



ONE – Workshop

DQ Aggregation Rules

Prepared for: v15.4

Prepared by: Ataccama

Dated: November 2024

Contents of the Document

Introduction	3
1 – Create a DQ aggregation rule to check the uniqueness of a group of values	4
1a – Creating a Group	5
1b – Developing Rule Logic	6
1c – Adding Aggregation rule to a Monitoring Project	8
2 – Create a DQ aggregation rule to compare each value with the average value	10
2a – Implementing Rule Logic	10
2b – Adding a rule to Monitoring Project	12
Conclusion	12

Introduction

Whenever a DQ check logic requires grouping and/or aggregation functions, you are required to create DQ Aggregation rules.

This can be achieved in three possible ways:

- Grouping of the data (using **Group by** section) and directly using it in the Rule Logic – **Condition Builder**.
- Grouping of the data (using **Group by** section) followed by the use of aggregate functions in Rule Logic – **Advanced Expressions**.
- Direct use of aggregate functions in the Rule Logic – **Condition Builder/Advanced Expressions**.

In this workshop, we will demonstrate aggregation rules in the following two examples:

▶ Check whether a combination of two attributes is unique within the whole dataset.

Calculate an average value of a group and see which values are outside its scope with the threshold of 10%.

1 – Create a DQ aggregation rule to check the uniqueness of a group of values

Firstly, we will see how an aggregation rule can combine several values together and evaluate them against the dataset. Values from a combination of '**customername**' and '**phone**' columns will be used for this example.

- › Navigate to the **Knowledge Catalog** and review the **customers** catalog item.
- › Look at the results for the column's **customername** and **phone**. In the **customers** table, we are supposed to have one record per customer. Therefore, we want to evaluate the uniqueness of the combination of these two across the dataset.

Sources > xy_training > postgres > public
customers

Overview History **Data** Data Structure Lineage Data Quality Profile & DQ insights Relationships Data Export & Transformations

This is a Data Sample
Displayed data is live. The data displayed is a sample of the first 50 lines of the data set.

customernumber	Abc customername	Abc contactlastname	Abc contactfirstname	Abc phone	Abc addressline1
	Surname		First name	+1	Street with number
103	Atelier graphique	Schmitt	Carine	40.32.2555	54, rue Royale
112	Signal Gift Stores	King	Jean	7025551838	8489 Strong St.
114	Australian Collectors, Co.	Ferguson	Peter	03 9520 4555	636 St Kilda Road
119	La Rochelle Gifts	Labruno	Janine	40.67.8555	67, rue des Cinquante Otages
121	Baane Mini Imports	Bergulfsen	Jonas	07-98 9555	Erling Skakkes gate 78

Let's start with creating a new rule and defining its logic:

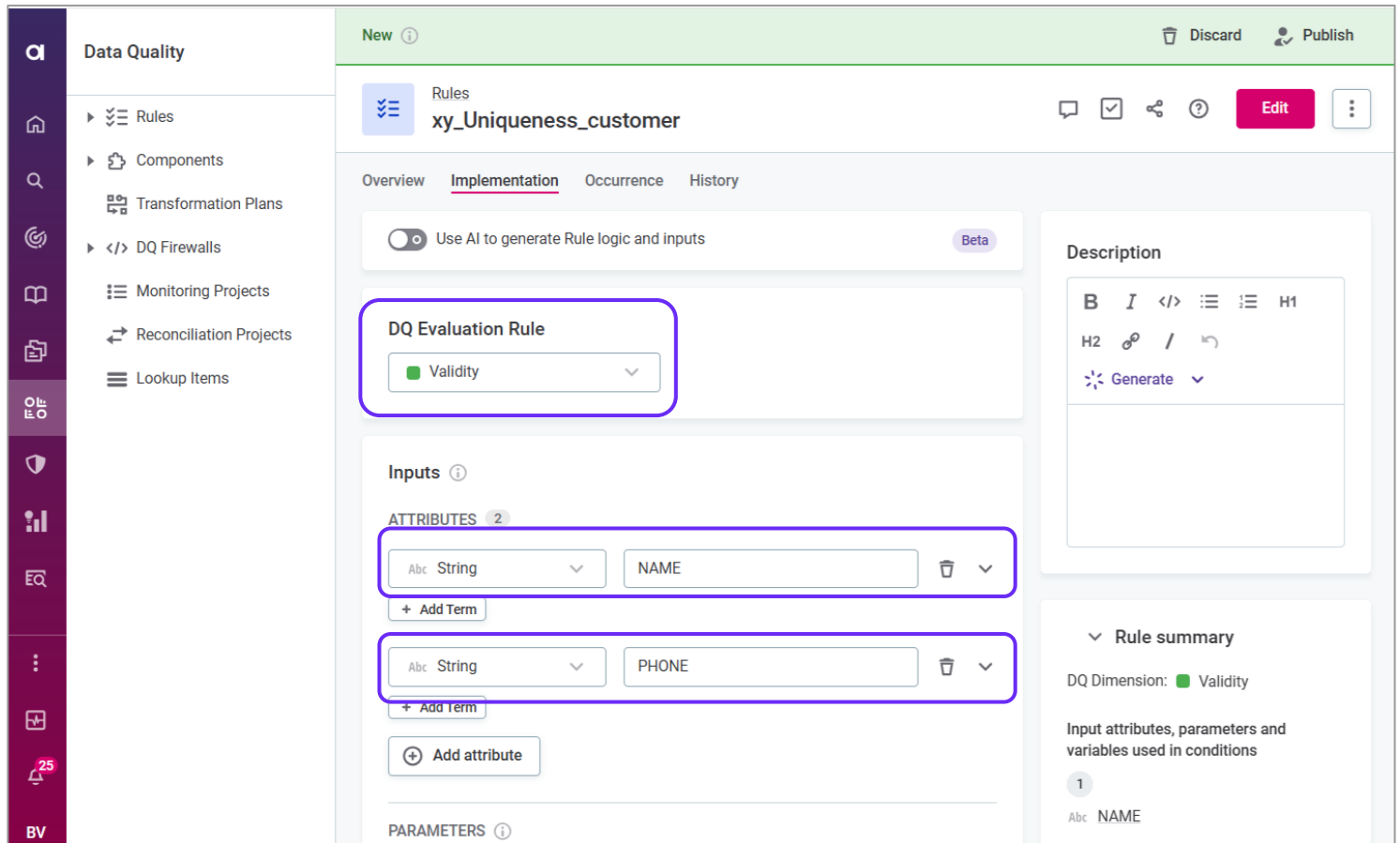
- › Navigate to the **Data Quality** → **Rules** section.
- › Create a new rule and provide a **Name**, for example: **<prefix>_Uniqueness_customer**.
- › Select the **Rule Type** as **DQ Evaluation: Validity**.
- › (Optional) Add a **Description** if needed.
- › Save your changes. You will then be redirected to the **Implementation** tab.

Once the rule is created, we need to specify the rule's logic definition:

- › In the **Implementation** tab, go to the Input Attributes section.

› Define the following two input attributes:

- **NAME** of type **STRING**.
- **PHONE** of type **STRING**.



› Right next to it, select the **Aggregation rule** option from the drop-down menu (it will become available after selecting this rule type).

A new section named **Group by** will be made visible.

Using the method that follows, we can use the **Group by** to group a single selected attribute or several attributes. These groups will be available as input in the Rule **Logic** section, under the **Group by results** provided.

The following instructions will show how to form a group of input values and use it for further evaluation:

1a - Creating a Group

- › Group by the two defined input attributes.

The image shows three sequential screenshots of a rule editor interface, connected by blue arrows, illustrating the steps to create a group of input attributes.

Step 1: The 'Rule Logic' dropdown menu is open, showing options: 'Rule' (selected), 'Component', and 'Aggregation rule'. The 'Aggregation rule' option is highlighted with a red box.

Step 2: The 'Rule Logic' dropdown is now set to 'Aggregation rule'. Below it, the 'Group By' section is visible. The 'Inputs' list contains 'NAME' and 'PHONE'. The 'Group By' field is highlighted with a red box, and a red '+' button is shown next to it.

Step 3: The 'Group By' field now contains both 'NAME' and 'PHONE' attributes, separated by a '+' sign, indicating they have been grouped together. The 'Group By' field is highlighted with a red box.

1b – Developing Rule Logic

- › Select the Condition Builder and check the emptiness of the grouped attributes first (as a best practice). In the WHEN section, select each of these attributes from the list of available inputs and retain the default settings of the WHEN section which is “is empty”. Then, use the ‘+ **AND**’ button to add the same condition for the other inputs. Don’t forget the explanation code!

The screenshot displays the 'Rule Logic' configuration interface. At the top, there are tabs for 'Rule Logic', 'Aggregation rule', and 'Test Rule'. Below this is a 'Group By' section with a dropdown menu showing 'NAME' and 'PHONE' attributes. The main rule configuration is titled '1 IS_EMPTY'. It features a 'WHEN' section with two conditions: 'NAME is empty' and 'PHONE is empty', connected by an 'AND' operator. Below the 'WHEN' section is a '+ Add expression' button. The 'THEN' section includes a 'Result' dropdown set to 'Invalid', a 'Score' field with the value '1000000', and an 'Explanation' field with the value 'IS_EMPTY'. At the bottom left, there is a '+ Add condition' button.

Now, use the ‘+ **Add condition**’ button and select the **Group by results** which is **NAME, PHONE** from the list of available inputs.

2 CONDITION

Ask AI

WHEN

Abc NAME value is empty

Inputs

Abc NAME

Abc PHONE

NAME, PHONE

Explanation

CONDITION

- Complete the WHEN configuration by changing the already existing condition to **'IS_NOT_UNIQUE'**, score: **100000**
- Make sure correct values are filled in the **Explanation Code** for INVALID and VALID statements in the THEN section.

2 IS_NOT_UNIQUE

Ask AI

WHEN

NAME, PHONE is not unique

AND

THEN

Result Score Explanation

Valid Invalid 100000 IS_NOT_UNIQUE

- Test the rule with appropriate values and publish it.

Test

Whole rule

NAME	PHONE	Message
ABC	123	Valid
XYZ	567999.9	IS_NOT_UNIQUE
ABC	456	Valid
XYZ	567999.9	IS_NOT_UNIQUE
STRING	STRING	IS_EMPTY

New Row

1c – Adding Aggregation rule to a Monitoring Project

To see how this rule works, link it to one of the existing Monitoring projects (e.g. Training project).

- › Go to the configuration and results tab.
- › Select and open the **customers** catalog item.
- › Add the rule in the Applied DQ Checks to either the **customername** or **phone** attribute.

The screenshot shows the Data Quality Monitoring interface for the 'xy_Training Project'. The left sidebar contains navigation options: Rules, Detection Rules, DQ Evaluation Rules, DQ Dimensions, Components, Transformation Plans, DQ Firewalls, Monitoring Projects, Reconciliation Projects, and Lookup Items. The main area displays a table of attributes and their associated rules. The 'customername' attribute is highlighted, and the '+ Add' button is circled in red. A modal window titled 'Add Rule for data type "string" to "customername"' is open, showing a list of rules. The rule 'xy_Uniqueness_customer' is highlighted, and the 'Configure' button is circled in red.

Name	Terms	Filter by	Structure	Anomaly Detection	Overall Quality	Applied Rules
123 customernumber			Mandatory	+ Enable Detection		+ Add
Abc customername			+ Make Mandatory	+ Enable Detection		+ Add
Abc contactlastname	Surname		+ Make Mandatory	+ Enable Detection	84%	+ Add 84%
Abc contactfirstname	First name		+ Make Mandatory	+ Enable Detection	91%	+ Add 91%
Abc phone	[North America] Phone Number		+ Make Mandatory	+ Enable Detection		+ Add
Abc addressline1	Street with number		+ Make Mandatory	+ Enable Detection		+ Add

Add Rule for data type "string" to "customername"

All Rules Suggestions 5 + Create Rule

xy

All dimensions

Name	Input type	Dimension	
xy_Product Line	VALUE (STRING)	Validity	Apply Rule
xy_State code	STATE (STRING)	Validity	Apply Rule
xy_Uniqueness_customer	NAME (STRING) +1	Validity	Configure

xy_Uniqueness_customer

Label

xy_Uniqueness_customer

Input configuration

Abc NAME *

customername

Abc PHONE *

phone

Notification settings

☒

⚠

Trigger Warning when main result is less than or equal to

70

%

Apply Rule

- Now you should see the rule in DQ Checks of both attributes (**customername** and **phone**).
- Publish** the changes and **run** the project to see the results.
- Choose the **attributes with DQ checks** option from the drop-down menu to display a list of **only** attributes with applied rules.

Data Quality

Rules

Detection Rules

DQ Evaluation Rules

DQ Dimensions

Components

Transformation Plans

DQ Firewalls

Monitoring Projects

Reconciliation Projects

Lookup Items

Monitoring projects

xy_Training Project

Overview

Configuration & Results

Report

Export

Notifications

History

xy_Training Project > customers

November 22, 2024, 3:43:02 PM (Latest)

Great! Your data is in good shape

Records 122

Attributes 15

Checks 9

Structure

OK

1 check

Anomaly Detection

OK

1 attribute and 1 catalog item check

Open profile inspector

Data Quality

OK

6 checks

Show invalid samples

Overall

VAL

ACC

74%

Filter Attributes and Rules

Attributes with DQ ...

Check for Rule suggestions

Standard view

Hidden columns

Name	Terms	Filter by	Structure	Anomaly Detection	Overall Quality	Applied Rules
Abc customername			+ Make Mandatory	+ Enable Detection	100%	+ Add 100% xy_Uniqueness_customer
Abc contactlastname	Surname		+ Make Mandatory	+ Enable Detection	84%	+ Add 84% validation Surname 93% accuracy Surname
Abc contactfirstname	First name		+ Make Mandatory	+ Enable Detection	91%	+ Add 91% validation First name 91% accuracy First
Abc phone	[North America] Phone Number		+ Make Mandatory	+ Enable Detection	100%	+ Add 100% xy_Uniqueness_customer

Page 10 of 15

2 – Create a DQ aggregation rule to compare each value with the average value

Navigate again to the **Knowledge Catalog**, catalog item **customers**, and look at the values of the column called **creditlimit**. We want to evaluate **whether the credit limit of each customer is within 60% of the total average credit limit (+/-30% from AVG)**.

2a – Implementing Rule Logic

- › Create a new rule (**<prefix>_Average_creditlimit**) in Data Quality > Rules and select **validity** rule.
- › Choose **Aggregation rule** Rule Logic in implementation tab and leave **Group by** blank.
- › Add the input attributes **CUSTOMERNUMBER(LONG)** and **CREDITLIMIT(FLOAT)**.

The screenshot shows the 'Data Quality' interface with the 'Rules' section expanded. A new rule named 'xy_Average_creditlimit' is being configured. The 'Implementation' tab is selected, showing the 'DQ Evaluation Rule' set to 'Validity'. The 'Inputs' section is highlighted with a purple box and contains two attributes: 'CUSTOMERNUMBER' (Long) and 'CREDITLIMIT' (Float). The right sidebar shows the 'Rule summary' and 'Rule logic' sections.

- › Set the rule logic so that higher or lower the 30% range of the average is considered **INVALID**. This has two conditions:
 - If the value is higher than 30% of the average.
 - Use the below expression or Ask AI to write appropriate code for you.

Expression: $CREDITLIMIT > (AVG(CREDITLIMIT) * 1.3)$

Explanation: IS_HIGHER

Score: 700000

1 IS_HIGHER

Ask AI

WHEN

1

CREDITLIMIT > (AVG(CREDITLIMIT)*1.3)

THEN

Result

Score ⓘ

Explanation

Valid

Invalid

7000000

IS_HIGHER

- If the value is lower than 30% of the average.
- Use the below expression or Ask AI to write appropriate code for you.

Expression: CREDITLIMIT < (AVG(CREDITLIMIT)*0.7)

Explanation: IS_LOWER

Score: 700 000

2 IS_LOWER

Ask AI

WHEN

1

CREDITLIMIT < (AVG(CREDITLIMIT)*0.7)

THEN

Result

Score ⓘ

Explanation

Valid

Invalid

7000000

IS_LOWER

- The rest of the values that are within the range would be considered as **Valid**.

IF none of the conditions above apply THEN

THEN

Result

Score ⓘ

Explanation

Valid

Invalid

0

OTHER

- › Now test the rule to see if it is functioning as intended before publishing it.

Test

Whole rule

CUSTOMERNUMBER	CREDITLIMIT		Message
1	30		IS_LOWER
2	50		IS_LOWER
3	1500		IS_HIGHER
4	56.99		IS_LOWER
5	FLOAT		Valid
6	88		IS_LOWER
7	500		IS_HIGHER
LONG	FLOAT		Valid
New Row			



All by yourself now! Try changing the rule to compare the credit limit of each customer with the average of his state.

2b – Adding a rule to a Monitoring Project

- › In **Monitoring Project**, go to **Training project** > **Configuration & Results** > **customers**
- › Add the rule in the Applied DQ Checks to either the **customernumber** or **creditlimit** attribute and configure the inputs accordingly.
- › **Publish** the changes and **run** the project to monitor the results.

Overview

Configuration & Results

Report

Export

Notifications

History

xx_Training Project > customers

March 4, 2024, 5:21:46 PM (Latest)

Issues found

Records 122Attributes 15Checks 18

Structure
OK
1 check

Anomaly Detection
OK
1 attribute and 1 catalog item check
Open profile inspector

Data Quality
Issues detected
15 checks
Show invalid samples

Overall VAL COM ACC
2%

Filter AttributesAttributes with DQ checks

8 suggestions

Standard viewHidden columns

Name	Terms	Filter by	Structure	Anomaly Detection	Overall Quality	Applied Rules
customernumber			Mandatory		37%	37% xx_Average_creditlimit
customername					100%	100% xx_Uniqueness_Cust... 100% xx_Uniqueness_Cust...
contactlastname	Surname				14%	19% validation First name 84% validation Surname +3
contactfirstname	First name				13%	19% validation Surname 91% validation First name +3
phone	[North America] Phone Number				100%	100% xx_Uniqueness_Cust... 100% xx_Uniqueness_Cust...
creditlimit				Enabled	37%	37% xx_Average_creditlimit

Conclusion

We have come to the end of this workshop!

We have created two Aggregation rules using Group by and Aggregate functions. If applicable, you can practice creating more complex Aggregation rules using both Group by and Aggregate functions in the same logic.