



aSIST

Analytical reports module / Catalogue variables

FINGO Team

version: 5.59.0.0 04.2024

All rights reserved.

Document should be kept and reproduced without any limitations only as the whole document.

Any part of this document, either as whole or as in fragments, should not be reproduced nor processed electronically, photographically, mechanically and in any other way and it should not be stored on any database without FINGO sp. z o.o. written agreement.

Table of Contents

1	Introduction.....	5
2	Analytical reports manager	5
2.1	“Report list” mode	6
2.1.1	Creating a new analytical report.....	6
2.1.2	Generating an analytical report.....	9
2.1.3	Report statuses.....	10
2.2	“Documents” mode	10
3	Catalogue variable manager	11
4	Defining expressions	12
4.1	Arithmetic functions.....	13
4.2	Text functions.....	14
4.3	Logical functions.....	14
4.4	Functions returning information based on a fact.....	15
4.5	Condition functions	16
4.6	Aggregating functions	16

Table of Images

Image 1. Selecting the “Analytical reports” module	6
Image 2. “Analytical reports” module – “Report list” mode.....	6
Image 3. Displaying the generated analytical report – “Documents” mode.....	11
Image 4. Selecting the “Catalogue variable” mode	11
Image 5. “Catalogue variable manager”	12

1 Introduction

The “Analytical reports” module is an additional tool available in aSIS^t to facilitate analysis of data reported with aSIS^t.

With these available functions it is easy to define and create any analytical reports with data from the aSIS^t base.

The created reports can include:

- real data gathered by the bank and/or
- aggregated data, compliant with the selected aggregate.


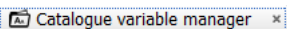
Data aggregation can be performed at different stages of report preparation:

- while selecting reporting periods,
- while selecting banks (if there are data of several entities used within the application),
- while selecting facts to analyse.

The defined reports can be:

- generated, edited and removed by the report creator and a person with administrator rights;
- generated by other users.

All functions facilitating easy defining and generating of created analytical reports were grouped on two module tabs:

- Analytical reports: 
- Catalogue variables: 

and can be selected at the following levels:

- menu screens:
 - report menu
 - view menu

or

- functional icons.

2 Analytical reports manager


To activate the “**Analytical reports manager**”, select:


-  – located on the module tab bar (see Image 1)

and next:




Image 1. Selecting the “Analytical reports” module

A new module tab  **Analytical reports manager** will each time:

- open in the **“Report list” mode**  (see Chapter [2.1 “Report list” mode](#) Image 2),
- display a **list of previously created reports**,
- enable defining of new and managing of previously created analytical reports.

At any moment it is possible to:

- move to the **“Documents” mode**  (see Chapter [2.2 “Documents” mode](#) Image 3),
- view a **previously generated report** (if such report was generated after the application launch),
- make changes to the layout of the displayed data.

2.1 “Report list” mode

Each time after activation of the “Analytical reports manager” mode a screen with available, previously created analytical reports will be displayed, as on the Image 2.

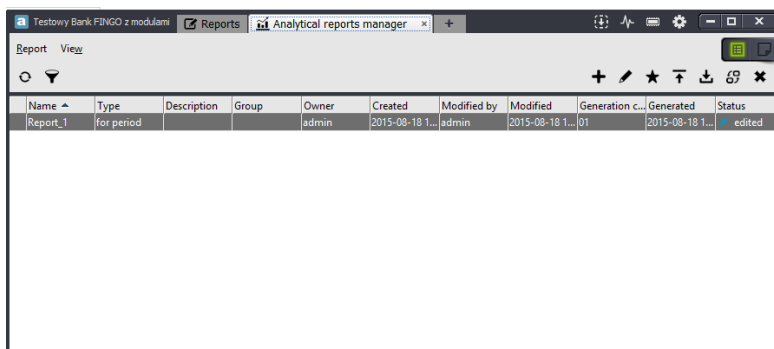


Image 2. “Analytical reports” module – “Report list” mode

2.1.1 Creating a new analytical report

To create a new analytical report, use the **“New report”** function by selecting:

- “Report” menu  New
- or
- icon 

It is necessary to:

- provide a unique name for the report to be created (any characters permitted, max 200),
- choose the:
 - report type,
 - period(s) to which the report will apply,
 - banks for which the data will be presented,
 - taxonomy to be used in the report,
 - variables used in the report,
- approve the selected settings.

When selecting periods, the user can choose between:

- a parameterized period – defined only at the moment of report generation,
- a period applying to individual:
 - months,
 - quarters (with additional specification of the **aggregation** type),
 - years (with additional specification of the **aggregation** type),
- a relative period in relation to the current report (the user may choose from the range: “from **current** to **current-24**”),
- all beginning with...,
- all/selected periods (with additional specification of the **aggregation** type).

When selecting entities, the user can choose between:

- a parameterized bank – defined only at the moment of report generation,
- any bank from the displayed list,
- all entities (with additional specification of the **aggregation** type) or
- a group of banks (with additional specification of the **aggregation** type).

When selecting a taxonomy, the user can choose between:

- one taxonomy (when items within one report are subject to analysis) or
- various taxonomies linked with the report to be created.

When selecting variables, the following functions can be used:

- **add fact** – requires the user to specify individual variables
 - each selected fact is automatically assigned a code (alias) identifying the selected variable in the following form: **FX** (where x is the consecutive fact number),
- **add expression** – using previously selected facts and/or aggregates and permitted arithmetic operations (see Chapter [4 Defining expressions](#))
 - each selected expression is automatically assigned a code (alias) identifying the selected expression in the following form: **VX** (where x is the consecutive expression number),
- **add aggregate** – where previously selected facts, created expressions or selected catalogue variables are subject to additional aggregation by selecting an appropriate **aggregation** type
 - each aggregate is automatically assigned a code (alias) identifying the selected aggregate in the following form: **: AX** (where x is the consecutive aggregate number),
- **add from catalogue** – by specifying a previously defined variable in **“Catalogue variables”** (see Chapter [3. Catalogue variable manager](#)),
- **add parameter** – i.e. specifying the parameterized variables for which the value is provided in the moment of analytical report generation
 - each parameter is automatically assigned a code (alias) identifying the selected parameter, in the following form: **PCX** (where x is the consecutive parameter number),
- **add aggregated dimensions** – by specifying variables which are the typed dimensions and specifying for them the data aggregation mode
 - each aggregated dimension is automatically assigned a code (alias) identifying the selected dimension, in the following form: **ATDX** (where x is the consecutive aggregated dimension number).

It is possible to select all functions simultaneously for one analytical report.

If created report has got some empty (null) values, then in the column: **“Empty value treated as 0”**

- after selecting feature by some position, application will treat this value as zero. Values of this type will be included to the report,
- if this feature will be not selected then in the case of lacking some data for given value, the value will be not presented in the created report.

If any variable selected while defining the report should not be displayed on the generated report then:

- in the **“Reported”** field – deselect this feature,
- it means it was included in the variables list to be a component of the created expression or aggregate.

Such situation can occur when the user creates various reports, especially when only one variable is permitted to be reported in:


- “Cross-section reports”,
- “Comparative reports for periods”,
- “Comparative reports for banks”.

Selected variables are subject to automatic accuracy validation and results are displayed in:

- the **“Validation status”** column in the displayed variables list,
- separately for each variable.

In the process of defining an analytical report it is also possible to:

- sort variables to be displayed upon report generation,
- make changes to header settings of the report to be defined.

With the function key:  **Layout setting**, the user can make changes to:

- the report layout (selecting a different layout than the one imposed by the system),
- header label types.

Selecting “Save and generate” or “Generate” will result in:

- automatic transfer to the “Documents” mode – see Chapter [2.2 “Documents” mode](#),
- displaying data compliant with the rules defined at the stage of report creation,
- displaying a screen as on Image 3.

If while defining an analytical report the user chose a parameterized period or a parameterized bank, then:

- the **report** as shown in Image 3 **will not be generated**, but
- the screen will appear with a prompt to select the period or bank, respectively, for which the report is to be generated.


A newly defined report always has the **“edited”** status, which means it still can be modified.

If the report defining process is completed and the user wishes to make the report available also to other users, it is necessary to change the report status to **“active”** (see Chapter [2.1.3 Report statuses](#)).

2.1.2 Generating an analytical report


Defined analytical reports can be generated at any time with the **“Generate”** function.

After highlighting the required report on the list, the user should select:

- "Report" menu  Generate

or

- icon 

In case of reports for which specific banks and periods to which the report applies were selected with the  function during the process of report defining:

- the system automatically displays the required data.

In case of reports for which a parameterized bank or period was selected during the process of report defining:

- this function has a longer cycle, because it is necessary to specify precisely the banks or periods to which the prepared report should apply.

2.1.3 Report statuses

Depending on the stage of preparation and availability during which the analytical report is defined, three different report statuses can be named:

- **"edited"** – the report can still be modified and for this reason **it is still not available to other users with rights lower than administrator rights,**
- **"active"** – report defining was completed and the **report was made available to other users,**
- **"inactive"** – the report is not available to other users.


To change the status of the defined report, the user should choose the **"Change status"** function by selecting:

- "Report" menu  Change status

or

- icon 

2.2 "Documents" mode

"Documents" mode  (see Image 3) is a mode in which the application **displays results of the generated analytical report.**

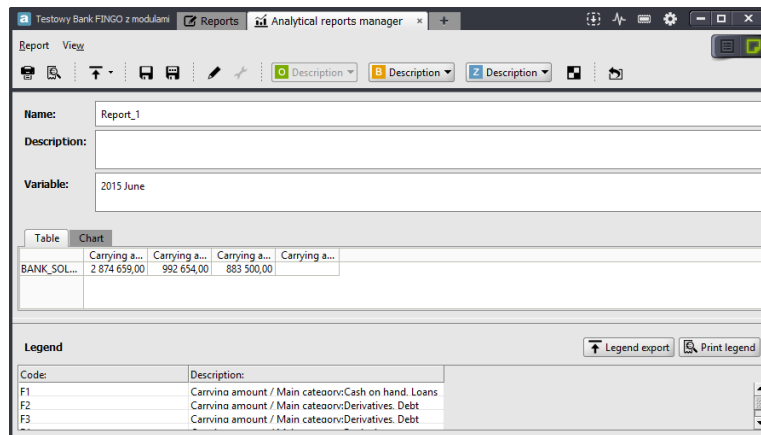




Image 3. Displaying the generated analytical report – “Documents” mode

Transition to “Documents” mode can be made after:

- analytical report generation – then results of such generated report are displayed,
- a change of the “view switch”  to ; then the data displayed are data from the previously generated report (if, after the application launch, no report was generated then the screen is empty)

Each user, after report generation, can make changes to the layout of the generated data, using available functions.

3 Catalogue variable manager

To define variables which will be used multiple times in various analytical reports, the user should activate the **“Catalogue variable manager”** mode by selecting:

-  – located on the module tab bar (see Image 4)

and then:

- 



Image 4. Selecting the “Catalogue variable” mode

Then a new module tab is created  Catalogue variable manager as on Image 5:

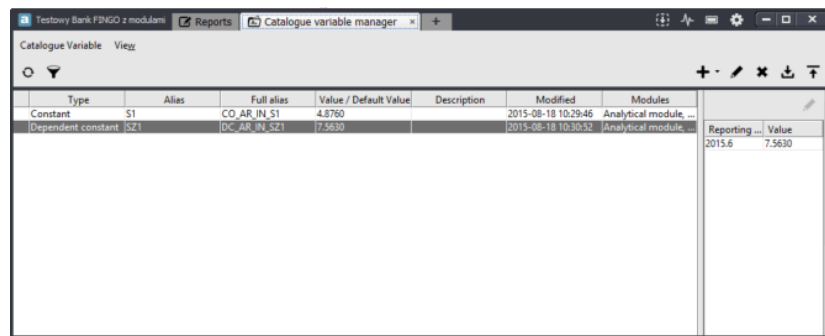


Image 5. "Catalogue variable manager"

In order to create a new catalogue variable, the user should use the **"Add a catalogue variable"** function by selecting:

- **"Catalogue variable" menu** Add 

or icon

- 

When selecting catalogue variables, the following functions can be used:

- **Add expression** – allowing the user to create an expression based on any taxonomy, all facts from such taxonomy and available arithmetic operations (see [Chapter 4 Defining expressions](#))
 - the system automatically assigns an **identification code** to created expressions in the following form: **Wx** (where x is the consecutive number of the created expression)
- **Add constant** – allowing the user to define a constant of the following type: arithmetic, data type, logical, string
 - the system automatically assigns an **identification code** to catalogue constants in the following form: **Sx** (where x is the consecutive number of the defined constant)
- **Add a dependent constant** – allows the user to define a constant of the following type: arithmetic, data type, logical, string, the value of which is automatically linked with all reporting periods created in the aSIST base (those already created and those to be created). It is possible to modify these variables for all as well for selected period.
 - the system automatically assigns an **identification code** to all such variables in the following form: **SZx** (where x is the consecutive number of the defined constant)

4 Defining expressions

Created expressions are based in algebra **(A,D)**, where:

- **A** is the sum of the set of facts in a taxonomy, the type of which is a numeric value, i.e. `monetary` or `pureItem`, and catalog data and formulated aggregates located on the list of defined variables
- **D** is a set of operators: `{+,-,*,/}`
- joining with brackets is also possible `()`.

When creating an expression the user can use the following functions: arithmetic, text, logical, returning information based on a fact, conditional, aggregating.

4.1 Arithmetic functions

Arithmetic functions – can be used freely in arithmetic expressions:

- **abs (<expression>)** – the function returns an absolute value from the specified argument. Individual facts as well as more complex arithmetic expressions can serve as function arguments;
- **step (<expression>, <expression>, <expression>, <expression>)** – the function verifies whether the value of the first argument is within the range specified by the second and the third argument. If the value is greater than or equal to the beginning range value and less than the end range value then the value of the fourth argument is returned, in other cases the function takes the zero value;
- **compare (<text_expression>, <text_expression>)** – the function used to compare two character strings. Any text expression can be the function argument. The value returned by the function is equal to 0 only when the arguments are equal. The application returns the value greater than 0 only when the first argument of the function follows the second argument in the lexicographical order. The function returns a value lesser than 0 only when the second argument follows the first argument in the lexicographical order;
- **indexOf (<text_expression>, <text_expression> [, <expression>])** – the function searching for the first occurrence of the text provided as the second argument in the text provided as the first argument. Optionally, a third argument is assumed in the form of a character after which the search is performed (0 means a value before the first character – a default value). The function returns a number of the character after which the searched text begins. In case the text provided does not contain the searched text -1 value is returned;
- **lastIndexOf (<text_expression>, <text_expression> [, <expression>])** – the function searching for the last occurrence of the text provided as the second argument in the text provided as the first argument. Optionally, a third argument is assumed in the form of a character after which the search is performed (the default search begins from the end of the text). The function returns a number of the character after which the searched text begins. In case the text provided does not contain the searched text -1 value is returned;
- **length (<text_expression>)** – the function returns the length of the text provided. The length of the empty text is 0.

4.2 Text functions

Text functions – can be used as arguments in other functions:

- **toString (<expression>)** – the function enables conversion of the numeric expression (in most cases it is a fact) to text. Conversion of the number is performed to the form with the scale equal to 2 and a comma as a decimal separator;
- **concat (<text_expression>, <text_expression>)** – the function joins two texts into one,
- **toUpperCase (<text_expression>)** – the function returns the text as the one provided, however all characters are in upper case;
- **toLowerCase (<text_expression>)** – the function returns the text as the one provided, however all characters are in lower case;
- **trim (<text_expression>)** – the function trims empty (non-printable) characters from the beginning and the end,
- **reverse (<text_expression>)** – the function returns the text which is a reversal of the text provided as an argument,
- **substring (<text_expression>, <expression> [, <expression>])** – the function returns the text created by trimming the text provided as a first argument to the part specified by created arguments. The first argument specifies the number of the character after which the trimming is performed. If it is equal to 0 the result starts with the beginning of the trimmed text. The second argument specifies the number of the character before which the trimming is performed. The second argument is optional and if it is not provided, the trimmed text ends with the end of the base text. If provided arguments have a negative value or exceed the text length, an error appears during calculation;
- **replaceAll (<text_expression>, <text_expression>, <text_expression>)** – the function which matches the template provided in the second argument to the text provided in the first argument. All matching places are replaced by a third argument in the returned text.

4.3 Logical functions

Logical functions – can be freely used in logical conditions:

- **equals (<text_expression>, <text_expression>)** – the function comparing two texts. The function returns truth only if the first argument is lexically of the same length as the second one;
- **equalsIgnoreCase (<text_expression>, <text_expression>)** – the function comparing two texts. It returns truth only if the first argument is lexically of the same length as the second one – irrespective of the case;

- **matchesRegex (<text_expression>, <text_expression>)** – the function which matches the template provided in the second argument to the text provided in the first argument. It returns truth only when the template matches the text.

4.4 Functions returning information based on a fact

Functions returning information based on the provided fact can be used, depending on the returned type:

- **isNotNull (<argument>)** – the function is defined as in the obligatoriness rule. It returns a logical value,
- **isNull (<argument>)** – the function is defined as in the prohibited value rule. It returns a logical value,
- **getReportYear (<argument>)** – the function returns a number corresponding to the year value in the reporting period for the provided fact,
- **getReportMonth (<argument>)** – the function returns a number corresponding to the month value in the reporting period for the provided fact,
- **getReportDay (<argument>)** – the function returns a number corresponding to the day value in the reporting period for the provided fact,
- **getMeasure (<argument>)** – the function returns text corresponding to the measure of the provided fact, expressed on the technical label,
- **getDimensionMember (<argument>, <text_expression>)** – the function returns text corresponding to the value of the dimension provided in the second argument for the fact provided in the first argument. The result is expressed on the technical label. In case a non-existent dimension is provided, empty text is returned. The dimension is provided in the form of a technical label;
- **getBankNumber(<argument>)** – the function returns the bank number,
- **getBankName(<argument>)** – the function returns the bank name,
- **getBankShortName(<argument>)** – the function returns the bank short name,
- **getBankBusinessNumber(<argument>)** – the function returns the bank REGON,
- **getBankCode(<argument>)** – the function returns the bank code,
- **prev (<argument> [, <value> [, <unit>]])** – the function allows for the use of facts from previous periods in the rule. It has two optional parameters. The first one is the **<value>** argument specifying a number indicating the amount of time units the user wished to go back. The default value of this parameter is '1'. The other argument is **<unit>**. The possible values of this attribute are as follows: 'Y', 'Q', 'M' (standing for year, quarter, month, respectively). The default value of the parameter is: 'M'.

4.5 Condition functions

Condition functions:

- can be used depending on the returned type,
- search for values by evaluating one by one the logical conditions (which are odd number arguments),
- take the value equal to the value of the expression after the first condition that is met. The remaining expressions are not evaluated;
- in case none of the conditions are true, the function takes the value of the expression provided as the last argument;
- need to have an odd number of arguments, greater than or equal to three.

Condition functions:

- **caseA([<log expression>, <expression>]+, <expression>)** – the function selecting an arithmetical expression,
- **caseS([<log expression>, <text_expression>]+, <text_expression>)** – the function selecting a text expression,
- **caseL([<log expression>, <log expression>]+, <log expression>)** – the function selecting a logical expression.

4.6 Aggregating functions

Aggregating functions:

- can be used in any arithmetical expression,
- collate aggregates from list arguments or typed dimension values.

The expression provided as the **first argument** is evaluated for each list position or dimension value and then an aggregating function is used.

The second argument is an optional condition that has to be met by list positions or dimension values in order to be an aggregation component. In case of no condition, the function assumes that it is always true, i.e. all positions are part of the aggregate.

Aggregating functions for the typed dimensions assume that:

- there is exactly one argument containing a typed dimension (it can be repeated and occur in any part of the expression or the condition),
- in case there are no arguments or there is more than one argument containing a typed dimension, a parse error occurs.

Aggregating functions for lists:

- allow for multiple list arguments,
- all such arguments have to be from one list,
- must have at least one list argument (in any place in the expression or the condition),
- in case there are no list arguments or the arguments come from more than one list, a parse error occurs.

Using arguments with typed dimensions and list arguments in one function, results in a parse error.

Aggregating functions:

- **sum(<expression>[, <log expression>])** – sum. For reasons of compatibility with the previous versions, the sum after one fact without an additional argument is saved as an address with the '+' sign as a default (it is only a text abbreviation);
- **avg(<expression>[, <log expression>])** – average,
- **min(<expression>[, <log expression>])** – minimum,
- **max(<expression>[, <log expression>])** – maximum,
- **median(<expression>[, <log expression>])** – median
- **standard_deviation(<expression>[, <log expression>])** – standard deviation,
- **variance(<expression>[, <log expression>])** – variance.