User Access Matrix – Column and Row Level Security for CAS Tables

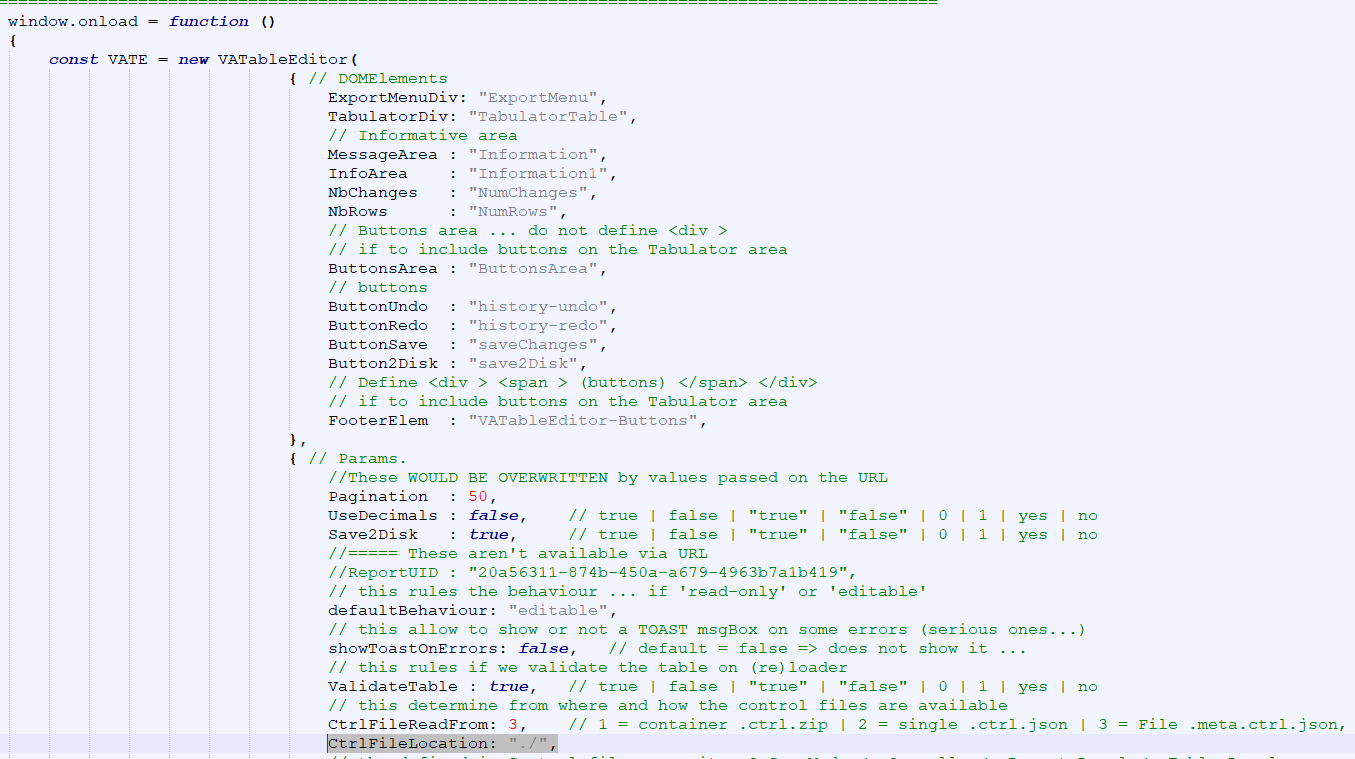
Documentation

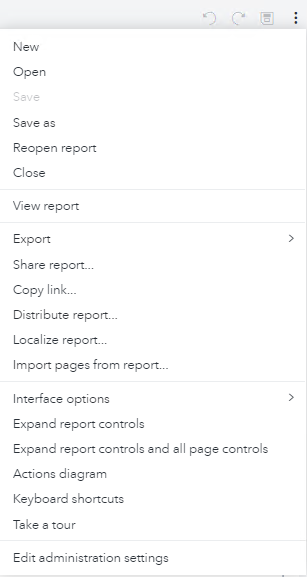
Product: Information Catalog

Version: SAS Viya 4

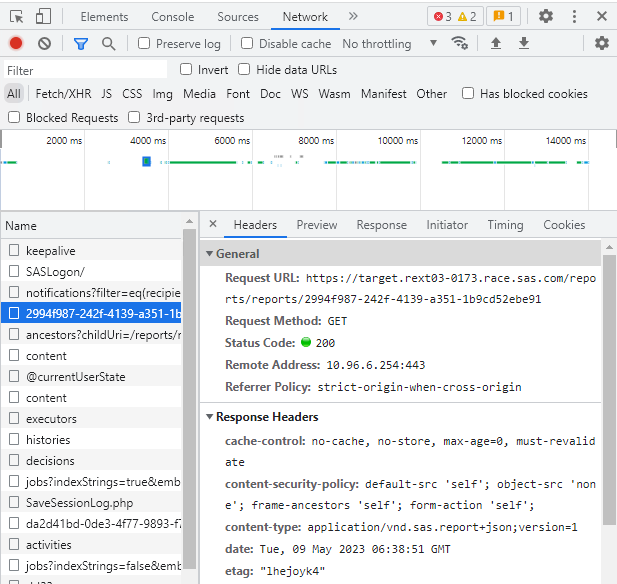
Date: 12/04/2023

**Setting up the VA Table Editor**

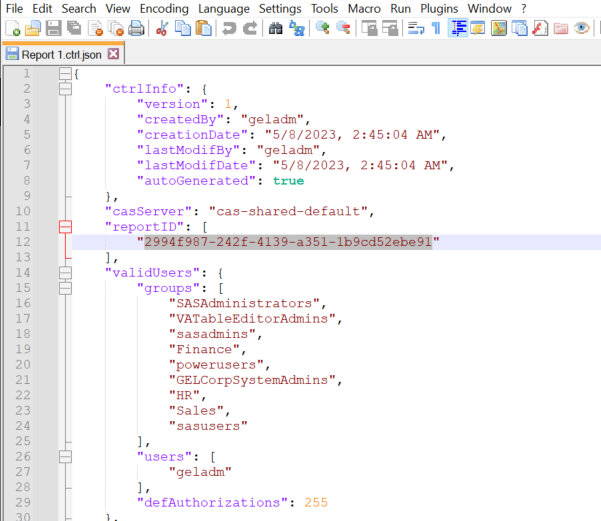
1. Unzip **va-table-editor-master.zip** to **D:\**
2. Navigate to **D:\va-table-editor-master\App** and select **VATableEditor\_DDC.html**
3. Scroll to the OnLoad and OnLoad functions and edit the ctrlFileLocation
4. ****
5. Go to SAS Visual Analytics and create a new report
6. Click **Objects** and select **Data-driven content**
7. Save the report
8. Retrieve the report id
9. Press **Ctrl + Shift + I**
10. Click the arrow followed by **Network**
11. Click on the clear button
12. Then, click on the more button followed by the option **Reopen report**



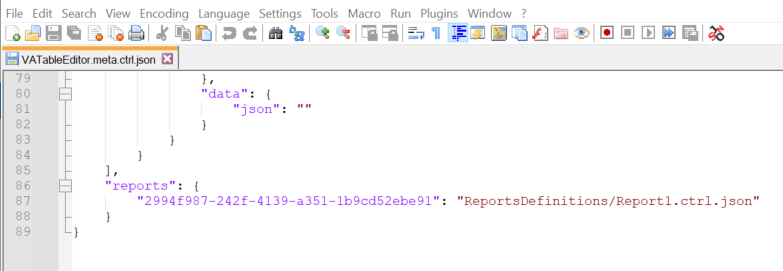
1. Find the Request URL containing the Report ID (i.e., **https://target.rext03-0173.race.sas.com/reports/reports/2994f987-242f-4139-a351-1b9cd52ebe91**, where **2994f987-242f-4139-a351-1b9cd52ebe91** is the report id)



1. Navigate to **D:\va-table-editor-master.zip \App\ReportDefinitions**, and click on **Report 1 ctrl.json** to edit in Notepad++
2. Change the report ID in the json file to the report ID of the report, rename the file to the name of the report saved



1. Navigate to **D:\va-table-editor-master.zip\App** and click on **VATableEditor.meta.ctrl.json**
2. Scroll to the bottom of the json file and edit the **reports** field to add the report ID of the saved report and the path



1. Zip the edited **va-table-editor-master** folder
2. In this section, we will be setting up VA Table Editor for the interface to view and edit the specific column and row level access permissions.
3. Using MobaXterm, upload **va-table-editor-master.zip** to **/home/cloud-user/**
4. Switch to root using   
   sudo su
5. Unzip **va-table-editor-master.zip**

unzip va-table-editor-master.zip

1. Change the current directory to **va-table-editor-master/Viya4/**

cd va-table-editor-master/Viya4/

1. Run **install\_vate.sh** to install VA Table Editor

./install\_vate.sh

1. Run **register\_client.sh** to register the client

./register\_client.sh

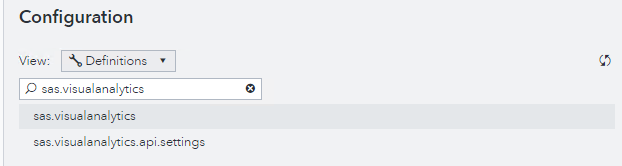
Note that if you encounter this when running **install\_vate.sh** and/or **register\_client.sh**:

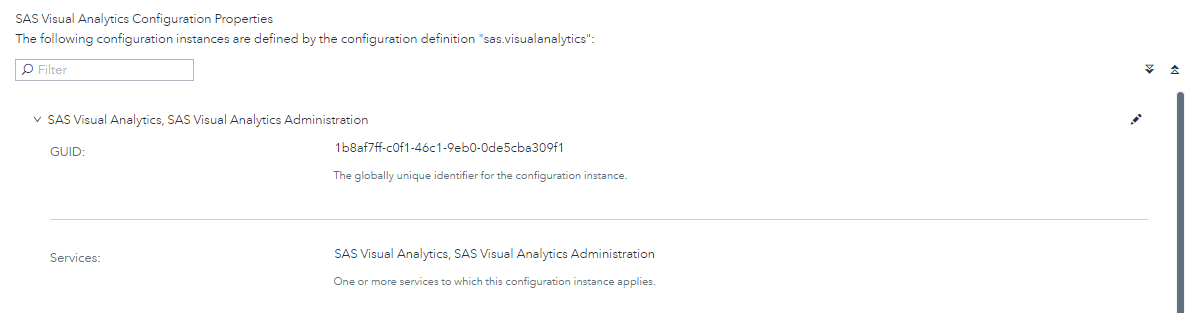


To change the permissions of **install\_vate.sh** and/or **register\_client.sh**

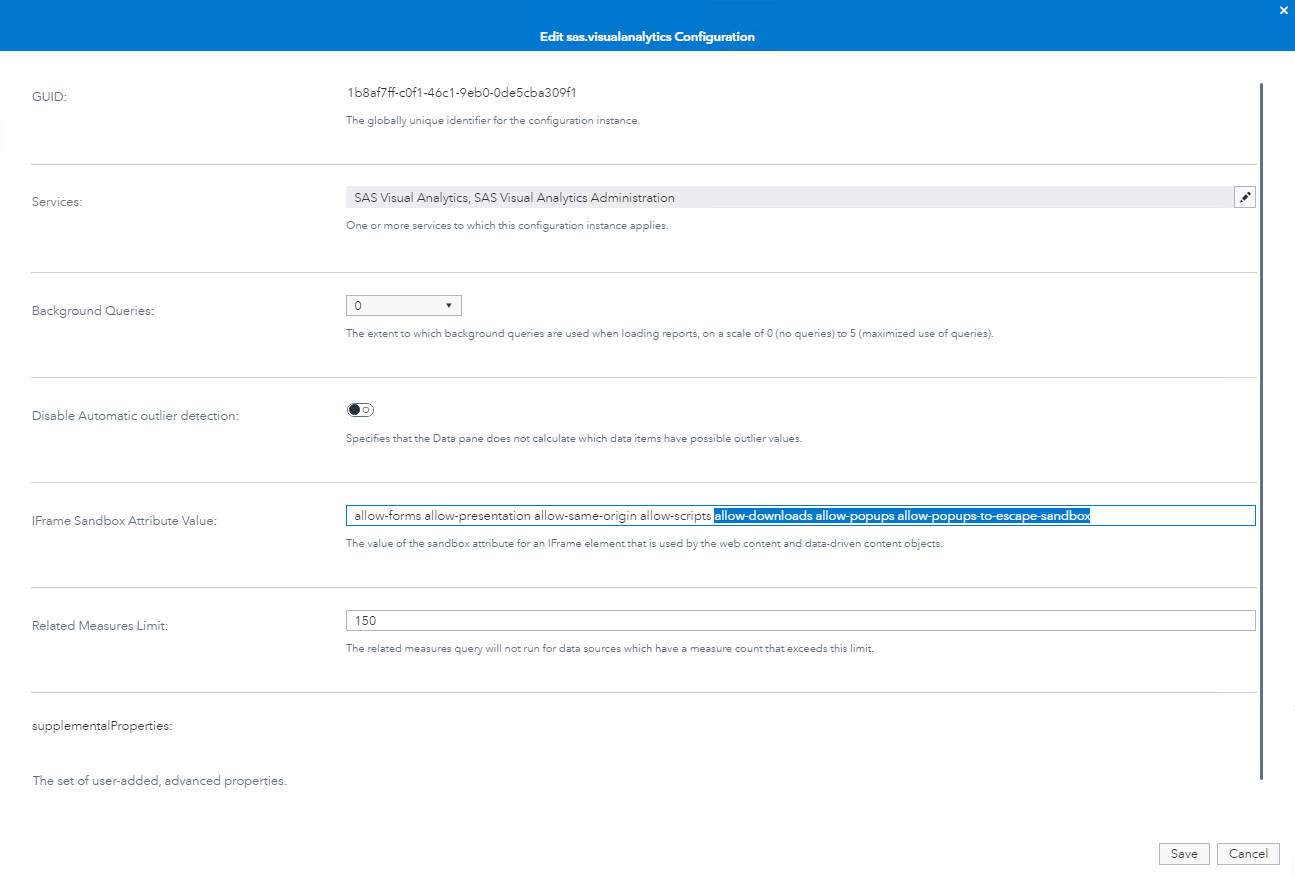
chmod 777 ./install\_vate.sh  
chmod 777 ./register\_client.sh

1. Login to SAS Environment Manager. Go to **Configuration**, under the View dropdown, select **Definitions**. Search for **sas.visualanalytics**, and edit **SAS Visual Analytics, SAS Visual Analytics Administration**.

****

****

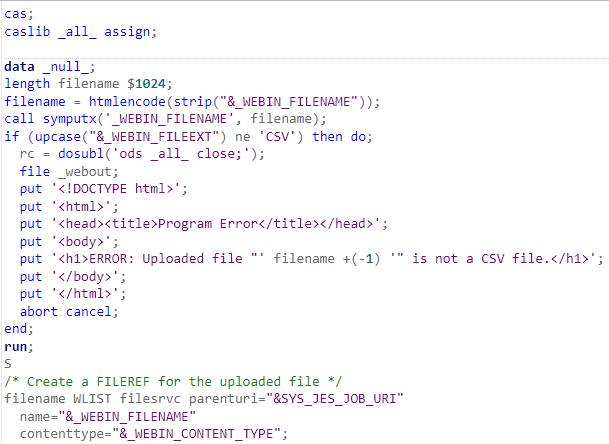
1. In the pop-up, under **IFrame Sandbox Attribute Value**, add “**allow-downloads allow-popups allow-popups-to-escape-sandbox**” and click **save**.

****

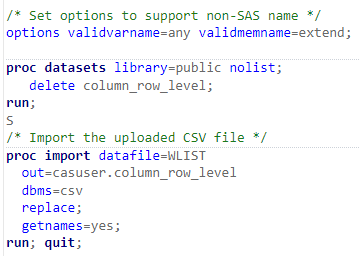
1. Go to SAS Visual Analytics and replace **https://BASEURL/SASVisualAnalytics/resources/custom\_table.html** to **https://BASEURL /vate/App/VATableEditor\_DDC.html?PageSize=100** (i.e., BASEURL = target.rext03-0173.race.sas.com)

**Setting up the Job Executions**

1. Create a new job execution called ‘*Upload UAM’*
2. In this section the process and the logic behind the SAS codes of **Upload UAM** will be explained
3. First, we initialize the CAS session and assign all available CAS libraries to allow access to all libraries. The data step then code checks if the uploaded file is a CSV file and displays an error message in HTML format if it is not. We also define a file reference named ‘**WLIST**’ and specify the file path and name of the uploaded CSV file.

****

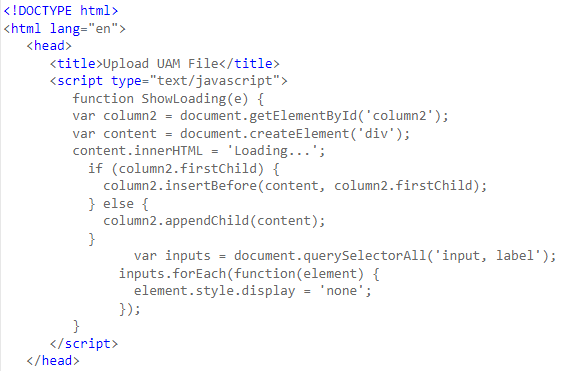
1. The default rules for variable and dataset naming are changed to allow any characters and extends the length of dataset names, providing more flexibility in naming variables and datasets. After, e delete the existing promoted dataset in the caslib called **column\_row\_level**, which we will replace shortly. The **PROC IMPORT** step will then import the csv file into the CAS table **column\_row\_level** located in the library **CASUSER.**

****

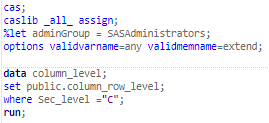
1. The data step creates the dataset **column\_row\_level** in the public library from the dataset **casuser.column\_row\_level**, the **caslib** column is renamed, and a unique ID is generated based on the values of other variables. The next data step will then generate HTML code to the output of the job execution to display a success message indicating that a file has been uploaded and prompts the user to reopen the report.

****

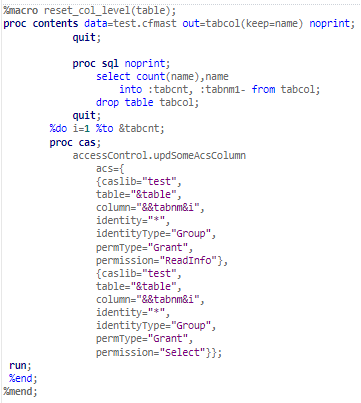
1. Right click the **Upload UAM** job, select **Edit**, then **HTML form**
2. In this section the process and the logic behind the HTML form of **Upload UAM** will be explained
3. The head section of the code contains the title of the page, along with a JavaScript function **ShowLoading** that is triggered when the form is submitted. The function displays a "Loading..." message and hides the input fields and labels while the file is being uploaded and processed.



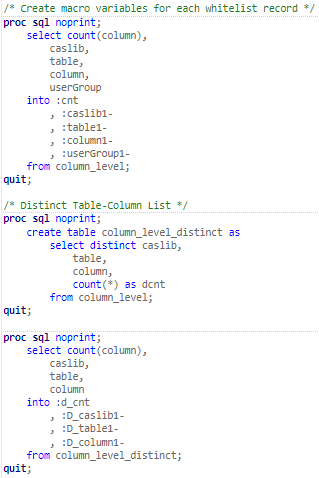
1. The body section of the code contains the HTML form used for file uploading. It includes a file input field where the user can select the CSV file to upload. Upon submitting the form, the **ShowLoading** function is triggered. ****
2. Create a new job execution called ‘*Run UAM’*
3. In this section the process and the logic behind the SAS codes of **Run UAM** will be explained
4. First, we initialize the CAS session and assign all available CAS libraries to allow access to all libraries. Then we define the macro variable ‘**adminGroup**’ and assign it to the value ‘**SASAdministrators**’. The default rules for variable and dataset naming are changed to allow any characters and extends the length of dataset names, providing more flexibility in naming variables and datasets. The DATA step creates dataset **column\_level** by selecting rows from the **column\_row\_level** table where *Sec\_level = ‘C’.*

****

1. Following that, we define a macro program called **reset\_col\_level**, which loops through the columns. The PROC CAS step uses the **accessControl.updSomeAcsColumn** action to grant Read Info & Select permission. The purpose of this is to reset the access control settings for all columns, so that column level security initially set can be revoked.

****

1. Next, we store the values and count of the **column\_level** table as macro variables. We also create another dataset named **column\_level\_distinct** by selecting distinct columns and counts the occurrences of distinct combinations of the columns. Then, it stores the values of **column\_level\_distinct** into macro variables.

****

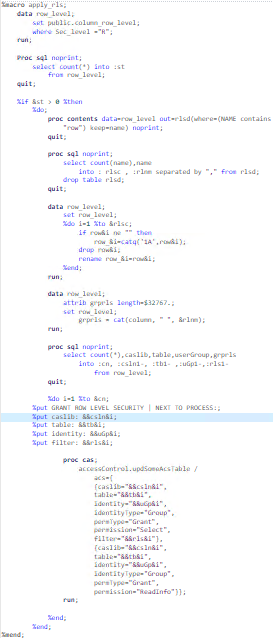
1. Following that, we define a macro program called **looper\_DenyALL** which loops through the distinct column-level rules. For each iteration, the macro variables are used to retrieve the caslib, table, and column values. The **PROC CAS** step uses the **accessControl.updSomeAcsColumn** action to grant Read Info & Select permission to SASAdministrators and denying Read Info & Select permission to all other identities for the specified column.



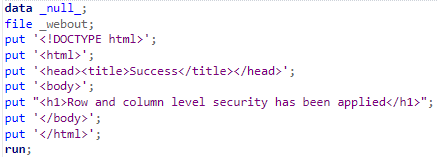
1. Subsequently, we define a macro program called **looper\_GrantWhitelist** which loops through the column-level rules. For each iteration, the macro variables are used to retrieve the caslib, table, and column values. The **PROC CAS** step uses the **accessControl.updSomeAcsColumn** action to grant Read Info and Select permissions to the specified user groups for the specified columns.

****

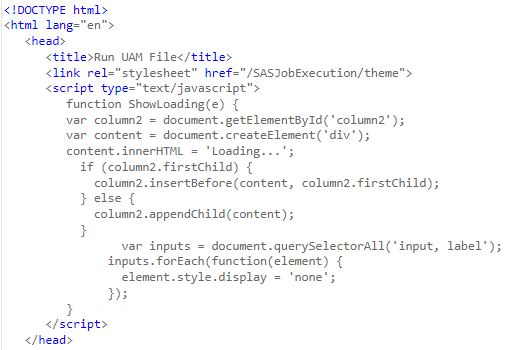
1. Finally**,** we define a macro program called **apply\_rls** which loops through the row-level rules. A dataset called **row\_level** is created where *Sec\_level = ‘R’*. A **PROC SQL** statement counts the number of rows in **row\_level** and runs the subsequential codes if **row\_level** is not empty. The **DATA** step modifies the **row\_level** dataset by creating a variable **grprls** which represents the filter condition for row-level security. The values of **row\_level** is then stored as macro variables. Finally, The **PROC CAS** step uses the **accessControl.updSomeAcsTable** action to grant Read Info and Select permissions to the specified user groups based on the defined filter condition.

****

1. The next data step will then generate HTML code to the output of the job execution to display a success message indicating that row and column level security has been applied.

****

1. In this section the process and the logic behind the HTML form of **Run UAM** will be explained
2. The head section of the code contains the title of the page, along with a JavaScript function **ShowLoading** that is triggered when the form is submitted. The function displays a "Loading..." message and hides the input fields and labels while the file is being uploaded and processed.

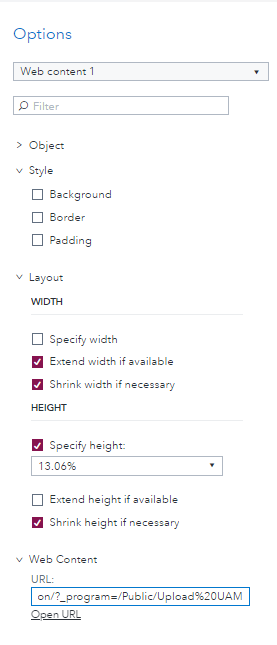


1. The body section of the code contains the HTML form used for running a UAM file. The form includes a submit button labelled “Run UAM”. Upon submitting the form, the **ShowLoading** function is triggered.

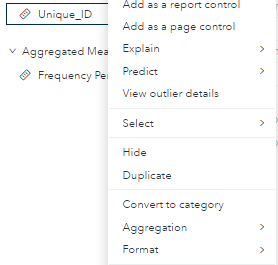
****

**Building the VA Report**

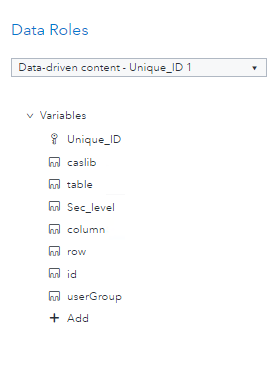
1. In the report previously created, click **Objects,** and select **Web Content**
2. Ensure the web content object is placed above the Data Driven Content and drag the web content until it is the smallest height.
3. Click **Options**, and under **Web Content** insert the **Upload UAM** job execution URL to the URL

****

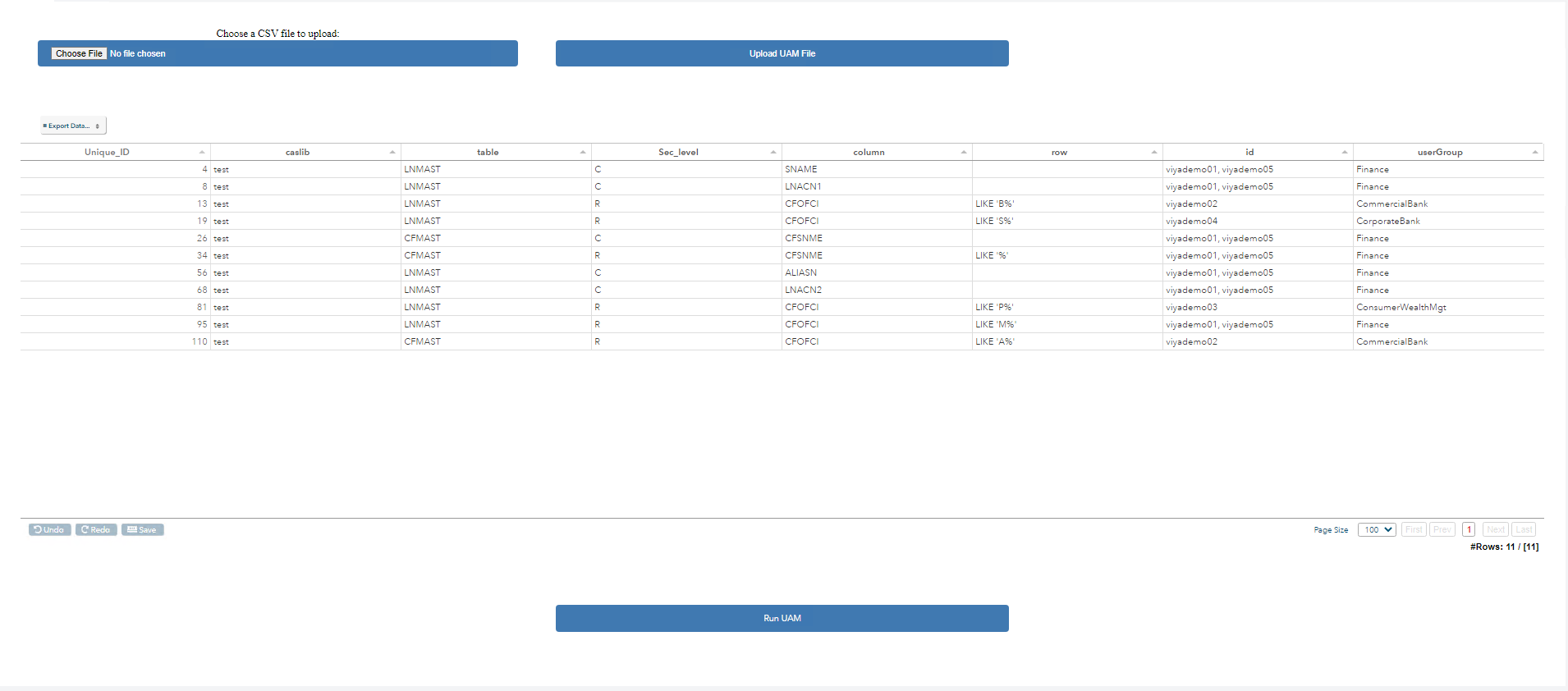
1. Upload the UAM csv file using the Upload UAM job in the report
2. Once the file has been uploaded. Restart the report and click **Data**
3. Add **public.column\_row\_level** to the report
4. Right click on **Unique\_ID** and select **Convert to category**

****

1. Once **Unique\_ID** is converted into a Category, right click on **Unique\_ID** again and select **Set as unique row identifier**
2. On the Data Driven Content object, click **Assign Data**, and add all variables except **Frequency** and **Frequency Percent**. Click **OK.**
3. Click Roles and drag rearrange the variable such that they are in this order.



1. Click Options, and under Web Content, replace the URL to **https://BASEURL /vate/App/VATableEditor\_DDC.html?PageSize=100** (i.e., BASEURL = target.rext03-0173.race.sas.com)
2. Go back to **Options** and select **Web Content**
3. Ensure the new web content object is placed below the Data Driven Content and drag the web content until it is the smallest height.
4. Click **Options**, and under Web Content insert the Run UAM job execution URL to the URL
5. The final report has the following layout

****