Advanced Formulas

This page is for storing various SQL formulas and regular expressions for potential re-use in Alma Analytics. The SQL code will state whether it should be used as columns or as filters.

- **Advanced Formula Definitions (Read First)**
- **Budget String Filters**
  - TUB (filters)
  - ORG (filters)
  - OBJECT (none)
  - FUND (filters)
  - ACTIVITY (filters)
  - SUB_ACTIVITY (filters)
  - ROOT (filters)
- **Time Period Manipulation**
  - Budget Year (filters)
  - Current Fiscal Year (filters)
  - Last Three Fiscal Years (filters)
  - Previous Fiscal Year (filters)
  - Prior/Future XXX Time Period Formula and Between XXX Time Periods (filters)
  - Difference Between Dates (column formula)
- **EVALUATE and EVALUATE_AGGR Database Functions**
  - EVALUATE
  - EVALUATE_AGGR
  - LISTAGG
  - REGEXP_COUNT
  - REGEXP_INSTR
  - REGEXP_REPLACE
  - REGEXP_SUBSTR
- **Miscellaneous SQL**
  - Modifying a Portion of a Column with a Case Statement (column formula)
  - Listing Repeated Fields in Different Columns (column formula)
  - Adding One or More Filters to a Column (column formula)
  - Counting (column formula)
  - Count Repeated Fields in a Concatenated Field (column formula)
  - Row Numbers (column formula)
  - Concatenate Two Fields with a Separator (column formula)
  - Percentages (column formula)
  - Network Number [035, OCLC number] for only OCLC numbers (column formula)
  - Extracting a subfield (column formula)
  - Comments in SQL
  - MAX / MIN / Count Distinct "Subtotal" Values in a List Row
- **Additional Formulas/SQL Queries from Ex Libris**
- **Containers for Additional Formulas/SQL Queries**
  - Placeholder for next formula
  - Placeholder for next formula
  - Placeholder for next formula

### Advanced Formula Definitions (Read First)

**Column Formula:** If a column formula is altered then the content produced in the column will be different. It can be altered with a filter or with other elements. When altered by a filter command, then the column–and its content–can be filtered by criteria from a different column. That way a column can contain a filter from a different column.

The content of a column can also be altered in the column formula. For example "Unknown" could be changed to "Non-LC Call number." A filter applied within column formula is a filter from another column. These alter the sets that the column produces.

**Filter:** Whatever the column generates, a filter chooses a part of that content to show. It does not alter the content of the column formula, it just determines what will be shown from based on column’s properties.

### Budget String Filters

See also: Budget String Components

#### TUB (filters)

```sql
SUBSTRING("Fund Ledger"."Fund Ledger Code" FROM 1 FOR 3)
```
### ORG (filters)

```
SUBSTRING("Fund Ledger"."Fund Ledger Code" FROM 5 FOR 5)
```

### OBJECT (none)

The OBJECT designator needs no SQL - use the Reporting Code in Analytics.

### FUND (filters)

```
SUBSTRING("Fund Ledger"."Fund Ledger Code" FROM 12 FOR 6)
```

### ACTIVITY (filters)

```
SUBSTRING("Fund Ledger"."Fund Ledger Code" FROM 19 FOR 6)
```

### SUB_ACTIVITY (filters)

```
SUBSTRING("Fund Ledger"."Fund Ledger Code" FROM 26 FOR 4)
```

### ROOT (filters)

```
SUBSTRING("Fund Ledger"."Fund Ledger Code" FROM 31 FOR 6)
```

### Time Period Manipulation

#### Budget Year (filters)

```
CAST(RIGHT("Fiscal Period"."Fiscal Period Description", 4) AS integer)
```

#### Current Fiscal Year (filters)

**One Option**

```
"Fiscal Period"."Fiscal Period Start Date" >= (YEAR(CURRENT_DATE)-1) * 10000 + MONTH(CURRENT_DATE) * 100 + DAY(CURRENT_DATE))
```

**Another Option**

```
RANK(CAST(RIGHT("Fiscal Period"."Fiscal Period Description", 4) AS integer),1) = 1
```

#### Last Three Fiscal Years (filters)

```
"Fiscal Period"."Fiscal Period Start Date" BETWEEN 
{}(YEAR(CURRENT_DATE) -4) * 10000 + MONTH(CURRENT_DATE) * 100 + DAY(CURRENT_DATE)) 
AND 
{}(YEAR(CURRENT_DATE) -1) * 10000 + MONTH(CURRENT_DATE) * 100 + DAY(CURRENT_DATE))
```
Previous Fiscal Year (filters)

"Fiscal Period"."Fiscal Period Start Date"
BETWEEN
((YEAR(CURRENT_DATE) - 2) * 10000 + MONTH(CURRENT_DATE) * 100 + DAY(CURRENT_DATE))
AND
((YEAR(CURRENT_DATE) - 1) * 10000 + MONTH(CURRENT_DATE) * 100 + DAY(CURRENT_DATE))

Prior/Future XXX Time Period Formula and Between XXX Time Periods (filters)

Basic formula

[Insert Alma Column SQL] = TIMESTAMPADD(SQL_TSI_DAY,-1,CURRENT_DATE)

Basic formula with the available variables for unit of time in brackets

[Insert Alma Column SQL] = TIMESTAMPADD(SQL_TSI_[DAY,MONTH,WEEK,YEAR],-1,CURRENT_DATE)

Example: the previous 10 years

[Insert Alma Column SQL] = TIMESTAMPADD(SQL_TSI_YEAR, -10, CURRENT_DATE)

Example: Due Dates is in the past 10 years

"Due Date"."Due Date" = TIMESTAMPADD(SQL_TSI_YEAR, -10, CURRENT_DATE)

Invoice line dates falling within the next three months

MONTH("Invoice Line"."Invoice-Date") = MONTH(TIMESTAMPADD(SQL_TSI_MONTH,3,CURRENT_DATE))

Example: When used with the between operator

"Due Date"."Due Date"
BETWEEN
TIMESTAMPADD(SQL_TSI_WEEK, -14, CURRENT_DATE)
AND
TIMESTAMPADD(SQL_TSI_DAY, -1, CURRENT_DATE)

Previous month

BETWEEN TIMESTAMPADD(SQL_TSI_MONTH, -1, TIMESTAMPADD(SQL_TSI_DAY, DAYOFMONTH(CURRENT_DATE) * -(1) + 1, CURRENT_DATE)) AND TIMESTAMPADD(SQL_TSI_DAY, -(1), TIMESTAMPADD(SQL_TSI_DAY, DAYOFMONTH(CURRENT_DATE) * -(1) + 1, CURRENT_DATE))

Difference Between Dates (column formula)

Basic formula with interval and timestamp

TIMESTAMPDIFF([interval], [Insert Alma Column SQL], [timestamp])
Example: The number of days between the creation date and a specific date

TIMESTAMPDIFF(SQL_TSI_DAY, "Creation Date"."Creation Date", TIMESTAMP '2023-06-30 23:59:59')

Example: The number of days between the creation date and today

TIMESTAMPDIFF(SQL_TSI_DAY, "Creation Date"."Creation Date", CURRENT_DATE)

EVALUATE and EVALUATE_AGGR Database Functions

EVALUATE and EVALUATE_AGGR functions allow database functions not otherwise available in Alma Analytics (OBIEE) to be used. These database functions allow advanced calculations and formatting and regular expression searching.

**EVALUATE**

**Basic formula**

EVALUATE('database function(%1,%2 ... %N)' [AS datatype], "Alma Column SQL", 'regular expression' [, %N parameters separated by commas])

Required parameters:

- %1 is the Alma column SQL (for example, "Holding Details"."852 MARC").
- %2 is the regular expression.

Other common parameters (parameter number varies depending on database function and these parameters are often optional):

- position: This is the starting position. The default is 1 (starting with the first character).
- occurrence: The default is 1 (first occurrence).
- match_parameter: This is the parameter for matching. NULL means no changes are made to the default. Other options are 'i' (case-insensitive matching), 'c' (case-sensitive matching), 'n' (allow '.' to match the newline character), 'm' (treat string as multiple lines), and 'x' (ignore whitespace characters).
- subexpr: This is for specifying capture groups within a regular expression. The default is 1 (first subexpression), and the value must be an integer from 0 to 9.

**EVALUATE_AGGR**

Enables aggregate functions with a GROUP BY clause

**LISTAGG**

Orders data in a group and concatenates values from a column

**Basic formula**

EVALUATE_AGGR('listagg(%1,%2) WITHIN GROUP (ORDER BY %3)', [Insert Alma Column SQL], 'delimiter', [Insert Alma Column SQL])

- %1 measure_expr: Alma column SQL to be concatenated (for example, "Physical Item Details"."Barcode")
- %2 delimiter_expr: Delimiter to separate values in the concatenated list (for example, ' | ' or ', ')
- %3 order_by_clause: Alma column SQL for ordering the list

The default order is ascending. For descending order, use "(ORDER BY %3 DESC)".

**REGEXP_COUNT**

Counts the number of times a pattern occurs
Basic formula

EVALUATE('regexp_count(%1,%2,%3,%4)', [Insert Alma Column SQL], 'regular expression', 1, 'c')

Optional parameters:
- %3 position
- %4 match_parameter

REEXP_INSTR
An extension of the INSTR function

Basic formula

EVALUATE('regexp_instr(%1,%2,%3,%4,%5,%6)', [Insert Alma Column SQL], 'regular expression', 1, 1, 0, 'c')

Optional parameters:
- %3 position
- %4 occurrence
- %5 return_option: This can be either 0 or 1, with 0 as the default. 0 returns the character position of the occurrence, and 1 returns the character position following the occurrence.
- %6 match_parameter

The Ex Libris Tech Blog post Alma Analytics Regular Expression Examples shows how this function can be used for filtering. The examples in that post are formatted with only the Alma column as a parameter. Either format is valid:

Formatting comparison

EVALUATE('regexp_instr(%1, ''^DISC+'', 1, 0, ''c'')', "Physical Item Details"."Public Note")
EVALUATE('regexp_instr(%1,%2,%3,%4,%5,%6)', "Physical Item Details"."Public Note", '^DISC+', 1, 1, 0, 'c')

REEXP_REPLACE
An extension of the REPLACE function

Basic formula

EVALUATE('regexp_replace(%1,%2,%3,%4,%5,%6)', [Insert Alma Column SQL], 'regular expression', 'replace string', 1, 0, 'c')

- %3 replace_string: The string or regular expression to replace the value found by the regular expression in %2.

Optional parameters:
- %4 position
- %5 occurrence: The occurrence to be replaced. If 0, all occurrences will be replaced.
- %6 match_parameter

REEXP_SUBSTR
An extension of the SUBSTR function

Basic formula

EVALUATE('regexp_substr(%1,%2,%3,%4,%5,%6)', [Insert Alma Column SQL], 'regular expression', 1, 1, NULL, 1)

Optional parameters:
- %3 position
- %4 occurrence
Miscellaneous SQL

Modifying a Portion of a Column with a Case Statement (column formula)

<table>
<thead>
<tr>
<th>Basic formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASE When [Insert Alma Column SQL] = ['Result you want replaced'] Then ['New result'] Else [Repeat Insert Alma Column SQL] END</td>
</tr>
</tbody>
</table>

Example with LC Classification Code

<table>
<thead>
<tr>
<th>CASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHEN &quot;LC Classifications&quot;.&quot;Classification Code&quot; = 'Unknown'</td>
</tr>
<tr>
<td>THEN</td>
</tr>
<tr>
<td>'Unclassed or mis-coded'</td>
</tr>
<tr>
<td>ELSE &quot;LC Classifications&quot;.&quot;Classification Code&quot;</td>
</tr>
<tr>
<td>END</td>
</tr>
</tbody>
</table>

Listing Repeated Fields in Different Columns (column formula)

<table>
<thead>
<tr>
<th>Basic formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVALUATE('regexp_substr(%1,%2,%3,%4)', [Insert Alma Column SQL], '[^; ]+', 1, 1)</td>
</tr>
</tbody>
</table>

The regular expression locates all characters up to the first occurrence of a semicolon followed by a space. The delimiter for concatenated values in Alma is “;” (semicolon followed by a space). The semicolon needs to be escaped for the regular expression.

This formula retrieves the first ISBN value.

Example with ISBN

| EVALUATE('regexp_substr(%1,%2,%3,%4)', "Bibliographic Details"."ISBN", '[^\; \]+', 1, 1) |

Repeat the formula and increment %4 to create columns for multiple ISBN values.

Example: ISBN 2 column

| EVALUATE('regexp_substr(%1,%2,%3,%4)', "Bibliographic Details"."ISBN", '[^\; \]+', 1, 2) |

The results are like this:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0374107017</td>
<td>0374511713</td>
<td>9780374107017</td>
<td>9780374511715</td>
</tr>
<tr>
<td>0691072086</td>
<td>0691019843</td>
<td>9780691072081</td>
<td>9780691019840</td>
</tr>
<tr>
<td>0151937656</td>
<td>0156935805</td>
<td>9780151937653</td>
<td>9780156935807</td>
</tr>
<tr>
<td>0698106695</td>
<td>9780698106697</td>
<td>9780698106697</td>
<td>9780698106697</td>
</tr>
<tr>
<td>0810807882</td>
<td>9780810807884</td>
<td>9780810807884</td>
<td>9780810807884</td>
</tr>
</tbody>
</table>

Adding One or More Filters to a Column (column formula)
### Basic formula for 1 filter

```
FILTER([Insert Alma Column SQL to be filtered] USING ([Insert Alma Column SQL for filter] = [Content of filter]))
```

#### Example

```
FILTER
("Fund Transactions"."Sum"
USING
("Fund Transactions Details"."Transaction Item Type" = 'ALLOCATION')
```

### Formula for two filters

```
FILTER(FILTER(SUM([Insert Alma Column SQL]) USING ([Insert Alma Column SQL Filter 1] = '[Content of filter 1]')) USING ([Insert Alma Column SQL Filter 2] = [Content of filter 2]))
```

#### Example using “Fund Transactions"."Transaction Amount" and "Transaction Date"."Transaction Date Fiscal Year"

```
FILTER(FILTER(SUM
("Fund Transactions"."Transaction Amount")
USING
("Fund Transaction Details"."Transaction Item Type" = 'ALLOCATION'))
USING
("Transaction Date"."Transaction Date Fiscal Year" = 2015))
```

### Counting (column formula)

#### Basic formula

```
COUNT([Insert Alma Column SQL])
```

#### Example: count all occurrences

```
COUNT("Bibliographic Details"."MMS Id")
```

#### Example: count each occurrence only once

```
COUNT(DISTINCT "Holding Details"."Holding Id")
```

### Count Repeated Fields in a Concatenated Field (column formula)

#### Basic formula

```
IFNULL(CAST(EVALUATE('REGEXP_COUNT(%1,%2)', [Insert Column SQL], '\;') AS INT), -1) + 1
```

REGEXP_COUNT returns the number of times that a semicolon occurs. IFNULL replaces the returned NULL value of zero (0) with minus one (-1) instead of zero (0). One (1) is added to count the field that is not followed by a semicolon.

This formula counts the number of OCNs in the OCLC Control Number (035a) field.
Example: Counting OCNs in OCLC Control Number (035a) field

```
IFNULL(CAST(EVALUATE('REGEXP_COUNT(%1,%2)',"Physical Items"."Bibliographic Details"."OCLC Control Number (035a)", '\;') AS INT), -1) + 1
```

Row Numbers (column formula)

This code will provide a column that will count the number of rows in an analysis.

Running number for all lines

```
RCOUNT(1)
```

This formula will give you the total number of rows. Use this in the header/title area.

Total number of lines

```
MAX(RCOUNT(1))
```

This formula will give you the total number of MMS ID lines including any duplicates.

Total number of MMS IDs

```
MAX(RCOUNT("Bibliographic Details"."MMS Id"))
```

Concatenate Two Fields with a Separator (column formula)

Basic formula

```
[Insert Alma Column SQL 1st position] \|| '[Separator]' \|| [Insert Alma Column SQL 2nd position]
```

Example with last and first names separated by a comma

```
"User Details"."Last Name" || ',', || "User Details"."First Name"
```

Another way to concatenate

```
CONCAT(CONCAT([Insert Alma Column SQL 1st position], ',', [Insert Alma Column SQL 2nd position])
```

Example

```
CONCAT(CONCAT("User Details"."First Name", ','), "User Details"."Last Name")
```

The result of the above looks like this:
You can use this function to solve the problem with the way Excel treats long numbers, such as the MSSID and the Barcode. One way to ensure that MMSIDs and Barcodes (or any other long numbers) translate properly when exported to Excel, is to prepend a non-numeric character or string to the numeric value.

**Example: Prepend an underscore to MSSID**

```sql
'_' || "Bibliographic Details"."MMS Id"
```

Result: `_990107328710203941`

**Example: Prepend an underscore to Barcode**

```sql
'_' || "Physical Item Details"."Barcode"
```

Result: `_32044100361559`

This will convert the columns in Excel to text and the numbers will not be rounded off by Excel. When you open the file in Excel, first remove the added character using the Excel function Find and Replace, and then you have full numbers. Note: do not use mathematical symbols.

**Percentages (column formula)**

**Basic formula**

```
[Insert Alma Column SQL 1] / [Insert Alma Column SQL 2] * 100
```

**Example**

```
"Physical Item Details"."Num of Loans - not sum" / "Physical Item Details"."Num of Items" * 100
```

**Converting to Day of the Week (e.g., Monday, Tuesday, etc.) (column formula)**

```sql
CASE
WHEN DAYOFWEEK("Loan Date"."Loan Date") = 1 THEN 'Sun'
WHEN DAYOFWEEK("Loan Date"."Loan Date") = 2 THEN 'Mon'
WHEN DAYOFWEEK("Loan Date"."Loan Date") = 3 THEN 'Tue'
WHEN DAYOFWEEK("Loan Date"."Loan Date") = 4 THEN 'Wed'
WHEN DAYOFWEEK("Loan Date"."Loan Date") = 5 THEN 'Thu'
WHEN DAYOFWEEK("Loan Date"."Loan Date") = 6 THEN 'Fri'
WHEN DAYOFWEEK("Loan Date"."Loan Date") = 7 THEN 'Sat'
ELSE '' END
```

**Network Number [035, OCLC number] for only OCLC numbers (column formula)**

This code will provide a column that will show only the OCLC in the 035, the Network Number.

This code will retrieve the first OCLC number:
If you suspect that there are more than one OCLC number in the 035, and there often are, then you modify one number for each OCLC number you may have.

Extracting a subfield (column formula)

**Basic formula**

EVALUATE('regexp_substr(%1,%2,%3,%4)', [Insert Alma Column SQL], '\$[subfield code][\^\$]+', 1, 1)

The regular expression searches for the subfield delimiter and code ($f, for example) followed by one or more characters that are not a dollar sign ($), meaning from the start of the subfield until the end of that subfield.

**Example: Extracting first subfield f from the Holding Local Param 03 (920 reporting field)**

EVALUATE('regexp_substr(%1,%2,%3,%4)', "Holding Details"."Holding Local Param 03", '\$f[^\$]+', 1, 1)

To extract the subfield text without the subfield delimiter and code ("abc123" instead of "$f abc123"), edit the regular expression to include a capture group using parentheses and add match and subexpression parameters to the regexp_substr function. The "i" (in double single quotes) means case-insensitive matching, and the third 1 specifies the first capture group (the "([\^\$]+)" following "$f " in the regular expression).

**Example: First subfield f from the Holding Local Param 03 (920 reporting field) without subfield tag**

EVALUATE('regexp_substr(%1,%2,%3,%4,%5,%6)', "Holding Details"."Holding Local Param 03", '\$f {{[^\$]+}', 1, 1, 'i', 1)

Use trim functions to remove leading or trailing white-space and trailing punctuation ("abc123" instead of " abc123;").

**Example: First subfield f from the Holding Local Param 03 (920 reporting field) without subfield tag**

TRIM(TRAILING ';' FROM TRIM(BOTH FROM EVALUATE('regexp_substr(%1,%2,%3,%4,%5,%6)', "Holding Details"."Holding Local Param 03", '\$f {{[^\$]+}', 1, 1, 'i', 1)))

**Comments in SQL**

You can add explanatory comments in SQL formulas or comment out any portion of a formula that you don't want to delete. One way to enter a comment in SQL is to surround the comment text with asterisks and slashes in this order:

**Use slash-asterisk and asterisk-slash to comment out portion of SQL code**

/* Comment */

If you comment out a whole filter in Analytics, add 1=1 (which is always true), so that the filter will not return an error.
Example of a comment in a filter

/*Bibliographic Details".MMS Id" = '99153760346403941'*/1=1

MAX / MIN / Count Distinct "Subtotal" Values in a List Row

Use this syntax to provide MAX / MIN / Count Distinct Subtotals for non-numeric fields

MAX("xxx.column"  BY "yyy.column")
   Example: MAX("Invoice Line"."Invoice-Date" BY "PO Line"."PO Line Reference") [i.e., most recent invoice date associated with a given PO Line Reference]

MIN("xxx.column"  BY "yyy.column")
   Example: MAX("Invoice Line"."Invoice-Date" BY "PO Line"."PO Line Reference") [i.e., oldest invoice date associated with a given PO Line Reference]

COUNT(DISTINCT "zzz.column" BY "xyz.column")
   Example: COUNT(DISTINCT "Invoice Line"."Invoice-Number" BY "PO Line"."PO Line Reference") [i.e., count of invoice numbers associated with a given PO Line Reference]

Additional Formulas/SQL Queries from Ex Libris

Ex Libris has a list of other calculations that may be useful.

They also have regular expression examples to use with filters.

Containers for Additional Formulas/SQL Queries

These blank code blocks are here to help the Analytics & Reporting Working Group add more SQL queries to this list. Other staff should ignore them.

Placeholder for next formula

Edit the Code Block and add a Title

Placeholder for next formula

Edit the Code Block and add a Title

Placeholder for next formula

Edit the Code Block and add a Title