAGRID_ADDROW Function (p. 39)	Appends new rows to the specified data grid.
	DATAGRID_CLEAR Function (p. 41)	Deletes all rows and all column metadata from the specified data grid.
	DATAGRID_CLEARCOLUMN Function (p. 41)	Deletes the data in the specified columns.
	DATAGRID_CLEARDATA Function (p. 42)	Deletes all rows from the specified data grid but does not remove the column metadata.
	DATAGRID_CONFORM Function (p. 46)	Adds the columns that are exclusive to dataGrid1 to dataGrid2, and adds the columns that are exclusive to dataGrid2 to dataGrid1.
	DATAGRID_COPY Function (p. 46)	Copies the source data grid into the target data grid.
	DATAGRID_COPYCOLUMN Function (p. 47)	Copies the specified column in the source data grid into the target data grid.
	DATAGRID_CREATE Function (p. 49)	Creates the data grid columns that are defined in the specified JSON string, and assigns to those columns any values that are specified in the JSON string.
	DATAGRID_DELETECOLUMN Function (p. 50)	Deletes the specified column from the specified data grid.
	DATAGRID_DELETEROW Function (p. 50)	Deletes one or more rows from the specified data grid.

Category	Language Elements	Description
Information	DATAGRID_INITIALIZE Function (p. 64)	Replaces all of the values in the specified data grid with the specified default values.
	DATAGRID_COLUMNCOUNT Function (p. 42)	Returns the number of columns in the specified data grid.
	DATAGRID_COLUMNINDEX Function (p. 43)	Returns the index of the specified column in the specified data grid.
	DATAGRID_COLUMNISNUMERIC Function (p. 43)	Determines whether the data type of the specified data grid column is numeric.
	DATAGRID_COLUMNNAME Function (p. 44)	Returns the name of the column that is in the specified ordinal position in the specified data grid.
	DATAGRID_COLUMNTYPE Function (p. 44)	Returns the data type of the specified column in the specified data grid.
	DATAGRID_COUNT Function (p. 48)	Returns the number of rows in the specified data grid.
	DATAGRID_DISTINCTROWCOUNT Function (p. 51)	Returns the number of unique rows in the specified data grid.
	DATAGRID_EQUALS Function (p. 51)	Compares the column count, row count, column names, data types, and values in specified data grids and determines whether they are identical.
	DATAGRID_FILTEREDGETINDEX Function (p. 53)	Returns the index of the first row in the specified column for which the specified comparison evaluates to true.
	DATAGRID_GETFIRSTINDEXBYVALUE Function (p. 59)	Returns the index of the first row that contains the specified value in the specified column.
	DATAGRID_GETFIRSTMISSINGINDEX Function (p. 60)	Returns the index of the first row that contains a missing value in the specified column.
	DATAGRID_GETLASTINDEXBYVALUE Function (p. 61)	Returns the index of the last row that contains the specified value in the specified column.

DATAGRID\_GETLASTMISSINGINDEX Function (p. 62)

DATAGRID\_GRIDMATCHCOUNT Function (p. 64)

DATAGRID\_MATCHCOUNT Function (p. 67)

DATAGRID\_NVALID Function (p. 75)

Category	Language Elements	Description
Join or Append	DATAGRID_ADDROW Function (p. 39)	Appends new rows to the specified data grid.
	DATAGRID_APPEND Function (p. 39)	Appends dataGrid2 to dataGrid1.
	DATAGRID_INNERJOIN Function (p. 65)	Performs an inner join of two data grids and populates the target data grid with the results of the join.
	DATAGRID_LEFTJOIN Function (p. 66)	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.
	DATAGRID_RIGHTJOIN Function (p. 76)	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.
Rename	DATAGRID_RENAMECOLUMN Function (p. 76)	Renames the specified column in the specified data grid.
Retrieve Values	DATAGRID_COLUMNVALUESTOLIST Function (p. 45)	Creates a comma-separated list of the values in the specified data grid column.
	DATAGRID_FILTEREDGET Function (p. 52)	Returns the value in the first row in the specified column for which the specified comparison evaluates to true.
	DATAGRID_GET Function (p. 58)	Returns the value of the cell in the specified row and column.
	DATAGRID_GETBOOL Function (p. 58)	Returns the value of the cell in the specified row and column as an integer.
	DATAGRID_GETDOUBLE Function (p. 59)	Returns the value of the cell in the specified row and column as a decimal value.
	DATAGRID_GETINT Function (p. 61)	Returns the value of the cell in the specified row and column as an integer.
	DATAGRID_GETLONG Function (p. 62)	Returns the value of the cell in the specified row and column as a long integer.
	DATAGRID_GETSTRING Function (p. 63)	Returns the value of the cell in the specified row and column as a character string.
Serialize	DATAGRID_TOSTRING Function (p. 87)	Returns the JSON string for the specified data grid.
Set Values	DATAGRID_FILTEREDSET Function (p. 54)	Sets the value in the specified row and column to the specified value if the specified comparison evaluates to true.
	DATAGRID_FILTEREDSETALL Function (p. 56)	Sets the cell in the specified column to the specified value for each row for which the specified comparison evaluates to true.

Category	Language Elements	Description
	DATAGRID_INITIALIZE Function (p. 64)	Replaces all of the values in the specified data grid with the specified default values.
	DATAGRID_SET Function (p. 77)	Assigns the specified value to the specified row and column.
	DATAGRID_SETALL Function (p. 78)	Assigns the specified value to all rows in the specified column.
	DATAGRID_SETBOOL Function (p. 79)	Assigns the specified Boolean value to the specified row and column.
	DATAGRID_SETDOUBLE Function (p. 79)	Assigns the specified decimal value to the specified row and column.
	DATAGRID_SETINT Function (p. 80)	Assigns the specified integer value to the specified row and column.
	DATAGRID_SETLONG Function (p. 81)	Assigns the specified long integer value to the specified row and column.
	DATAGRID_SETSTRING Function (p. 82)	Assigns the specified character string to the specified row and column.
Statistical	DATAGRID_CORR Function (p. 47)	Returns the Pearson product-moment correlation coefficient for the specified columns in the specified data grid.
	DATAGRID_FREQ Function (p. 57)	Returns the number of distinct values for the specified column.
	DATAGRID_MAX Function (p. 68)	Returns the maximum value for the specified column.
	DATAGRID_MAXBOOL Function (p. 68)	Returns the maximum value for the specified column.
	DATAGRID_MAXINT Function (p. 69)	Returns the maximum value for the specified column.
	DATAGRID_MAXLONG Function (p. 69)	Returns the maximum value for the specified column.
	DATAGRID_MAXSTRING Function (p. 70)	Returns the maximum value for the specified column.
	DATAGRID_MEAN Function (p. 70)	Returns the mean value for the specified column.
	DATAGRID_MEDIAN Function (p. 71)	Returns the median value for the specified column.
	DATAGRID_MIN Function (p. 71)	Returns the minimum value that appears in the specified column.

Category	Language Elements	Description
	DATAGRID_MINBOOL Function (p. 72)	Returns the minimum value that appears in the specified column.
	DATAGRID_MININT Function (p. 72)	Returns the minimum value that appears in the specified column.
	DATAGRID_MINLONG Function (p. 73)	Returns the minimum value that appears in the specified column.
	DATAGRID_MINSTRING Function (p. 73)	Returns the minimum value that appears in the specified column.
	DATAGRID_NMISS Function (p. 75)	Returns the number of missing values for the specified column of the specified data grid.
	DATAGRID_STDDEV Function (p. 83)	Returns the standard deviation of the values in the specified column.
	DATAGRID_SUM Function (p. 86)	Returns the sum of the values in the specified column.
	DATAGRID_SUMLONG Function (p. 86)	Returns the sum of the values in the specified column.
Subset and Sort	DATAGRID_BOTTOMN Function (p. 40)	Populates the target data grid with the rows from the source data grid that contain the lowest number values in the specified column.
	DATAGRID_MULTISORT Function (p. 74)	Sorts a data grid based on the values of one or more columns, and then populates the target data grid with the sorted data.
	DATAGRID_SORT Function (p. 82)	Sorts a data grid based on the values in a single column, and then populates the target data grid with the sorted data.
	DATAGRID_SUBSETBYVAL UE Function (p. 84)	Populates the target data grid with the rows from the source data grid for which the specified comparison evaluates to true.
	DATAGRID_TOPN Function (p. 87)	Populates the target data grid with the rows from the source data grid that contain the highest number values in the specified column.



---

# Dictionary

---

## DATAGRID\_ADDCHARACTERCOLUMN Function

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\_ADDCOLUMN Function

Adds a column of the specified type 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\_ADDCOLUMN** (*dataGrid*, *column*, *type*)

### 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.

***type***

specifies the data type of the new column. Specify one of the following data types: `boolean`, `bool`, `double`, `decimal`, `integer`, `int`, `long`, or `string`. If you specify an invalid data type, the new column is not created.

---

## DATAGRID\_ADDNUMERICCOLUMN Function

Adds a numeric 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\_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 Function

Appends new rows to the specified data grid.

Categories: Create, Update, or Delete  
Join or Append

Returned data type: INTEGER

Note: This function returns the number of rows that are in the data grid after the new row has been added.

## Syntax

**DATAGRID\_ADDROW** (*dataGrid*<, *numberOfRows*=*n*>)

## Required Argument

### ***dataGrid***

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

## Optional Argument

### ***numberOfRows*=*n***

specifies the number of rows that you want to add to the data grid. If you do not specify this argument, the DATAGRID\_ADDROW function adds only one row to the data grid.

**Default** 1

# DATAGRID\_APPEND Function

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 Function

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 Function

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\_CLEARCOLUMN Function

Deletes the data in the specified columns.

Category: Create, Update, or Delete

Returned data type: INTEGER

Note: This function always returns a zero.

---

### Syntax

**DATAGRID\_CLEARCOLUMN** (*dataGrid*, *column<*, *start\_row\_number*, *end\_row\_number>*)

### 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.

## Optional Arguments

**start\_row\_number**

specifies the index of the first row from which you want to delete the data in the specified column.

**Interaction** If you specify *start\_row\_number*, then you must also specify *end\_row\_number*. To delete the data in a single row, specify the same value for both arguments.

**end\_row\_number**

specifies the index of the last row from which you want to delete the data in the specified column.

---

## DATAGRID\_CLEARDATA Function

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 Function

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\_COLUMNINDEX Function

Returns the index of the specified column in the specified data grid.

Category: Information

Returned data type: INTEGER

Note: If the column does not exist in the specified data grid, an error is written to the log.

---

## Syntax

**DATAGRID\_COLUMNINDEX** (*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\_COLUMNISNUMERIC Function

Determines whether the data type of the specified data grid column is numeric.

Category: Information

Returned data type: INTEGER

Note: This function returns a 1 if the column data type is numeric and a 0 if it is not numeric.

---

## Syntax

**DATAGRID\_COLUMNISNUMERIC** (*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\_COLUMNNAME Function

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\_COLUMNTYPE Function

Returns the data type of the specified column in the specified data grid.

Category: Information

Returned data type: CHARACTER

Note: This function returns either `string` or `decimal`, depending on the column data type.



---

## Syntax

**DATAGRID\_COLUMNTYPE** (*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\_COLUMNVALUESTOLIST Function

Creates a comma-separated list of the values in the specified data grid column.

Category: Retrieve Values

Returned data type: INTEGER

---

## Syntax

**DATAGRID\_COLUMNVALUESTOLIST** (*dataGrid*, *column*, *valueList*)

## 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.

***valueList***

specifies a name for the list of values.

---

## Details

Trailing blanks are removed from the values in the data grid before they are added to the list. Missing values in the data grid are indicated in the value list by two comma separators without any content between them. All values are added to the list as character values. The list is enclosed in square brackets. For example:

```
[50,,150]
[, ,Australia,New Zealand]
```

---

## DATAGRID\_CONFORM Function

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

**DATAGRID\_CONFORM** (*dataGrid1*, *dataGrid2*)

### Required Argument

***dataGrid1***

***dataGrid2***

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

---

## DATAGRID\_COPY Function

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

**DATAGRID\_COPY** (*source\_dataGrid*, *target\_dataGrid*)

## Required Arguments

***source\_dataGrid***

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

***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\_COPYCOLUMN Function

Copies the specified column in the source data grid into the target data grid.

Category: Create, Update, or Delete

Returned data type: INTEGER

Note: This function returns 1 if the operation is successful and 0 if it is not successful..

---

## Syntax

**DATAGRID\_COPYCOLUMN** (*source\_dataGrid*, *target\_dataGrid*, *source-column*, *target-column*)

## Required Arguments

***source\_dataGrid***

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

***target\_dataGrid***

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

***source-column***

specifies the name of the column in the source data grid that you want to copy.

***target-column***

specifies a name for the new column in the target data grid.

---

## DATAGRID\_CORR Function

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

Category: Statistical

Returned data type: DOUBLE

---

## Syntax

**DATAGRID\_CORR** (*dataGrid*, *column1*, *column2*)

## 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:

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

---

## DATAGRID\_COUNT Function

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 Function

Creates the data grid columns that are defined in the specified JSON string, and assigns to those columns any values that are specified in the JSON string.

Category:	Create, Update, or Delete
Requirement:	The data grid variable must already be defined as a variable of type DATAGRID.
Returned data type:	INTEGER
Note:	This function returns the number of rows that are in the data grid.

# Syntax

**DATAGRID\_CREATE** (*dataGrid*, *JSON\_string*)

## Required Arguments

**dataGrid**

specifies the name of a variable of type DATAGRID.

**Important** This variable must already be defined as a variable of type DATAGRID. If the variable is not defined as a DATAGRID variable, this function creates a variable of type CHARACTER that contains the JSON string. If the variable is defined as a DATAGRID variable, this function creates the columns in the data grid and assigns to those columns any values that are specified in the JSON string.

**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. The JSON string must have the same format as the string that is produced by the [%DCM\\_SERIALIZEDGRID](#) macro:

```
[{"metadata":[{"column1":"data_type"}< ,{"column2":"data_type"}...]}, >
{"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 26](#)

---

## DATAGRID\_DELETECOLUMN Function

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 Function

Deletes one or more rows 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*, *start\_row\_number*<, *end\_row\_number*>)

## Required Arguments

***dataGrid***

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

***start\_row\_number***

specifies the index of the first row that you want to delete.

## Optional Argument

***end\_row\_number***

specifies the index of the last row that you want to delete.

---

## DATAGRID\_DISTINCTROWCOUNT Function

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\_EQUALS Function

Compares the column count, row count, column names, data types, and values in specified data grids and determines whether they are identical.

Category: Information

Returned data type: INTEGER

Note: This function returns 1 if the two data grids are identical and 0 if they are not identical.

---

## Syntax

**DATAGRID\_EQUALS** (*dataGrid1*, *dataGrid2*)

## Required Argument

***dataGrid1***

***dataGrid2***

specifies the names of the two data grids that you want to compare.

---

## DATAGRID\_FILTEREDGET Function

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

Category: Retrieve Values

Returned data type: STRING

Note: If no rows match the specified comparison, an error is written to the log.

---

## Syntax

**DATAGRID\_FILTEREDGET** (*dataGrid*, *columnToSearch*, *filterColumn*, *operator*, *variableOrValue*)

## Required Arguments

***dataGrid***

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

***columnToSearch***

specifies the name of the column whose value you want to retrieve. 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 “[Comparing Values in Data Grids](#)” on page 30. 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, or 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 Low, and if so, returns the value of the approvalStatus column:

```
DATAGRID_FILTEREDGET(DebtEval, 'approvalStatus', 'riskGroup', 'EQ', 'Low')
```

---

# DATAGRID\_FILTEREDGETINDEX Function

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

Category: Information

Returned data type: STRING, DOUBLE

Note: If no rows match the specified comparison, this function returns -1.

---

## Syntax

**DATAGRID\_FILTEREDGETINDEX** (*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 [“Comparing Values in Data Grids” on page 30](#). 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, or the name of a variable or of an expression. Trailing blanks are included in the comparison. The value that you specify must be or must evaluate to the same data type as *filterColumn*.

---

## Details

The DATAGRID\_FILTEREDGETINDEX function compares the value in *filterColumn* to the specified *variableOrValue* by using the comparison operator. The comparison is as follows:

```
filterColumn operator variableOrValue
```

This function returns the row number of the first row for which the comparison evaluates to true.

---

## Example

The following example searches for the first row in the DebtEval data grid where the value of the riskGroup column is equal to Low. It returns the row number of the row, or if it does not find the value Low, it returns -1.

```
DATAGRID_FILTEREDGET(DebtEval, 'riskGroup', 'EQ', 'Low')
```

---

## DATAGRID\_FILTEREDSET Function

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

## Syntax

**DATAGRID\_FILTEREDSET** (*dataGrid*, *columnToAssign*, *rowNumber*, *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.

### **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 *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 [“Comparing Values in Data Grids” on page 30](#). 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, or 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 cell at *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:

```
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:

```
DATAGRID_FILTEREDSET(DebtEval, 'riskGroup', customer, 'debts', 'GT', assets, 'High')
```

## DATAGRID\_FILTEREDSETALL Function

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

**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 [“Comparing Values in Data Grids” on page 30](#). 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, or 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 Function

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\_GET Function

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

Category: Retrieve Values

Returned data type: STRING

---

### 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\_GETBOOL Function

Returns the value of the cell in the specified row and column as an integer.

Category: Retrieve Values

Returned data type: INTEGER

Note: The value True is returned as the integer 1, and False is returned as the integer 0.

---

### Syntax

**DATAGRID\_GETBOOL** (*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\_GETDOUBLE Function

Returns the value of the cell in the specified row and column as a decimal value.

Category: Retrieve Values

Returned data type: DOUBLE

---

## Syntax

**DATAGRID\_GETDOUBLE** (*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\_GETFIRSTINDEXBYVALUE Function

Returns the index of the first row that contains the specified value in the specified column.

Category: Information

Returned data type: INTEGER

Note: This function returns a zero if the specified value does not exist in the specified column.

---

## Syntax

**DATAGRID\_GETFIRSTINDEXBYVALUE** (*dataGrid*, *column*, *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 in the data grid. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

***variableOrValue***

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

---

## DATAGRID\_GETFIRSTMISSINGINDEX Function

Returns the index of the first row that contains a missing value in the specified column.

Category: Information

Returned data type: INTEGER

Note: This function returns a zero if the specified column does not contain any missing values.

---

## Syntax

**DATAGRID\_GETFIRSTMISSINGINDEX** (*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\_GETINT Function

Returns the value of the cell in the specified row and column as an integer.

Category: Retrieve Values

Returned data type: INTEGER

---

### Syntax

**DATAGRID\_GETINT** (*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\_GETLASTINDEXBYVALUE Function

Returns the index of the last row that contains the specified value in the specified column.

Category: Information

Returned data type: INTEGER

Note: This function returns a zero if the specified value does not exist in the specified column.

---

### Syntax

**DATAGRID\_GETLASTINDEXBYVALUE** (*dataGrid*, *column*, *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 in the data grid. You can specify a column name in single quotation marks or a variable that evaluates to a column name.

### ***variableOrValue***

specifies the value to compare to the value of *column*. You can specify a number, a character string enclosed in single quotation marks, or 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 *column*.

---

## DATAGRID\_GETLASTMISSINGINDEX Function

Returns the index of the last row that contains a missing value in the specified column.

Category: Information

Returned data type: INTEGER

Note: This function returns a zero if the specified column does not contain any missing values.

---

## Syntax

**DATAGRID\_GETLASTMISSINGINDEX** (*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\_GETLONG Function

Returns the value of the cell in the specified row and column as a long integer.

Category: Retrieve Values

Returned data type: LONG

---

## Syntax

**DATAGRID\_GETLONG** (*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\_GETSTRING Function

Returns the value of the cell in the specified row and column as a character string.

Category: Retrieve Values

Returned data type: STRING

---

## Syntax

**DATAGRID\_GETSTRING** (*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 Function

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***

***dataGrid2***

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\_INITIALIZE Function

Replaces all of the values in the specified data grid with the specified default values.

Categories: Create, Update, or Delete  
Set Values

Returned data type: INTEGER

CAUTION: This function replaces any values that are currently in the data grid.

## Syntax

**DATAGRID\_INITIALIZE** (*dataGrid*, *character-value*, *numeric-value*)

## Required Arguments

***dataGrid***

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

***character-value***

specifies a default value for character columns.

***numeric-value***

specifies a default value for numeric columns.

## DATAGRID\_INNERJOIN Function

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*, *target\_dataGrid*)

## 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.

***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\_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 Function

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******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.

***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\_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 Function

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

Category: Information

Returned data type: INTEGER

## 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 [“Comparing Values in Data Grids” on page 30](#). 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, or the name of a variable or of an expression. Trailing blanks are included in the comparison. The value that you specify must be or must evaluate to the same data type as *filterColumn*.

---

## DATAGRID\_MAX Function

Returns the maximum value for the specified column.

Category: Statistical

Returned data type: DOUBLE

---

### 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\_MAXBOOL Function

Returns the maximum value for the specified column.

Category: Statistical

Returned data type: INTEGER

Note: If any row in the specified column contains a 1, this function returns 1. If all rows in the specified column contain zeros, this function returns 0.

---

### Syntax

**DATAGRID\_MAXBOOL** (*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\_MAXINT Function

Returns the maximum value for the specified column.

Category: Statistical

Returned data type: INTEGER

---

### Syntax

**DATAGRID\_MAXINT** (*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\_MAXLONG Function

Returns the maximum value for the specified column.

Category: Statistical

Returned data type: LONG

---

### Syntax

**DATAGRID\_MAXLONG** (*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\_MAXSTRING Function

Returns the maximum value for the specified column.

Category: Statistical

Returned data type: STRING

Note: This function sorts the specified column in ascending order and returns the last value.

---

## Syntax

**DATAGRID\_MAXSTRING** (*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 Function

Returns the mean value for the specified column.

Category: Statistical

Returned data type: DOUBLE

## 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 Function

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 Function

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\_MINBOOL Function

Returns the minimum value that appears in the specified column.

Category: Statistical

Returned data type: INTEGER

Note: If any row in the specified column contains a 0, this function returns 0. If all rows in the specified column contain a 1, this function returns 1.

---

## Syntax

**DATAGRID\_MINBOOL** (*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\_MININT Function

Returns the minimum value that appears in the specified column.

Category: Statistical

Returned data  
type: INTEGER

---

## Syntax

**DATAGRID\_MININT** (*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\_MINLONG Function

Returns the minimum value that appears in the specified column.

Category: Statistical

Returned data  
type: LONG

---

## Syntax

**DATAGRID\_MINLONG** (*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\_MINSTRING Function

Returns the minimum value that appears in the specified column.

Category:	Statistical
Returned data type:	STRING
Note:	This function sorts the specified column in ascending order and returns the first value.

---

## Syntax

**DATAGRID\_MINSTRING** (*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\_MULTISORT Function

Sorts a data grid based on the values of one or more columns, and then populates the target data grid with the sorted data.

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

---

## Syntax

**DATAGRID\_MULTISORT** (*source\_dataGrid*, *sort\_column\_1*, *sort\_order\_1* < , *sort\_column\_2* > < , *sort\_order\_2* > ..., *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\_n***

specifies the column whose values are used to sort the rows in the target data grid.

**sort\_order\_n**

specifies whether the target data grid is sorted in ascending or descending order according to the values in the column specified by *sort\_column\_n*. 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.

You can specify up to six pairs of *sort\_column\_n* and *sort\_order\_n* parameters.

**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\_NMISS Function

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

Category: Statistical

Returned data type: INTEGER

---

### 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 Function

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

Category: Information

Returned data type: INTEGER

---

## 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 Function

Renames the specified column in the specified data grid.

Category: Rename

Returned data type: INTEGER

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

---

## Syntax

**DATAGRID\_RENAMECOLUMN** (*dataGrid*, *oldName*, *newName*)

## 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 Function

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

**DATAGRID\_RIGHTJOIN** (*dataGrid1*, *dataGrid2*, *keyColumn1*, *keyColumn2*, *target\_dataGrid*)

## 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.

***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 Function

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 Function

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

Category: Set Values

Returned data type: INTEGER

---

## Syntax

**DATAGRID\_SETALL** (*dataGrid*, *column*, *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 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 evaluate to the same data type as *column*.

---

## DATAGRID\_SETBOOL Function

Assigns the specified Boolean value to the specified row and column.

Category: Set Values

Returned data type: INTEGER

Note: A zero return value indicates success, and a non-zero return code indicates failure.

---

## Syntax

**DATAGRID\_SETBOOL** (*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. The column must be a Boolean column.

### ***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 numeric boolean value to be assigned to *column*. You can specify 1 for True or 0 for False. Alternatively, you can specify the name of a boolean variable or a numeric variable that evaluates to 0 or 1.

**Note** Do not specify the character strings "True" or "False".

---

## DATAGRID\_SETDOUBLE Function

Assigns the specified decimal value to the specified row and column.

Category: Set Values

Returned data type: INTEGER

Note: A zero return value indicates success, and a non-zero return code indicates failure.

---

## Syntax

**DATAGRID\_SETDOUBLE** (*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. The column must be a numeric column.

***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, the name of a numeric variable, or an expression. The value that you specify must be or must evaluate to a numeric value.

---

## DATAGRID\_SETINT Function

Assigns the specified integer value to the specified row and column.

Category: Set Values

Returned data type: INTEGER

Note: A zero return value indicates success, and a non-zero return code indicates failure.

---

## Syntax

**DATAGRID\_SETINT** (*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. The column must be an integer column.

**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 numeric literal, the name of a variable, or an expression. The value that you specify must be or must evaluate to a numeric value.

**Note** If you specify a decimal value, only the integer part (characteristic) is assigned as the cell value. The fractional part (mantissa) is truncated.

---

## DATAGRID\_SETLONG Function

Assigns the specified long integer value to the specified row and column.

Category: Set Values

Returned data type: INTEGER

Note: A zero return value indicates success, and a non-zero return code indicates failure.

---

## Syntax

**DATAGRID\_SETLONG** (*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. The column must be a numeric column.

**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 numeric literal, the name of a variable, or an expression. The value that you specify must be or must evaluate to a numeric value.

**Note** If you specify a decimal value, only the integer part (characteristic) is assigned as the cell value. The fractional part (mantissa) is truncated.

---

## DATAGRID\_SETSTRING Function

Assigns the specified character string to the specified row and column.

Category: Set Values

Returned data type: INTEGER

Note: A zero return value indicates success, and a non-zero return code indicates failure.

---

### Syntax

**DATAGRID\_SETSTRING** (*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. The column must be a character column.

***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 character string enclosed in single quotation marks, the name of a variable, or an expression. The value that you specify must be or must evaluate to a character string.

---

## DATAGRID\_SORT Function

Sorts a data grid based on the values in a single column, and then populates the target data grid with the sorted data.

Category: Subset and Sort

Returned data type: INTEGER

Note: This function returns zero if the sort is successful.

## 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.

## DATAGRID\_STDDEV Function

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

Category: Statistical

Returned data type: DOUBLE

## 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 Function

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*)

**DATAGRID\_SUBSETBYVALUE** (*source\_dataGrid*, *criteria\_dataGrid*, *target\_dataGrid*)

### Required Arguments

***source\_dataGrid***

specifies the name of the source 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.

**Restriction** You can specify either the *filterColumn*, *operator*, and *varOrValue* arguments or the *criteria\_dataGrid* argument. For more information, see, [“Details” on page 85](#).

***operator***

specifies one of the following operators shown in [“Comparing Values in Data Grids” on page 30](#). 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, or 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*.

***criteria\_dataGrid***

specifies the filter criteria for cases for which you want to filter the source data grid on multiple columns. All of the criteria that are specified in the criteria data grid must evaluate to true in order for a row in the source data grid to be added to the target data grid.



**Restriction** You can specify either the *criteria\_dataGrid* argument or the *filterColumn*, *operator*, and *varOrValue* arguments. For more information, see, “[Details](#)” on page 85.

***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 uses the comparison criteria that you specify to subset the data in the source data grid. You can specify the comparison criteria in one of two ways:

- Use the *filterColumn*, *operator*, and *variableOrValue* arguments to filter the source data grid based on the values in a single column. For an example, see “[Example 1: Subset Based on a Single Column](#)” on page 85.
- Use the *criteria\_dataGrid* argument to filter the source data grid based on the values of more than one column. The criteria data grid must contain only the three columns that specify the filter criteria. The content of these columns is the same content that you would specify for the *filterColumn*, *operator*, and *variableOrValue* columns. See the description of these arguments in “[Required Arguments](#)”.

The metadata for this data grid could be defined as follows:

```
[{"metadata":[{"filterCol":"string"}, {"op":"string"}, {"value":"string"}]}
```

For an example, see “[Example 2: Subset Based on Multiple Columns](#)” on page 85.

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

## Examples

### Example 1: Subset Based on a Single Column

The following example populates the CUST5 data grid variable with the subset of records from the CUSTOMERS data grid for which the `income > 50000` comparison is true:

```
DATAGRID_SUBSETBYVALUE(customers,income,'gt','50000',cust5)
```

### Example 2: Subset Based on Multiple Columns

The following example populates the CUST5 data grid variable with the subset of records from the CUSTOMERS data grid for which both of the comparisons `income > 50000` and `loanAmt < equity` evaluate to true:

```
DATAGRID_SUBSETBYVALUE(customers, subCriteria, cust5)
```

The SUBCRITERIA data grid is defined as follows:

```
subCriteria = cat(' [{"metadata": [{"filterCol": "string"},
                        {"operator": "string"}, {"varOrValue": "string"}]}, ',
                  ' {"data": [{"income", "gt", "50000"}, ',
                  ' ["loanAmt", "lt", "equity"]}]}');
```

---

## DATAGRID\_SUM Function

Returns the sum of the values in the specified column.

Category: Statistical

Returned data type: DOUBLE

---

### 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\_SUMLONG Function

Returns the sum of the values in the specified column.

Category: Statistical

Returned data type: LONG

---

### Syntax

**DATAGRID\_SUMLONG** (*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 Function

Populates the target data grid with the rows from the source data grid that contain the highest *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 descending order of the specified column.

---

## 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 Function

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.