In this advanced session we will investigate the use of Alma Analytics to support Resource Management tasks. We assume some familiarity with common Alma Analytics tasks such as creating a new report, adding fields and filters, and saving modified reports. If you are new to creating reports in Analytics or need a refresher we recommend reviewing our [Introduction to Analytics](https://knowledge.exlibrisgroup.com/@api/deki/files/73280/Introduction_to_Analytics.pptx?revision=1) presentation slides before continuing.

**Table of Contents**

[**Electronic Collections Management in Connection With COVID-19** 2](#_Toc39595990)

[Copying the Example Reports 3](#_Toc39595991)

[Using Relative Dates 5](#_Toc39595992)

[Adding the Report to an Alma Widget 12](#_Toc39595993)

[**Using Functions to Format Data** 14](#_Toc39595994)

[Using CAST and REPLACE 17](#_Toc39595995)

[Bonus: Using CONCAT and CASE WHEN Instead of CAST and REPLACE 22](#_Toc39595996)

[**Using Advanced Functions to Extend Analytics Functionality** 29](#_Toc39595997)

[Using Database Functions with EVALUATE 32](#_Toc39595998)

[REGEXP\_SUBSTR: Extending SUBSTRING with Regular Expressions 33](#_Toc39595999)

[Using Regular Expressions to Match Patterns in Data 34](#_Toc39596000)

[Putting It All Together 36](#_Toc39596001)

[**Bonus: Using Advanced Functions to Work With Messy Data** 37](#_Toc39596002)

[Extracting Volume Data From Messy Descriptions 37](#_Toc39596003)

[REGEXP\_REPLACE: The Regular Expression Replace Function 42](#_Toc39596004)

# **Electronic Collections Management in Connection With COVID-19**

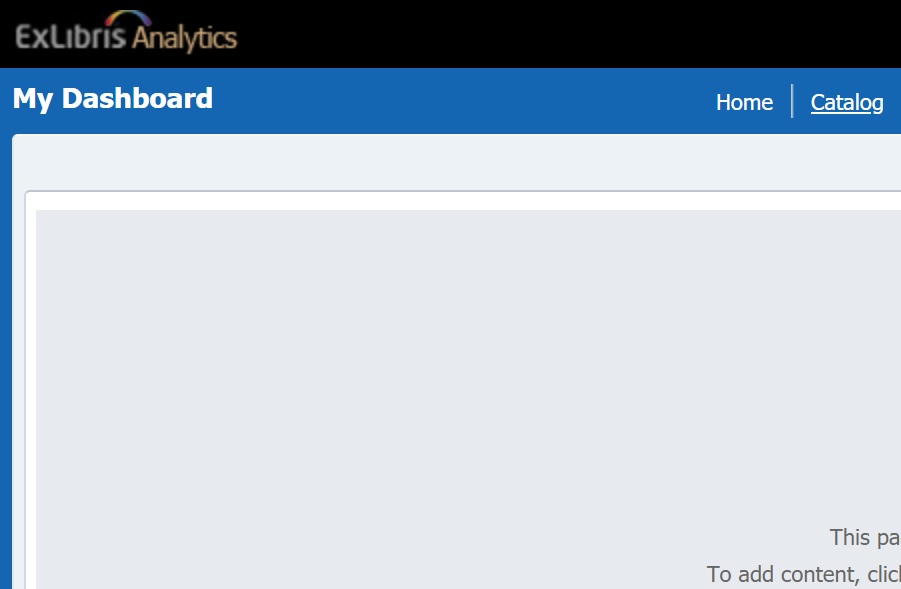
In response to the current global pandemic, Ex Libris has compiled lists of [best practices](https://knowledge.exlibrisgroup.com/Cross_Product/Best_Practices_for_the_Ex_Libris_Community_During_COVID-19), [COVID-19 and temporarily free resources](https://knowledge.exlibrisgroup.com/Alma/Content_Corner/Supporting_Resources/List_of_COVID-19_and_Temporarily_Free_Resources), and is hosting a series of webinars to help libraries and higher education institutions adjust to these difficult times. We would like to begin this session by exploring how Analytics can be used to help you face some of these challenges.

In the [Q&A section](https://knowledge.exlibrisgroup.com/?title=Cross_Product/Best_Practices_for_the_Ex_Libris_Community_During_COVID-19/Alma_and_Primo/04Q%26A) of the best practices article, we have made some recommendations for activating electronic collections in Alma for a limited period of time. [This presentation](https://knowledge.exlibrisgroup.com/@api/deki/files/80880/How_to_activate_an_electronic_collection_for_a_limited_time_period.pptx?revision=3) provides instructions for the recommended activation steps in Alma. We will not go over these steps in detail, but to summarize: when activating electronic collections in Alma fill out the “Active until date” field (we have dates for some of the electronic collections in our [COVID-19 and temporarily free resources](https://knowledge.exlibrisgroup.com/Alma/Content_Corner/Supporting_Resources/List_of_COVID-19_and_Temporarily_Free_Resources) article that came from the provider’s announcement). For electronic collections where you already had partial access and are now receiving temporary expanded access, activate the electronic collection for a second time (filling out the “Active until date” field as before), select the new portfolios, and then use the electronic collection’s “Public name (override)” field to distinguish it from your normal access (this will make it easier to deactivate later as you will not need to separate the temporary portfolios from your normal portfolios). If you follow a different activation workflow you may find that the example reports do not return the expected data and require some adjustment.

While it is possible to search Alma for electronic collections with specific active until dates (as demonstrated in the presentation linked above), many of these electronic collections will have different active until dates and these dates may be subject to change as some providers decide to extend access further. What would be preferable is a way to identify (on an ongoing basis) any temporarily activated electronic collections where access will be expiring soon so that you can review these and deactivate or change the active until date as appropriate. In our first example report we will do just that.

## Copying the Example Reports

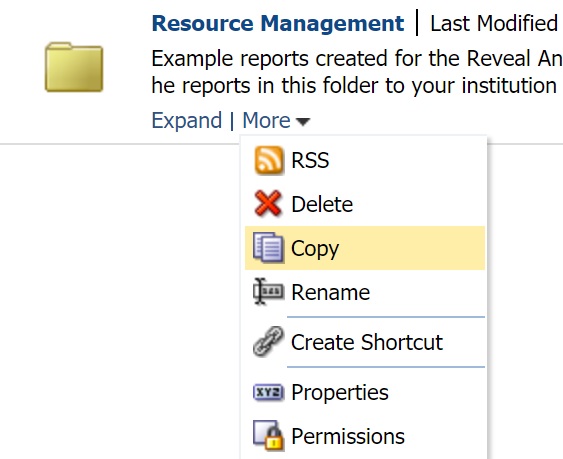
In this session we will be modifying some example reports that we created earlier. The example reports for this session can all be found under the Shared Folders/Community/Reports/Shared Reports/Knowledge Days 2020/Resource Management folder (please note that if you are reading this before May 9th, 2020 the reports may not appear in this folder yet). If you would like to follow along and complete the exercises, we ask that you first copy these reports to your My Folders section or a folder under your institution’s directory in the Shared Folders section and work on the copies. To do this, click on the Catalog link in Analytics:



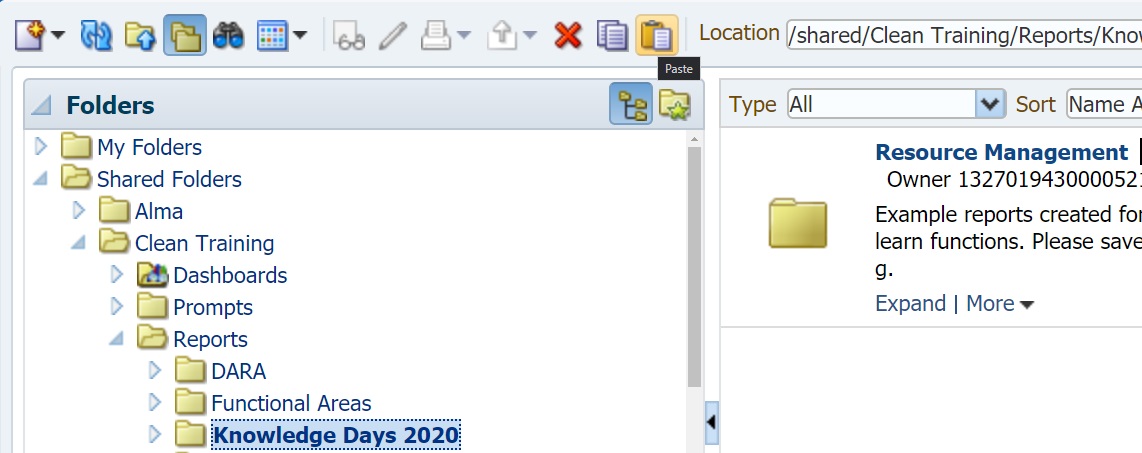
Navigate to the Shared Folders/Community/Reports/Shared Reports/Knowledge Days 2020 folder:



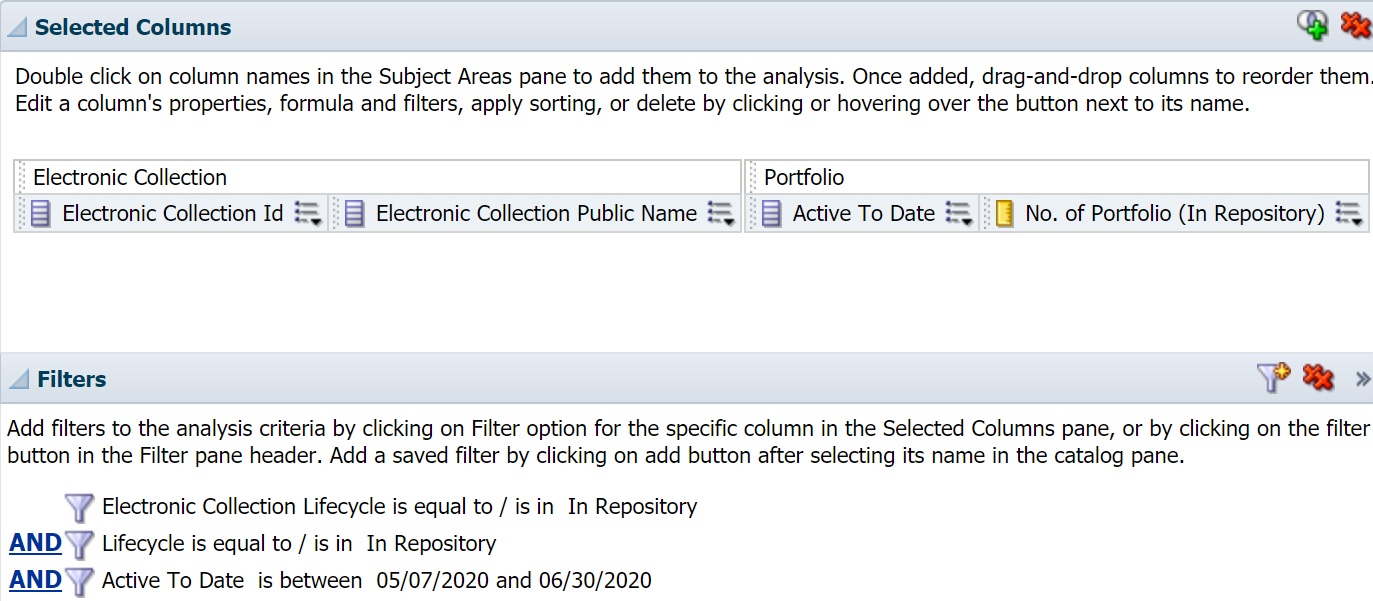
Under the Resource Management folder click More and then click Copy:



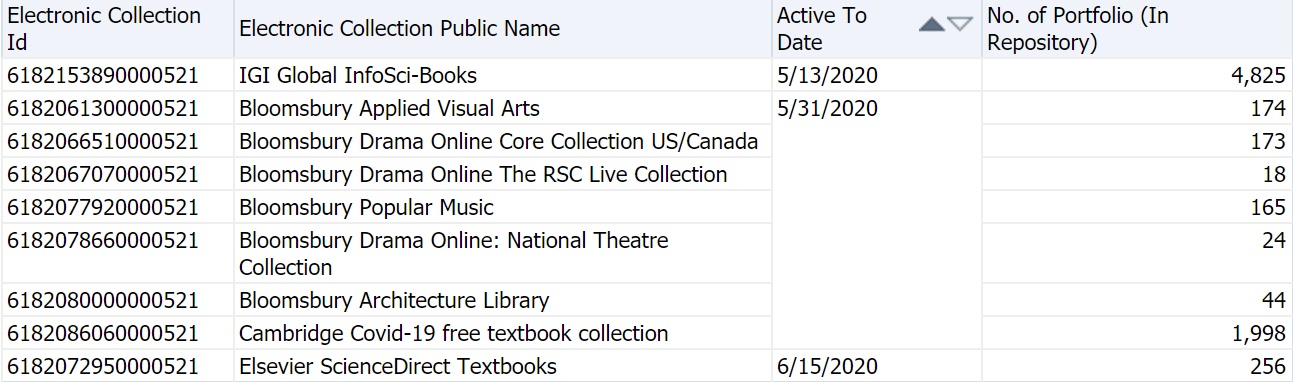
Navigate to a directory in your institution’s folder (our training environment’s institution name is “Clean Training”) and click the Paste icon to copy the folder and all the reports inside to the desired directory:



Expand the Resource Management folder and Edit the first example report “Upcoming Electronic Collection Deactivations”. This E-Inventory report displays active electronic collections with Active To Dates between two specific dates (the Active To Date field is associated with the portfolios, which inherit the Active until dates in Alma from the electronic service):



And here are the results in our training environment, where we have temporarily activated several electronic collections following the best practices recommended in the beginning. Your results will differ but if at least one result appears you will be able to complete the exercise (if no results appear try adjusting the dates in the Active To Date filter):

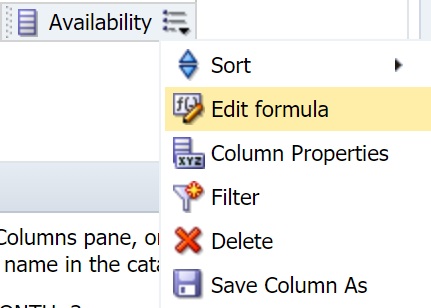


We encourage you to add other fields and filters to this report at the end so that it better meets your reporting needs. However, for now it is a good idea to keep the report simple so that we can verify the report’s filter logic is working when we make the replacements in the next section.

## Using Relative Dates

The report in its current form is not much different from an Alma search, it would be much more useful to us if we could run the report at any time in the future and not need to adjust the Active To Date filter. Instead of entering specific dates like “5/07/2020” we’d prefer to specify relative dates like “today” “next month” or “next week”. To do this we will be replacing the dates with SQL functions, but first we will be temporarily adding them to this report as fields to show how they work. If you happen to already be familiar with the CURRENT\_DATE and TIMESTAMPADD functions, you can skip to the next section.

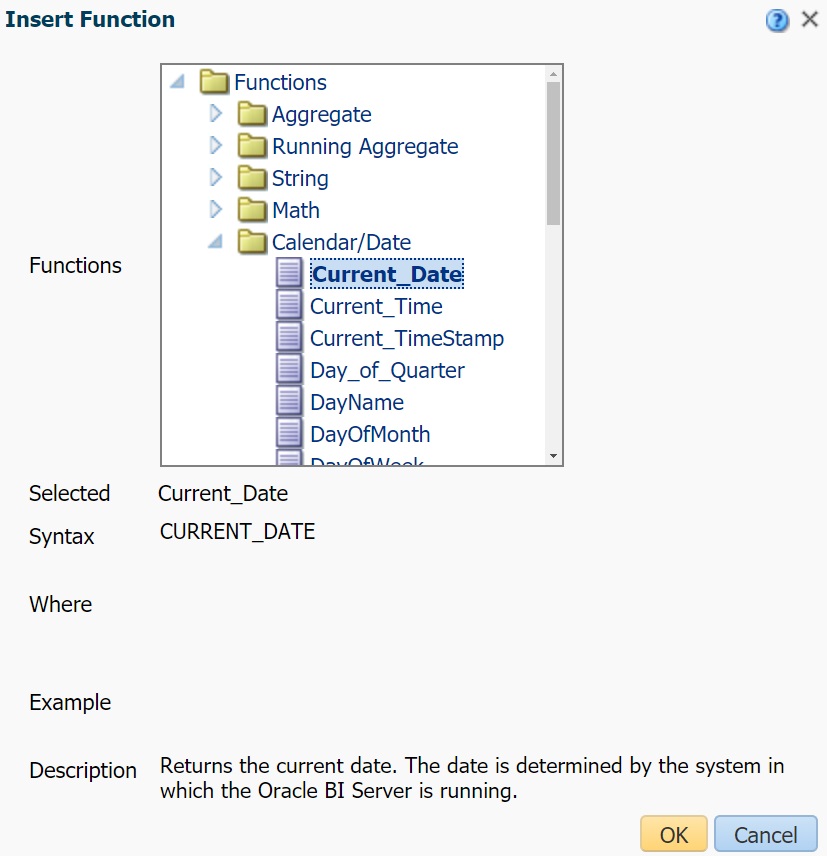
Go to the Criteria tab and add a new field to your report, it does not matter which field you select as we will be replacing it entirely with a custom formula (we chose “Availability” simply because it was near the top of the list of available fields). In the drop-down menu next to the field select “Edit formula”:



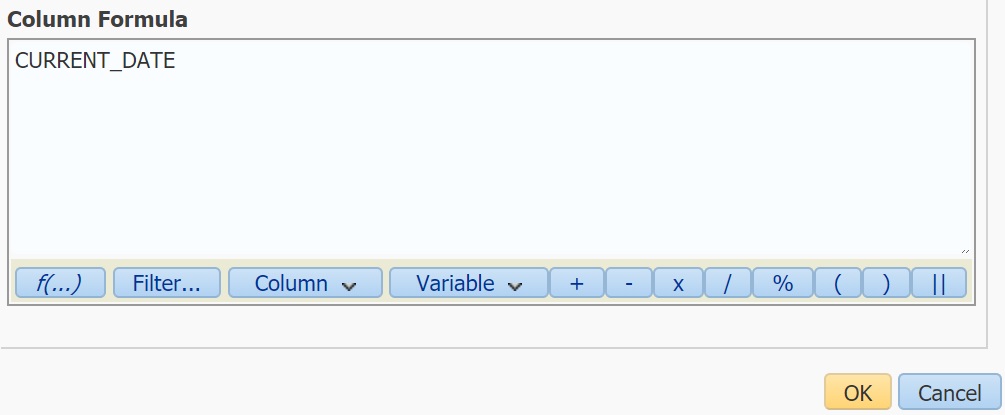
Remove the field you selected from the Column Formula so that this section is blank and then click the *f(…)* button:



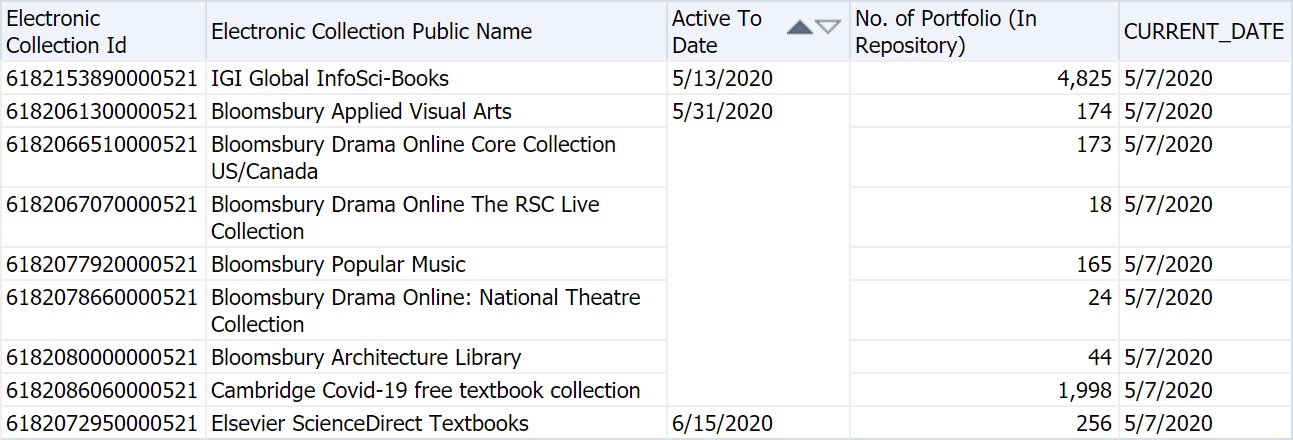
Open the Calendar/Date folder and select Current\_Date. Analytics will provide a brief description of the field. Click OK:



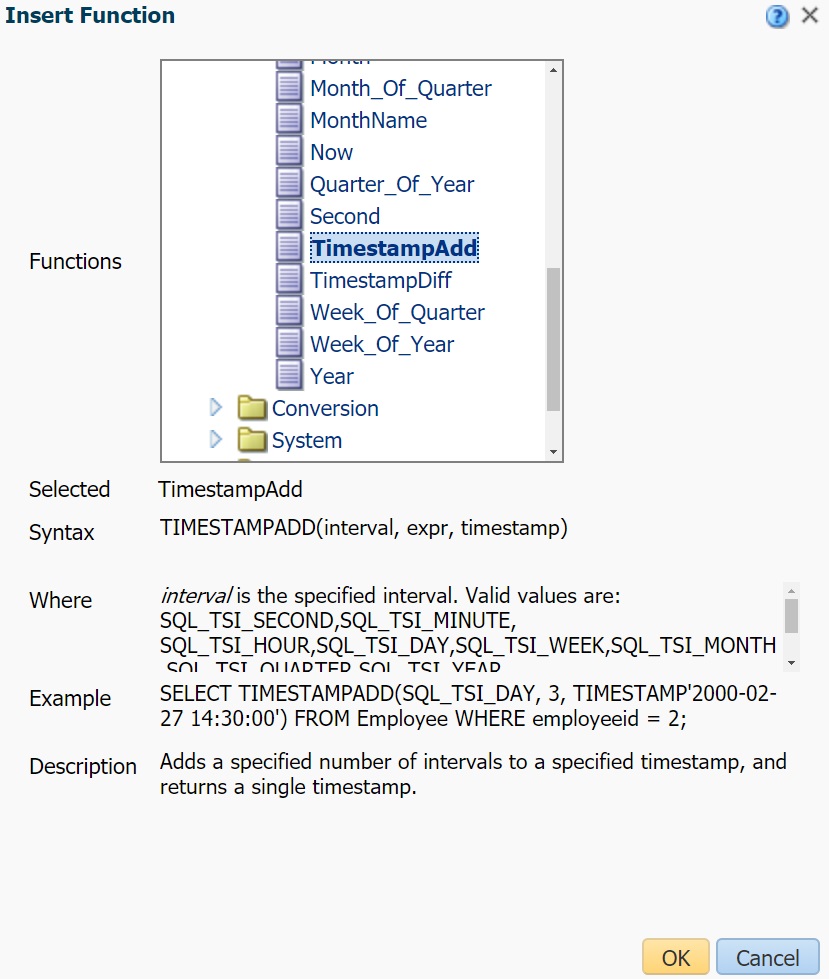
Confirm that the Column Formula section has the value CURRENT\_DATE and click OK again:



Go to the Results tab and confirm that the CURRENT\_DATE field shows today’s date in every row:



Go back to Criteria and select Edit formula again. As before, clear the Column Formula section, open *f(…)* and this time select the TimestampAdd function from the Calendar/Date folder. Before clicking OK again, note that helpful information appears explaining the syntax for the TIMESTAMPADD function:



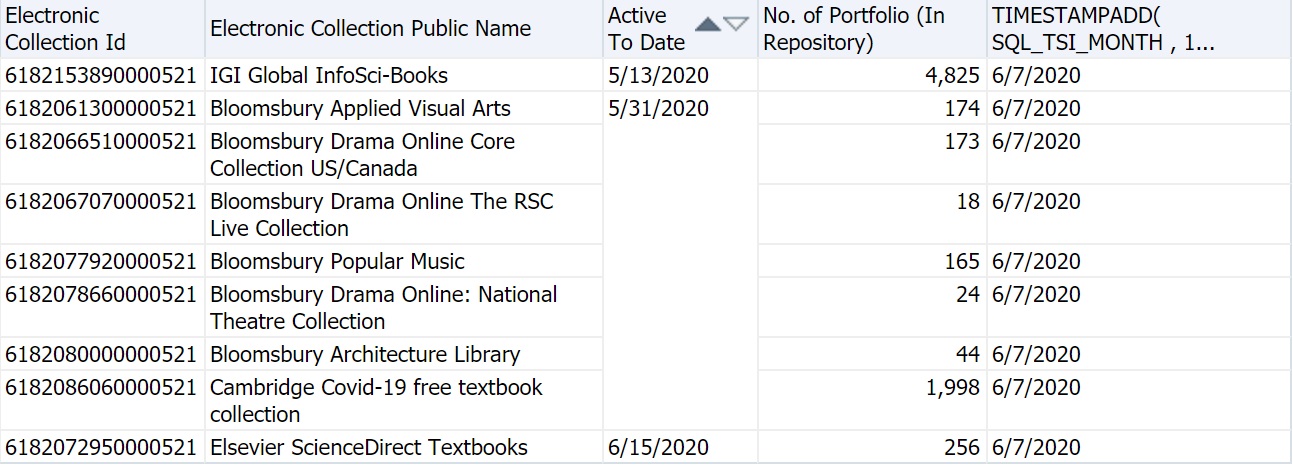
The Column Formula section should now contain TIMESTAMPADD(interval, expr, timestamp). The TIMESTAMPADD function accepts a timestamp (the example above shows a timestamp for a specific date and time, 2:30pm on February 27th, 2000, but we will be using CURRENT\_DATE to create a relative date) and returns another timestamp that is increased or decreased by the amount of time specified. Interval is the unit of time (e.g. days, weeks, months, etc.) that is increased or decreased and expr (“expression”) is a positive or negative integer that specifies how much the timestamp is increased or decreased.

In the Column Formula section, we will need to replace interval, expr, and timestamp in this function with our desired settings. Replace interval with SQL\_TSI\_WEEK or SQL\_TSI\_MONTH (depending on whether you would like your report to show electronic collections with an Active To Date within the next week or next month respectively). Replace expr with a positive number such as 1 or 2 (negative numbers can be used to create relative dates in the past such as “last week” or “last month”) and replace timestamp with CURRENT\_DATE. Here are two example TIMESTAMP functions after these replacements are made:

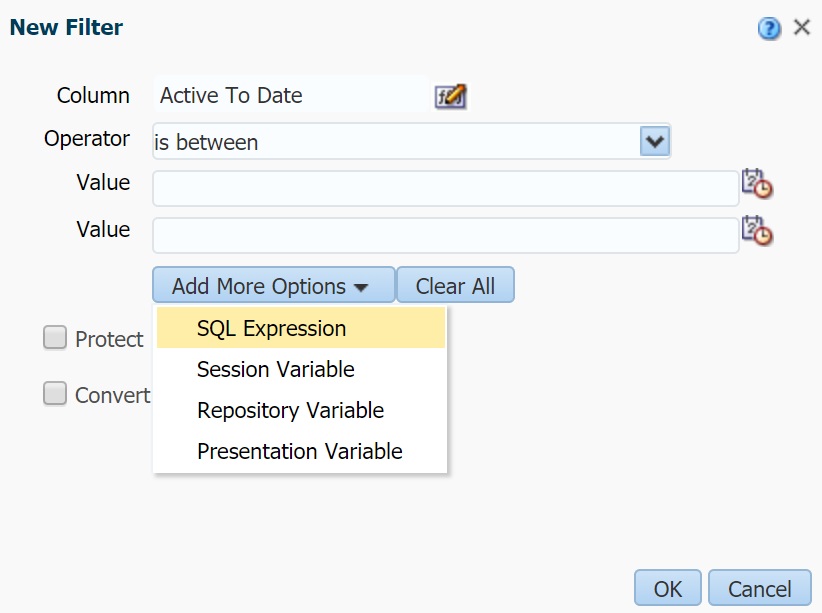
Two weeks from today: TIMESTAMPADD(SQL\_TSI\_WEEK, 2, CURRENT\_DATE)

One month from today: TIMESTAMPADD(SQL\_TSI\_MONTH, 1, CURRENT\_DATE)

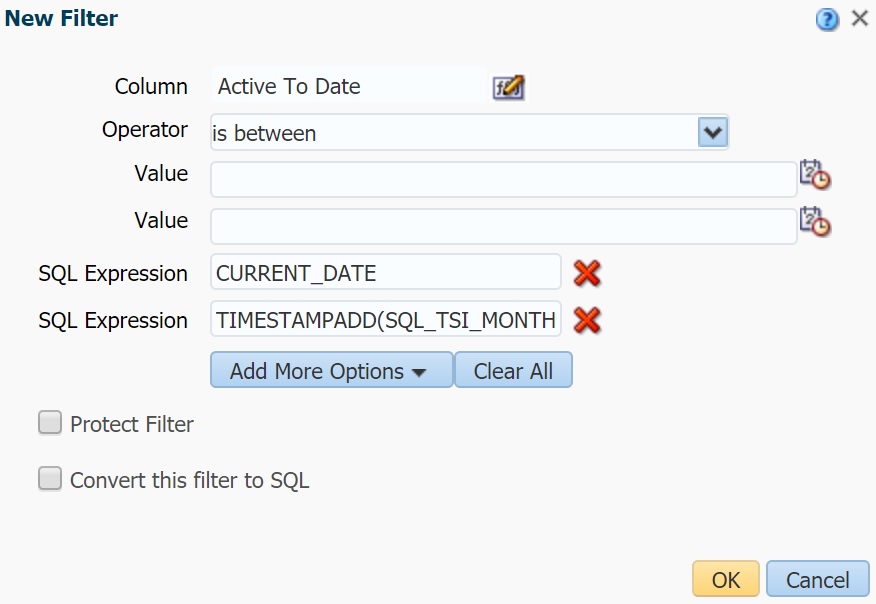
After making your desired replacements, click OK go to the Results tab again to confirm that the date displayed in this field is one week, month, or some other amount of time from today’s date:



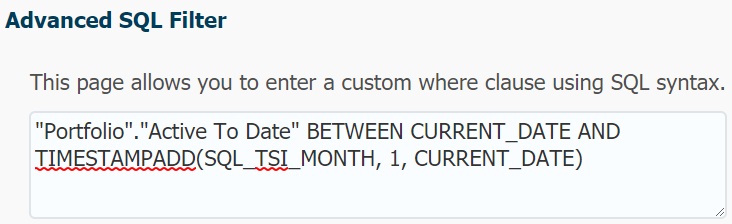
Now that we have familiarized ourselves with these functions, we can remove this custom field from the report and add the functions to the Active To Date filter instead. Go back to the Criteria tab and remove the field by selecting Delete from the drop-down menu next to it (you may want to copy your TIMESTAMPADD function to your clipboard first). Hover your mouse over the Active To Delete filter and click the Edit Filter button (pencil icon). Remove the values you entered previously, click the Add More Options button and select SQL Expression from the drop-down menu:



A new SQL Expression field will be added to the menu. In this field enter CURRENT\_DATE. Repeat the above steps to create another SQL Expression field and enter your copied TIMESTAMPADD function into the second field:



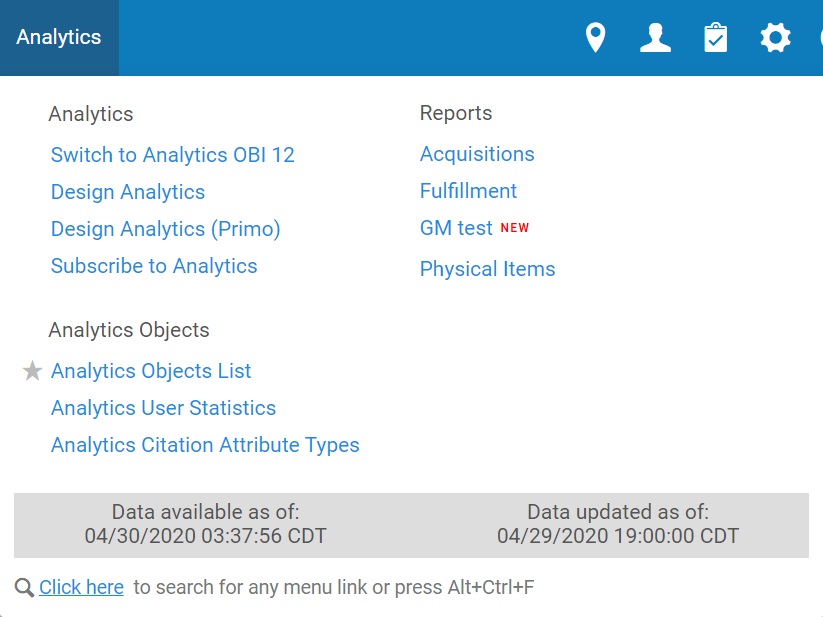
Alternatively, you can check the “Convert this filter to SQL” option, click OK and replace the dates that appear in the Advanced SQL Filter menu with these functions (this filter will display slightly differently in the Criteria tab from the filter above, but they are equivalent filters):



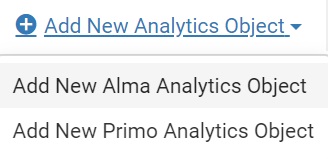
Save your report and check the Results to confirm that the modified filter is working. If you do not see any results you may want to try increasing the number (expr) in the TIMESTAMPADD function of your Active To Date filter until a result appears (this will help confirm that your report is working but no temporarily activated electronic collections have active until dates that are expiring soon).

## Adding the Report to an Alma Widget

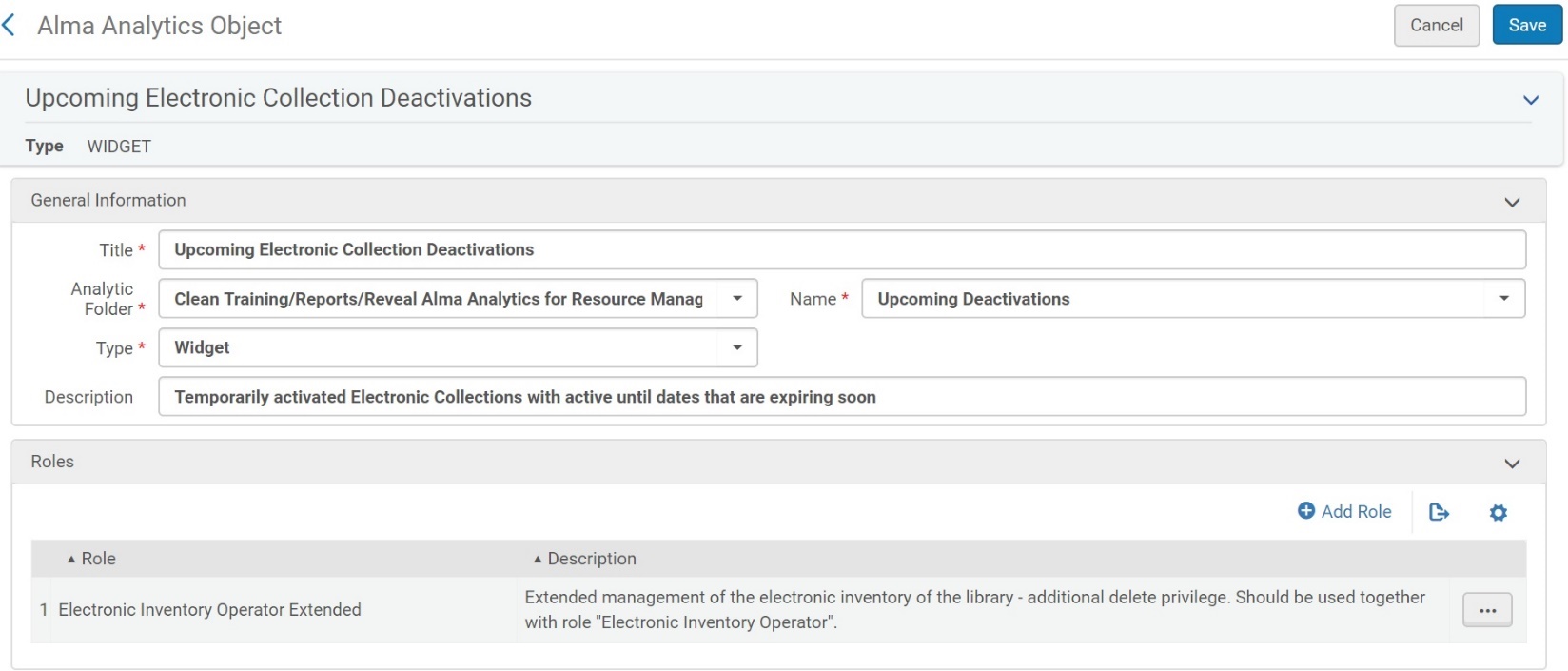
Now that we have created our desired report, we can make it even more convenient to keep track of these electronic collections by adding it to Alma as a widget. In Alma, open the Analytics tab and click Analytics Objects List under the Analytics Objects section:



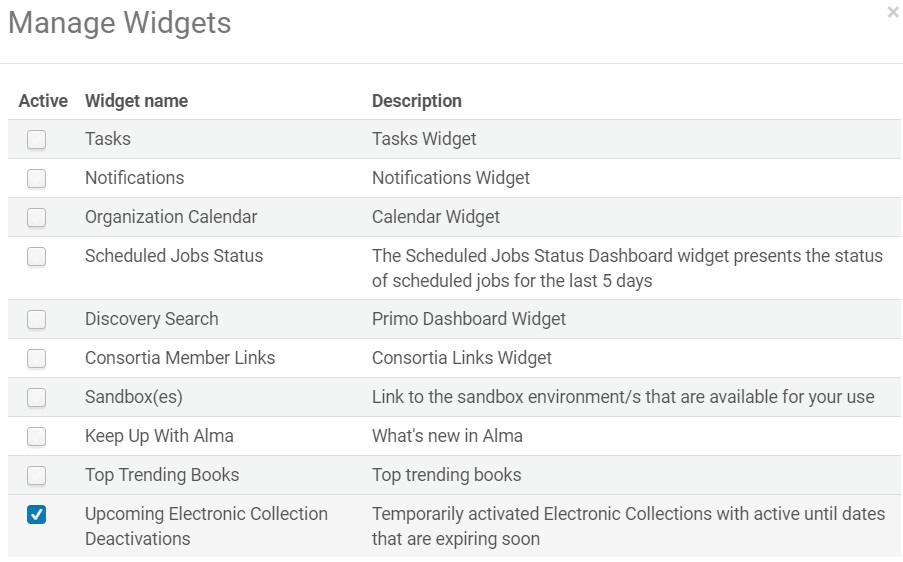
From the Analytics Configuration and Subscription page click the Add New Analytics Object option on the top right and select Add New Alma Analytics Object:



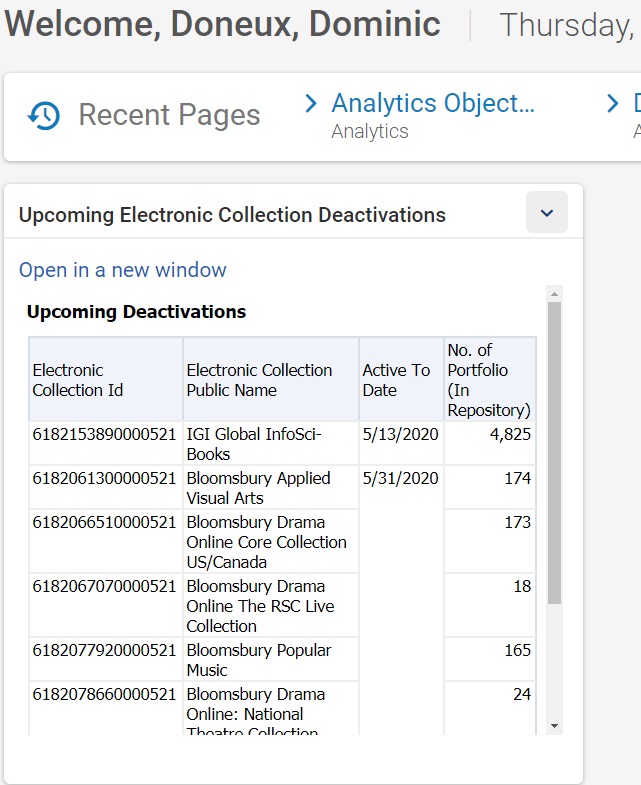
From the Alma Analytics Object page, enter a title and description for your widget, select the folder the report is saved from the Analytic Folder menu (please note that your report must be saved to one of the Shared Folders directories) and then select your report in that folder from the Name menu. Finally, you must add at least one Role to the widget that will determine which users can add your widget to their Alma homepage. For this report the Electronic Inventory Operator Extended role is a good choice as users with this role can delete electronic collections:



From the Alma home page click the + icon to open the Manage Widgets page and check the box next to your new widget:



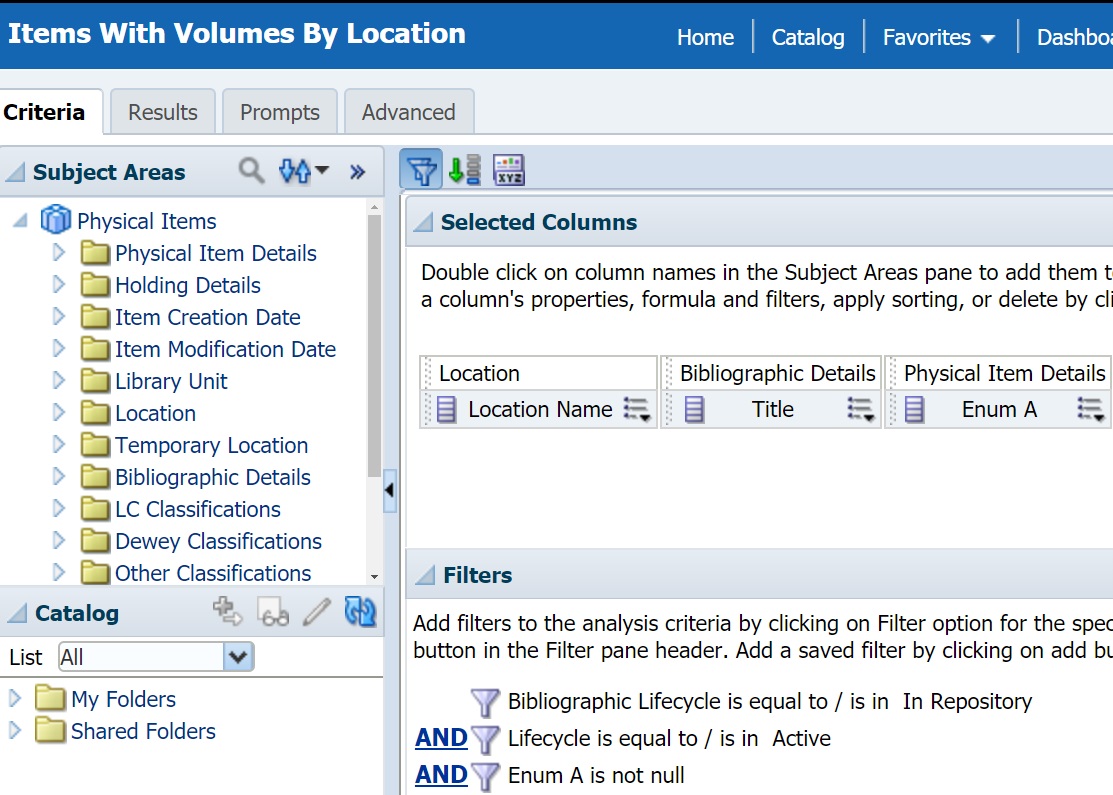
Now your report will display on your Alma homepage and will update once per day to show any new electronic collections with access that will be expiring soon:



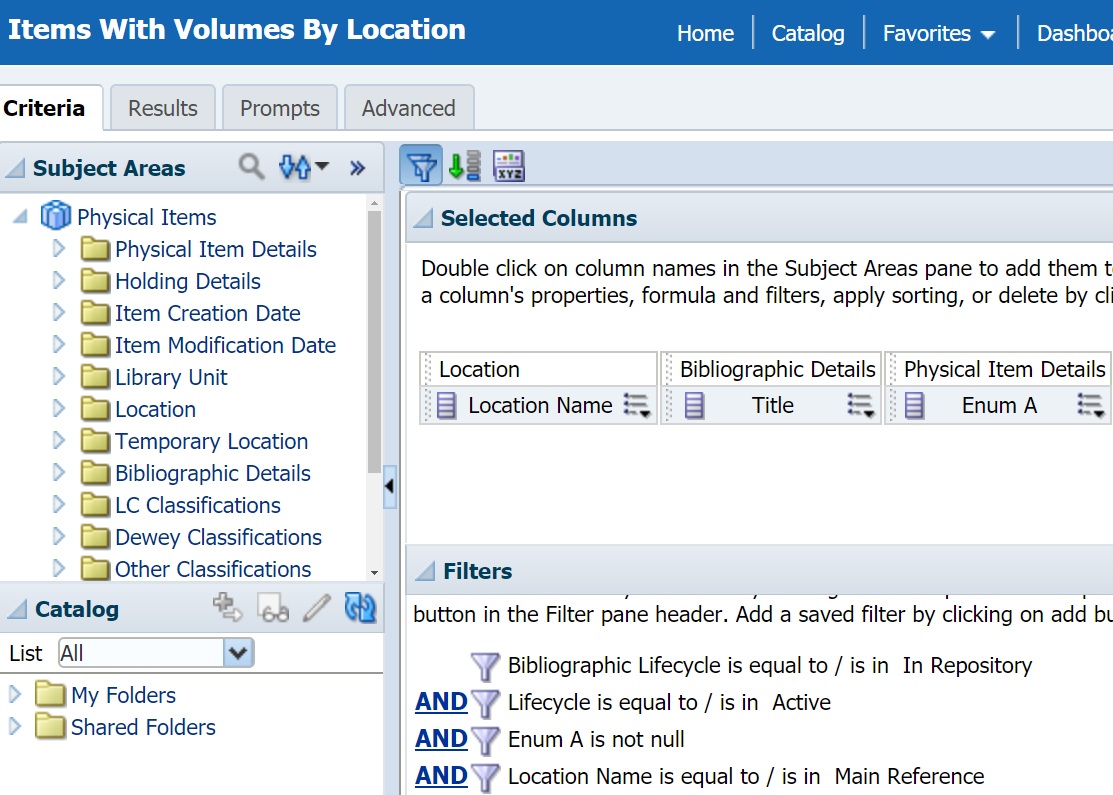
# **Using Functions to Format Data**

In this next section we will be exploring ways to modify the data that appears in Analytics reports using functions. Sometimes the data that appears in Analytics does not quite meet our reporting needs for various reasons, for example we might not have entered this data into Alma in the format we now need for reporting, we might have been inconsistent when entering this data over time, or the field in Analytics that contains the data we need might not have been designed with our current use case in mind. For this last scenario, we recommend reporting these issues to us by filing Support cases or submitting ideas to NERS/the Ex Libris Ideas Exchange as appropriate so that our developers can add or modify the fields in Analytics to better address your needs. In the meantime, functions can help meet your reporting needs until the field is added. For example, in the January 2020 release our developers added OCLC Control Number fields to the Bibliographic Details folder; prior to this it was possible to extract these OCLC numbers from the Network Number field using an advanced function called REGEXP\_SUBSTR, which we will cover towards the end of the presentation.

For our first example we will be looking at a report that has relatively clean data (at least in our training environment) that can be formatted with built-in functions. The “Items With Volumes By Location” report shows titles by location that have volume information in the Enum A field in Alma (at least in our training environment, if you do not use this field in your Alma environment you may want to go to the next report):



First, add a filter to this report for the Location Name so that we can work with a small number of results. We chose the Main Reference location in our training environment because there was about one page of results for this location and the Enum A data for these titles in Alma was fairly clean (only volume numbers were entered in Enum A and only one minor inconsistency appeared for a couple results: “vol. 1” and “vol. 2” instead of “1” and “2”. If the Enum A data is much messier in your Alma environment, you may need to use the advanced functions demonstrated in the next report):



Looking at the Results tab, we can see an annoying sorting problem:

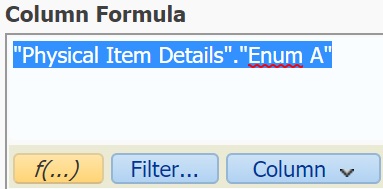
A screenshot of a social media post

Description automatically generated

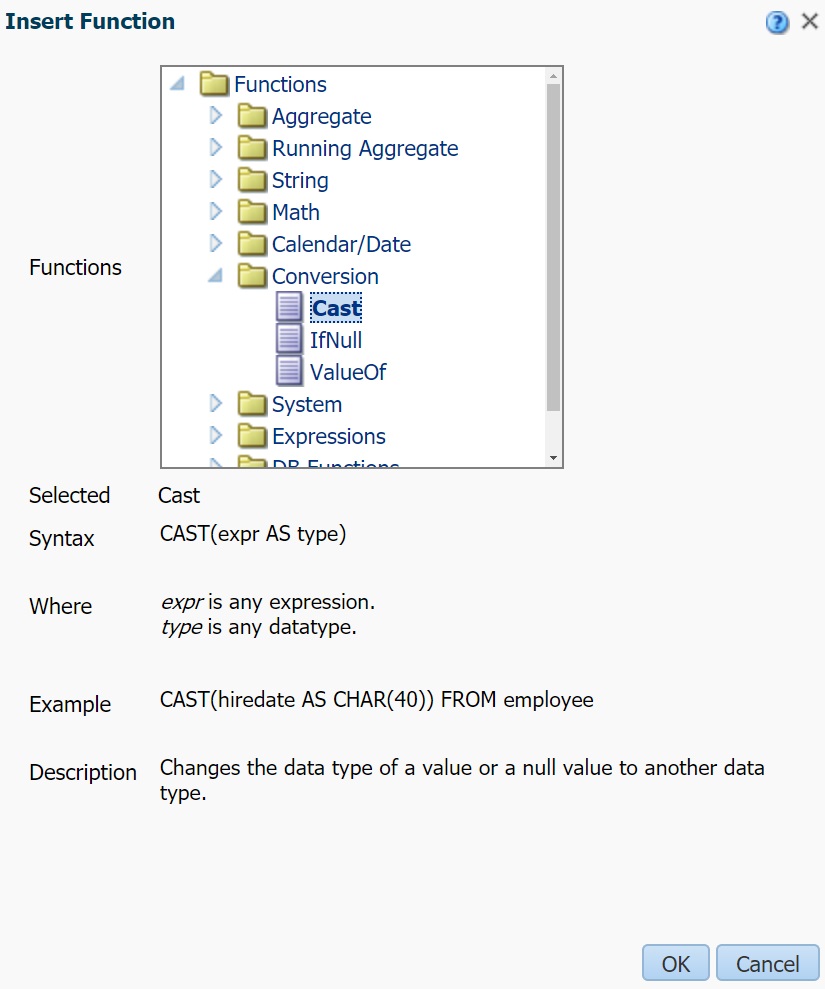
Because the Enum A field in Alma allows you to enter data in any format (as seen in the “21st Century” titles above where “vol.” was included) it has been mapped to Analytics as a text field. As a result, instead of sorting the field numerically Analytics is sorting the field alphanumerically, so “10” appears after “1” instead of “2” because the text begins with “1”.

## Using CAST and REPLACE

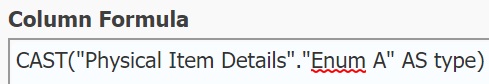
There are many ways we could fix this issue with functions and we will explore a couple of them here. First, if we can convert Enum A from a string field into a numeric field then Analytics will sort it numerically like we wanted. This can be accomplished using the Cast function. Go back to the Criteria tab and select “Edit formula” under the drop-down menu for Enum A. By default the text in the Column Formula section should be highlighted; if not, highlight it with your mouse as shown below so that this field will appear inside the function once we add it (if it is deleted you can always add it back from the Subject Areas section on the left). Click the *f(…)* button:



Open the Conversion folder, select Cast and click OK:



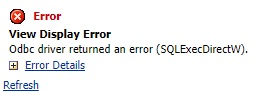
If the Enum A field was highlighted it should be added to the beginning of the CAST function in the Column Formula section, if not move it there. A placeholder for the type parameter appears which we will need to replace:



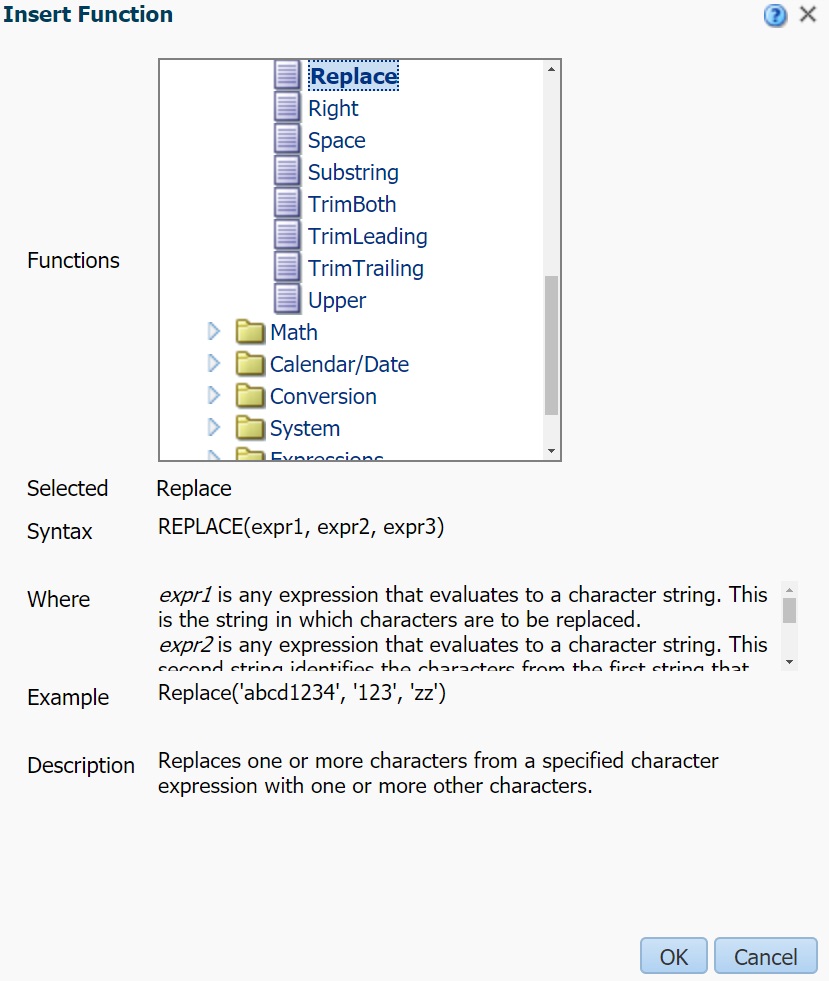
[Oracle documentation](https://docs.oracle.com/cd/E21764_01/bi.1111/e10544/appsql.htm#CHDFHDEJ) contains a list of the allowed data types that you can specify in the type section of the CAST function. [This table](https://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqlj33562.html) is also very helpful to check as not every data type can be converted into every other data type using CAST. For this report we want to use INT for the integer data type:



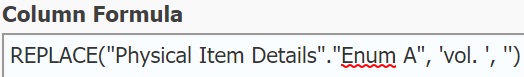
When using the CAST function, you need to be very careful to ensure that all the results in your report contain data that will be valid for the new data type you are converting it to. If even a single value is invalid, you will see an error message like this when you go to the Results tab:



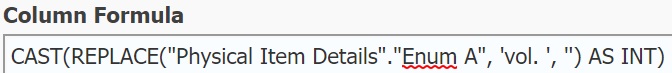
For the data in our training environment, the cause of this error was the “vol.” values we saw earlier (“vol. 1” and “vol. 2”). Depending on the data in your Alma environment and which location you chose for your filter, you might not see this error at all or the cause of the problem might be a different series of non-digit characters (e.g. it could be “v.1” or “Volume 1”). Regardless, one limitation of the CAST function is that it does not know how to ignore strings like “vol. ” when converting the data into a number and returns an error. We will need to remove “vol.” from the string before using CAST and can do this with the Replace function, which can be found under the String folder:



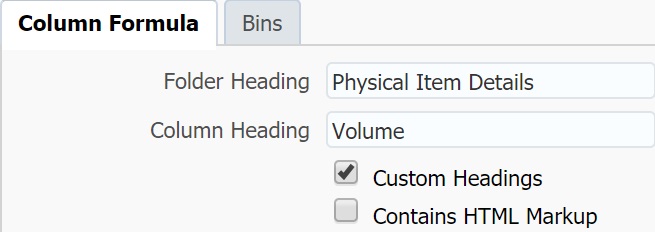
When using REPLACE, the first expression contains the string you will be replacing (in this case the Enum A field), the second expression is the string to match, and the third expression is the string that will replace the matched string. We want to remove “vol. ” from Enum A so we will use ‘vol. ’ (make sure to include the space at the end) as our second expression and ‘’ (i.e. an empty string) as our third expression:



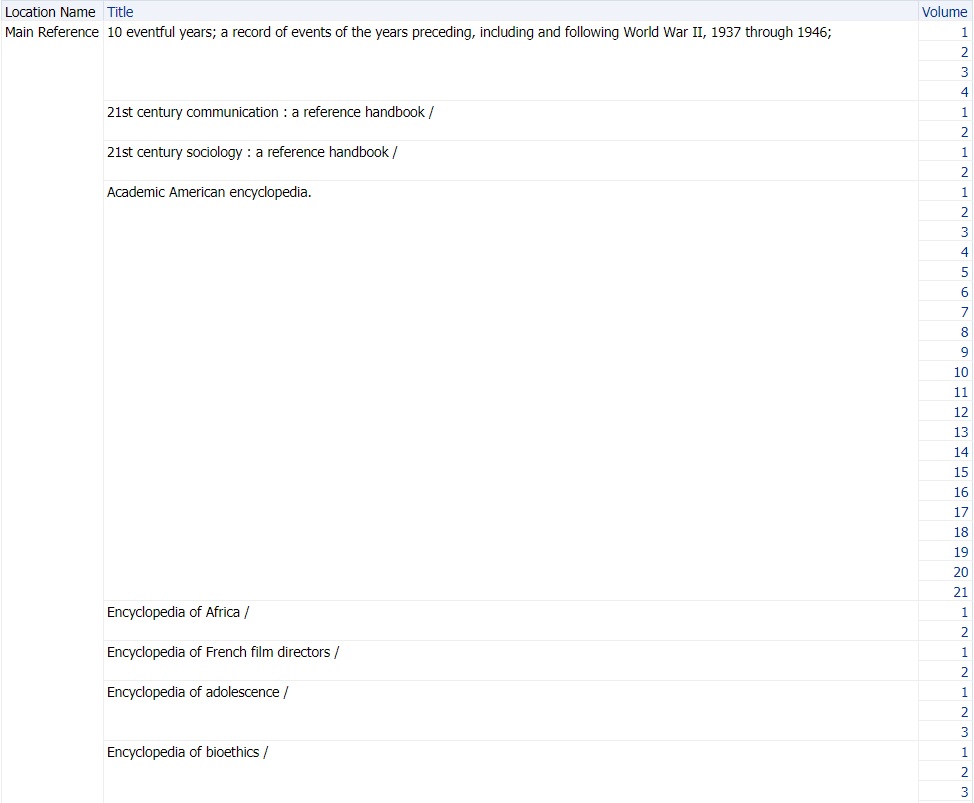
Now we can nest the REPLACE function inside the CAST function and convert the string to an integer with INT:



By default, Analytics will use the function as the column heading, but this would produce a very long column name in our report that you probably do not want. We will replace this with the custom column heading “Volume” by checking the Custom Headings option and entering a custom column heading in the Column Heading field:



The results are now sorted numerically and as a bonus the inconsistent volume numbers that began with “vol.” are in the same format as the other volume numbers in the report:



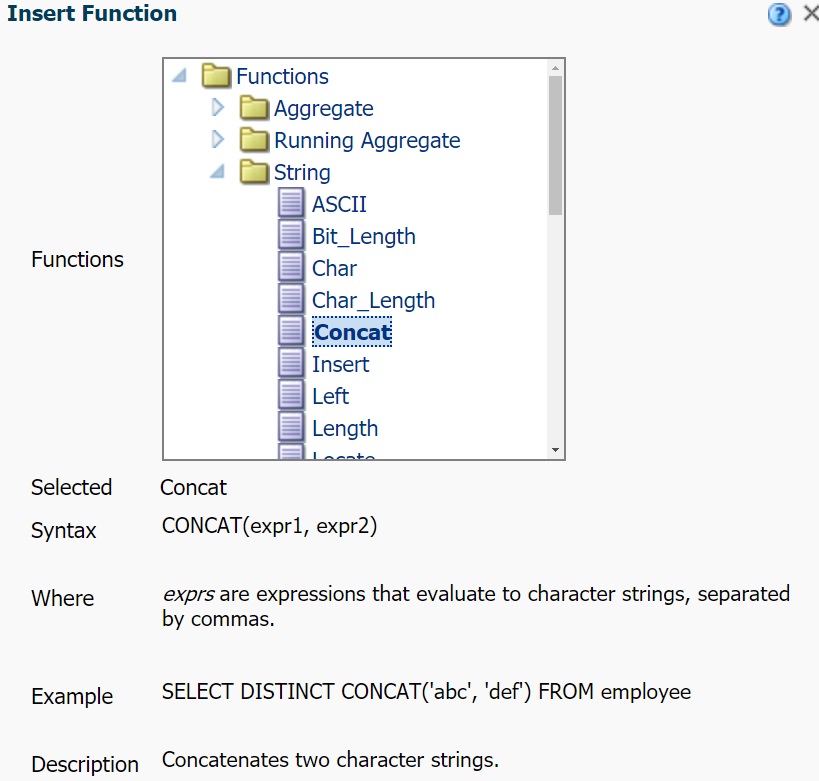
## Bonus: Using CONCAT, LENGTH and CASE WHEN Instead of CAST and REPLACE

While this function happened to work for the current data in the report, what happens if we remove the Location Name filter to expand this data or one of our staff later enters a new variation such as “Volume 1” for a new item? Unfortunately, the CAST function would return an error again. We could add another REPLACE function for “Volume”, but now we are starting to play “Whac-A-Mole” with our data.

It would be better if we could find a way to sort Enum A without using a sensitive function like CAST, or if there was some way to remove everything except the digits from this field so that CAST will always work properly. We will explore the former first, but the latter is also possible using the advanced function we will introduce in the next report.

One way to fix the sorting issue without needing to convert the data with CAST (and still use built-in functions) would be to add a “0” to the beginning of the Enum A values that are only a single digit (e.g. “1” becomes “01” but “10” stays the same) so that the alphanumeric sort will sort them as desired.

The Concat function can be found in the String folder. It will concatenate two strings together and is what we will use to add “0” to the beginning of some Enum A values. Analytics shows the syntax and example usage:



After clicking OK, Analytics adds the Enum A field to the first expression, which would be good if we wanted to add something to the end of the Enum A field, but since we want to add it to the beginning we will need to change this:



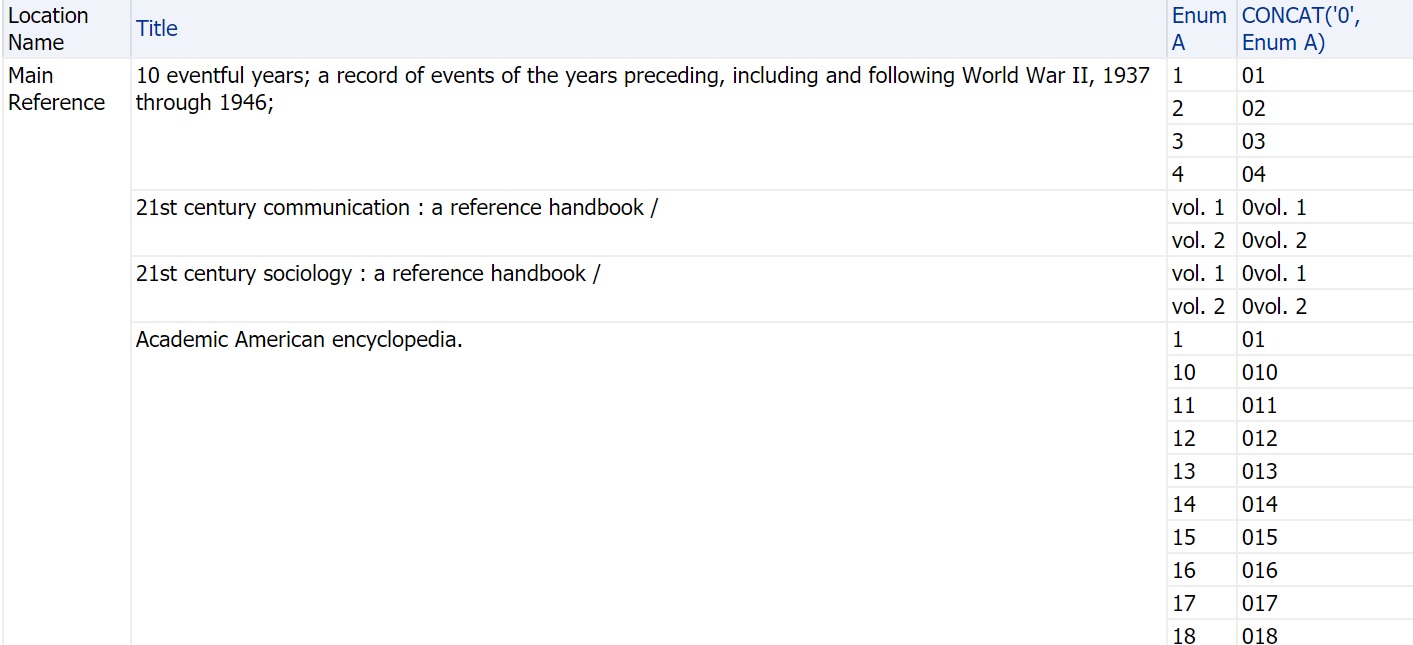
Move the Enum A field to the second expression and in the first enter ‘0’ (single quotes must be used for string literals, i.e. the string “0” instead of the number 0. If just 0 is entered it will be treated as a number and the concatenation function will not work):



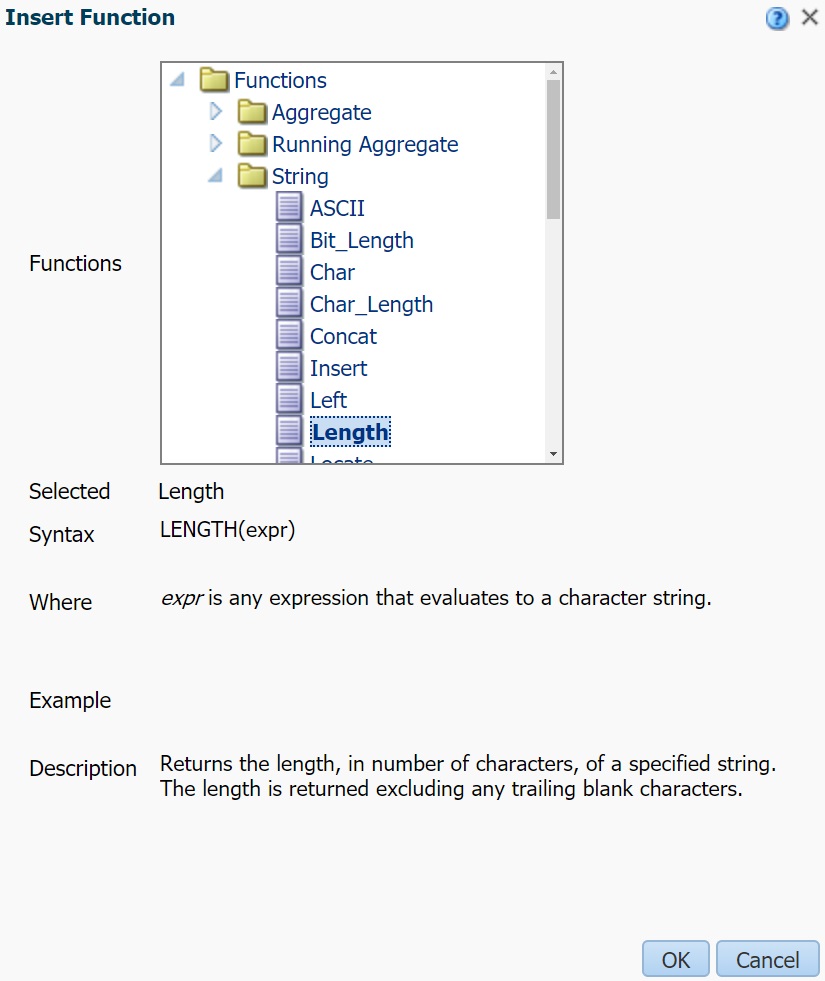
String concatenation is such a common operation that there is a shortcut, you can replace the CONCAT function by entering two pipes || or clicking the || button. If you use this option, the first string must appear to the left of the pipes and the second string must appear on the right:



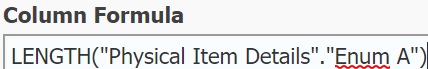
If we go to the results, we can see that “0” was added but to all values, which doesn’t fix the sorting issue and introduces a new problem for the “vol.” values. We will need to revise this function so that it only applies to Enum A values with a single digit:



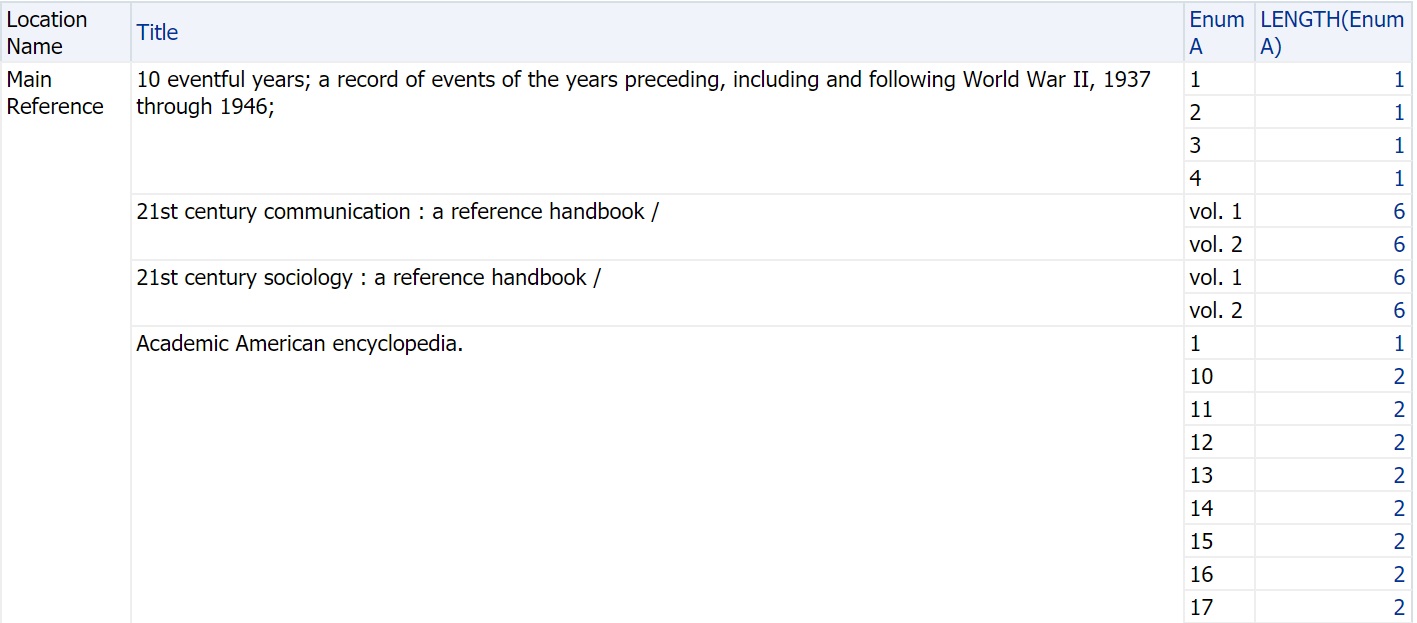
First, we need a way to determine the length of the Enum A field. The Length function, also under the String folder, will help us do just that:



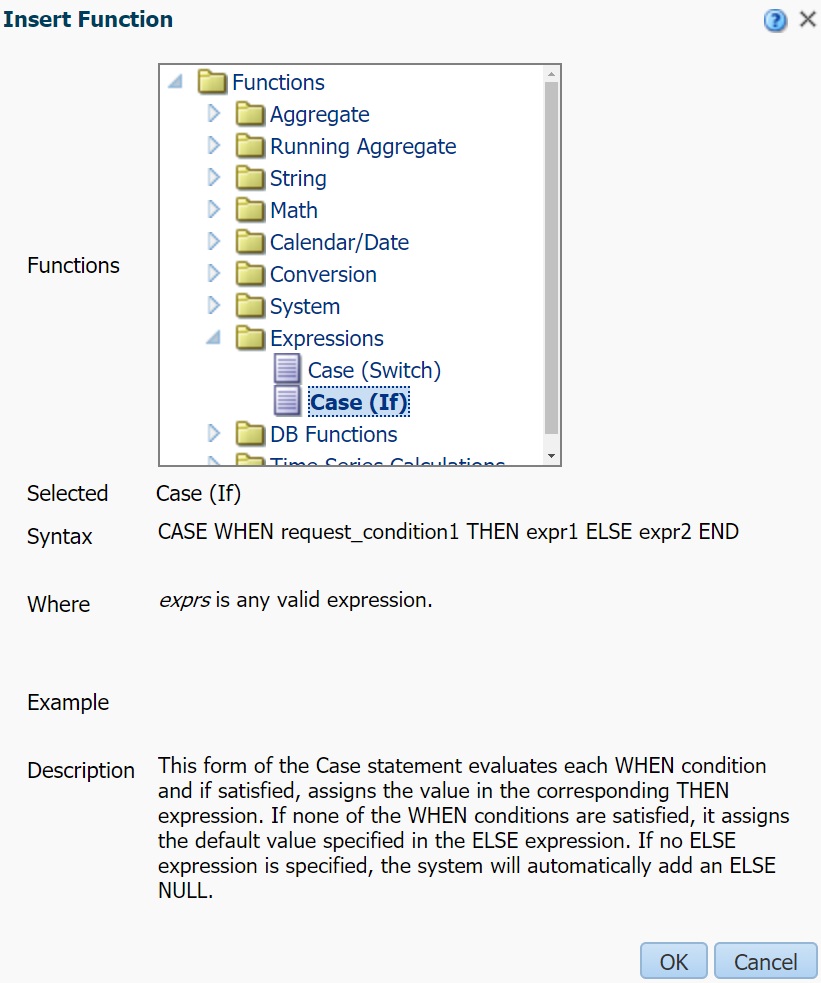
Here is the length function being applied to the Enum A field:



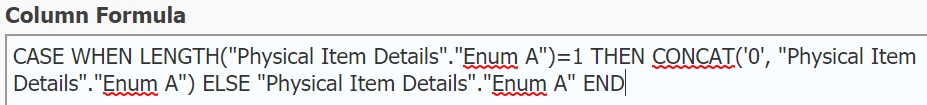
Checking the results, we can see that length will help us distinguish values like “1” and “2” from “10” and “vol. 1”. We will want to apply our CONCAT function only to Enum A values where the length is 1:



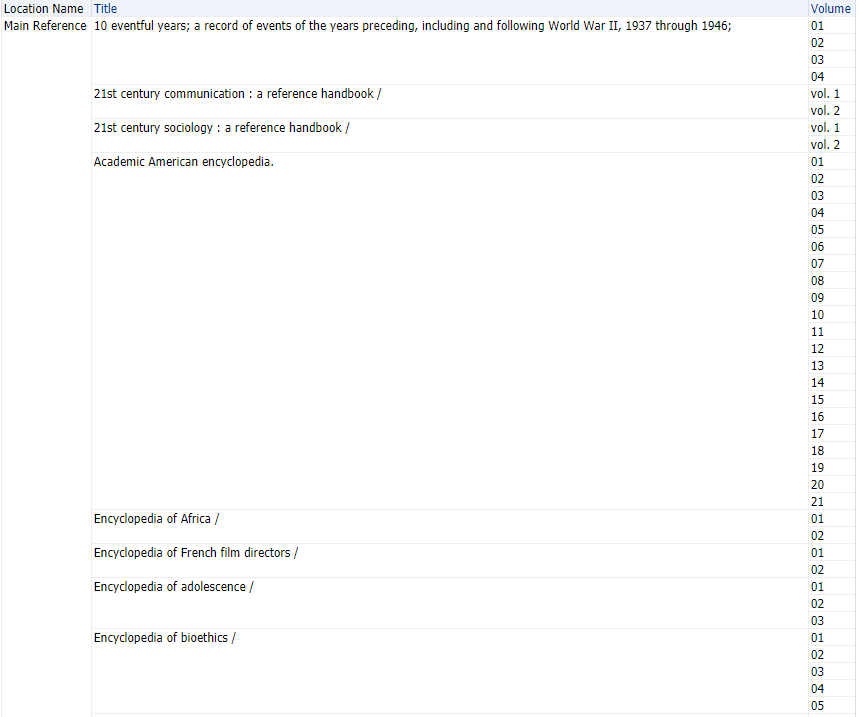
In the Expressions folder there are two Case functions that both use similar syntax. Case (Switch) allows for multiple WHEN conditions but we only need one in this report so we will use Case (If). This function returns the value of one expression if a condition is true and another expression if the condition is false:



Applying this function will require a few changes and reintroducing our previous functions (functions can be nested in other functions to produce even more complex functions). For the condition we want to add LENGTH(“Physical Item Details”.”Enum A”)=1 (since LENGTH returns a number, we will use 1 instead of “1” here). The expression after THEN will be used if the length is 1, so we will enter the CONCAT expression from earlier here (either CONCAT(‘0’, “Physical Item Details”.”Enum A”) or ‘0’||”Physical Item Details”.”Enum A” depending on what you find easier to read). Finally, the expression in the ELSE section will be used if the length is not 1, so in this case we want to return the unmodified Enum A value using “Physical Item Details”.”Enum A”:



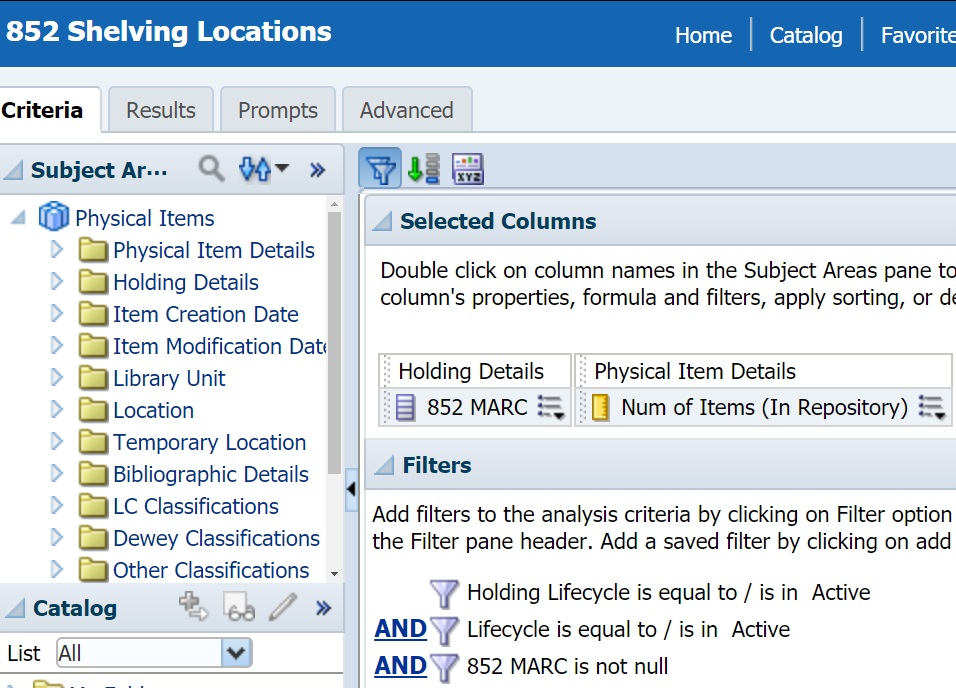
Finally, we will remove the original Enum A field and go to the results. Our custom “Volume” field is sorting correctly with only the single digit volumes receiving a “0” at the beginning:



Of course, if we had any items with “vol. 10” or above we would run into the sorting problem again. From these examples we can see that there are some limitations with the built-in functions when it comes to messy data. While this confirms the importance of entering data consistently into Alma, there are some advanced functions that can extend the built-in functions to handle a wider variety of messy data. We will explore these functions with the next report.

# **Using Advanced Functions to Extend Analytics Functionality**

Consider the “852 Shelving Locations” report below in which we would like to count the number of items associated with shelving locations that are specified in the holding records’ 852$c. This data appears in Analytics, but only as the full 852 field (852 MARC, which can be found under Holding Details). There are other fields like this in Analytics such as the Local Param and Holding Local Param fields where you may want to do something similar, we are using the 852 MARC field in this example mainly to ensure that the report will work in any Alma environment:



Because of the other subfield data present in the 852 MARC field, it is difficult to determine the totals for each shelving location code in the 852$c:



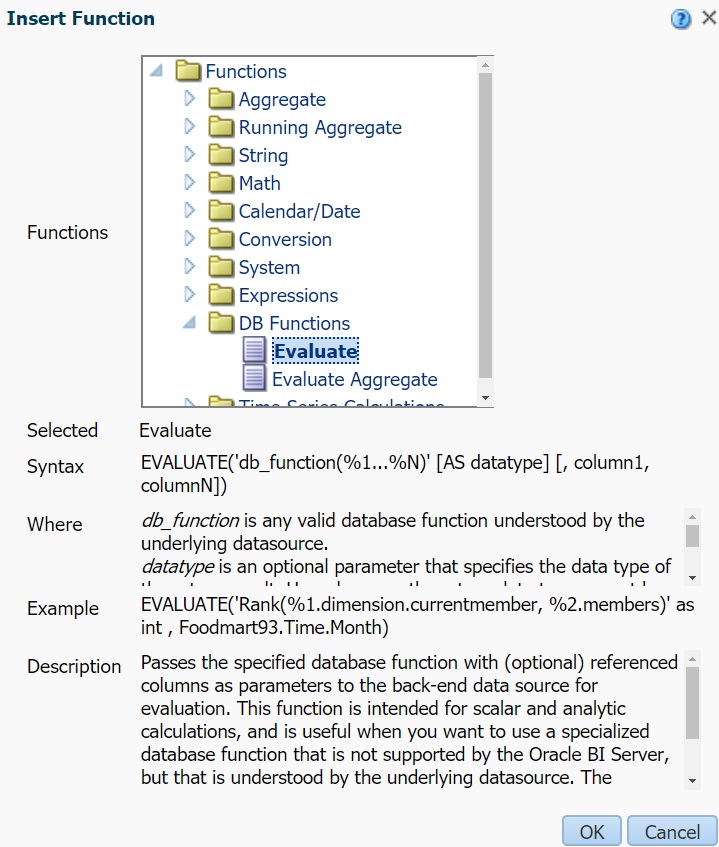
It is possible to extract the 852$c using many built-in functions, but the resulting formula is quite complex and difficult to read (more importantly, it would be difficult to maintain or reuse with another field if any adjustments to the subfield separators needed to be made):

LEFT(SUBSTRING("Holding Details"."852 MARC" FROM LOCATE('$$c', "Holding Details"."852 MARC")+4 FOR 12), LOCATE(' ', SUBSTRING("Holding Details"."852 MARC" FROM LOCATE('$$c', "Holding Details"."852 MARC")+4 FOR 12)))

What would be preferable is a single function that can identify the $$c part and extract whatever comes next, ending before the next subfield. It should also be possible to easily reuse the function for other fields or extract different subfields such as $b with a single change.

## Using Database Functions with EVALUATE

Such functions do exist, but they are not built-in functions listed in the Functions menu of Analytics and using them is a little more complicated. We will discuss how to use these functions in layers, starting with the outermost layer, the Evaluate function that allows us to use special database functions in Analytics. This can be found in the DB Functions folder:



The syntax and example that appear in this menu do not make it especially obvious how to use the function, and this is mainly because the syntax will depend on which database function we are using. A simplified version of this syntax for the function we will be looking at is:

EVALUATE('db\_function', expr1, expr2…)

Where 'db\_function' is a placeholder for the database function and expr1 is the expression we are applying the database function to (in this case "Holding Details"."852 MARC"). Within db\_function we will be able to reference expr1 using %1 and can optionally do this for other expressions if the database function supports it (e.g. %2 if expr2 is specified, etc.).

## REGEXP\_SUBSTR: Extending SUBSTRING with Regular Expressions

The special database function we will be using for this report is an extension of the SUBSTRING function called [REGEXP\_SUBSTR](https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/REGEXP_SUBSTR.html). Here is the syntax for this function:

regexp\_substr(source\_char, pattern, *position*, *occurrence*, *match\_param*, *subexpr*)

The source\_char parameter is the expression that this function will be performing a search on. For all of the examples we will be looking at, we will be entering %1 for this parameter so that the Evaluate function will use the expr1 we provided it (for this report that is "Holding Details"."852 MARC"). The pattern parameter will contain our regular expression, which we will cover in the next section.

The remaining parameters are optional and have default values if they are not specified, but we will need to specify all of them in this example so that we can change the final parameter (subexpr). The position parameter determines where the substring function begins searching (if not specified this is 1, which will be the beginning of the 852 MARC data in our example). The occurrence parameter can be used when your regular expression matches multiple strings in your data and you want to retrieve a specific one (if not specified this is 1, i.e. the first match, but supposing there were multiple $c in our 852 data and we wanted to retrieve the second one we would specify 2 for occurrence). The match\_param parameter allows you to change the default matching behavior and the available options have been defined separately in the [REGEXP\_COUNT](https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/REGEXP_COUNT.html) documentation (we do not cover this function here, but REGEXP\_COUNT extends the COUNT function to allow for counting the occurrences of patterns specified by regular expressions). In this first example we will not be changing the default matching behavior so we will enter NULL for match\_param; in other circumstances, the most common match\_param you are likely to use is 'i' which will make the search case-insensitive. The subexpr parameter can be used match a subexpression within the regular expression, which is used for what are called “capture groups” (we will explain capture groups in the regular expression section below).

Here is what the database function will look like once it is added to the Evaluate function (the “regular expression” placeholder is the final layer we will discuss next):

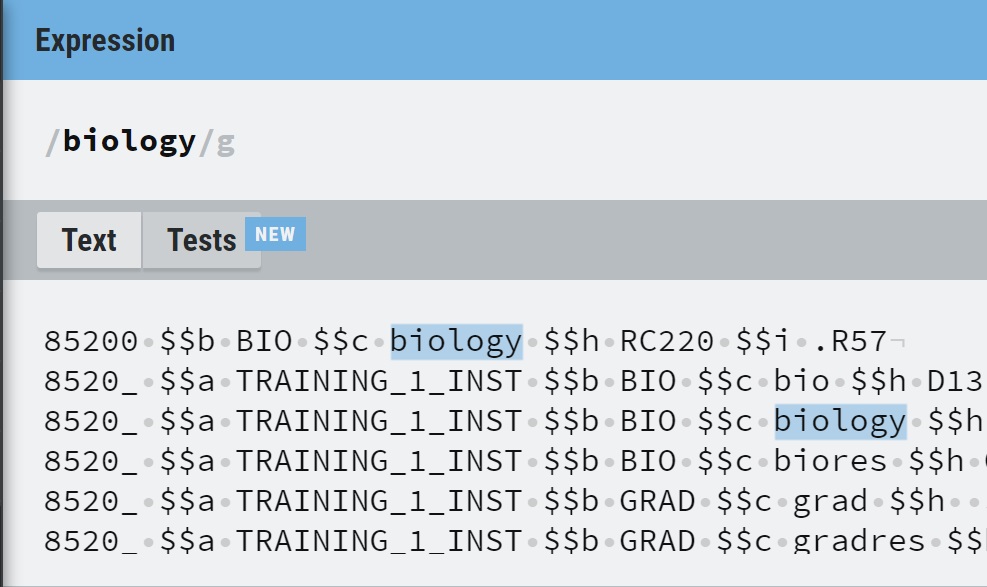
Evaluate('regexp\_substr(%1,%2,%3,%4,%5,%6)',"Holding Details"."852 MARC", 'regular expression', 1, 1, NULL, 1)

## Using Regular Expressions to Match Patterns in Data

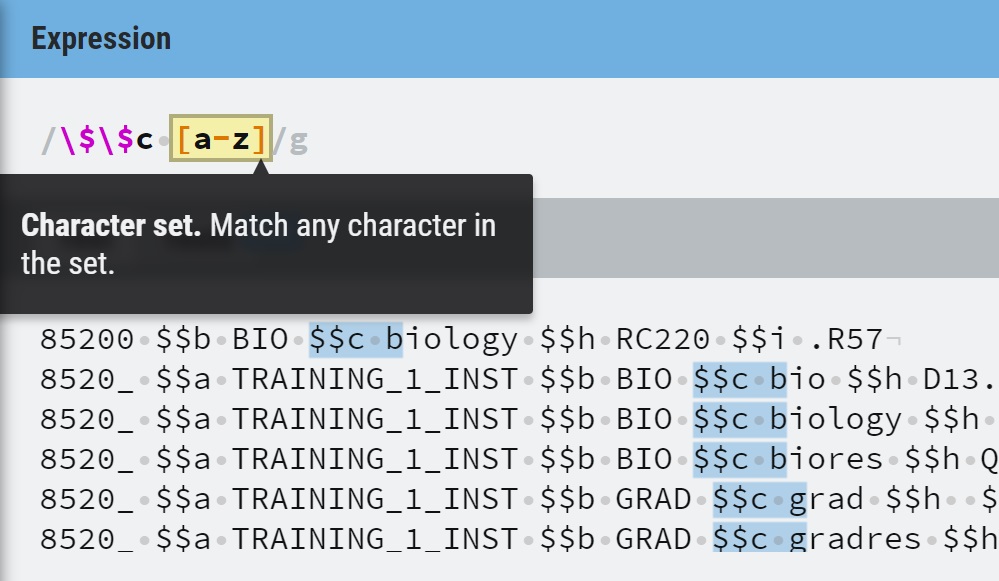
The final layer of our function is the regular expression itself. A regular expression is a sequence of characters that define a search pattern and they appear in many applications besides Analytics (in Primo VE they can be used in normalization rules for display and local fields, as well as for matching file name patterns in import profiles). They are supported by most programming languages, can be used in many command-line tools such as grep or sed, and some text editors (such as Notepad++) allow you to use them to find and replace text. It is beyond the scope of this session to provide a comprehensive overview of regular expressions, but we will explain the operators used in the examples below.

The three articles linked in [this Oracle documentation](https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Oracle-Regular-Expression-Support.html) lists the regular expression operators that are supported by these database functions. When creating a regular expression we recommend using [RegExr](https://regexr.com/) or something similar that allows you to test the regular expression on some example data, highlights the matches as you are typing, and explains the syntax step by step.

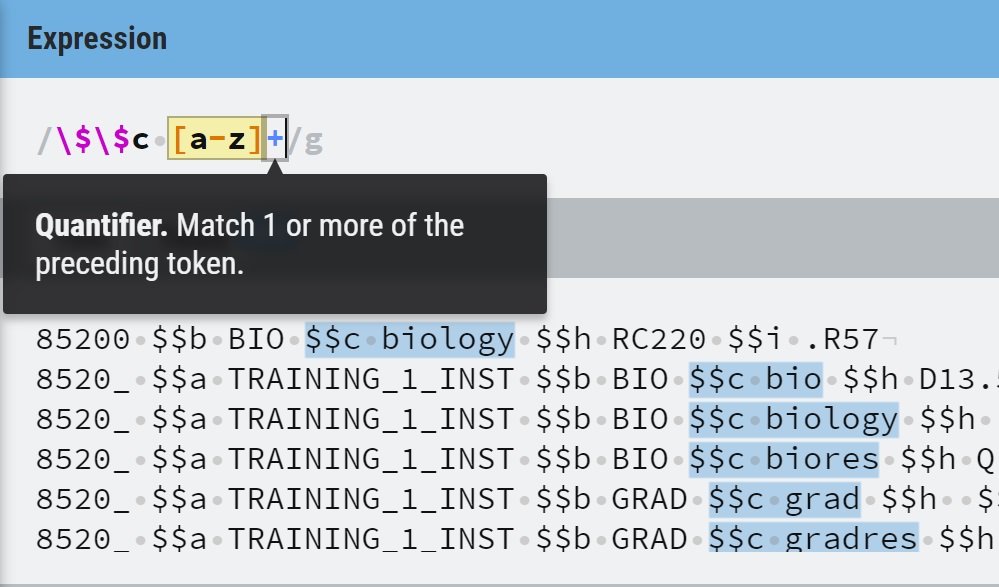
We have copied some 852 MARC values from the Results tab into RegExr to develop the regular expression. This first regular expression is extremely unimpressive and would be possible to match using built-in function, however we wanted to note at the beginning that it is possible to match specific characters in regular expressions without any special syntax:



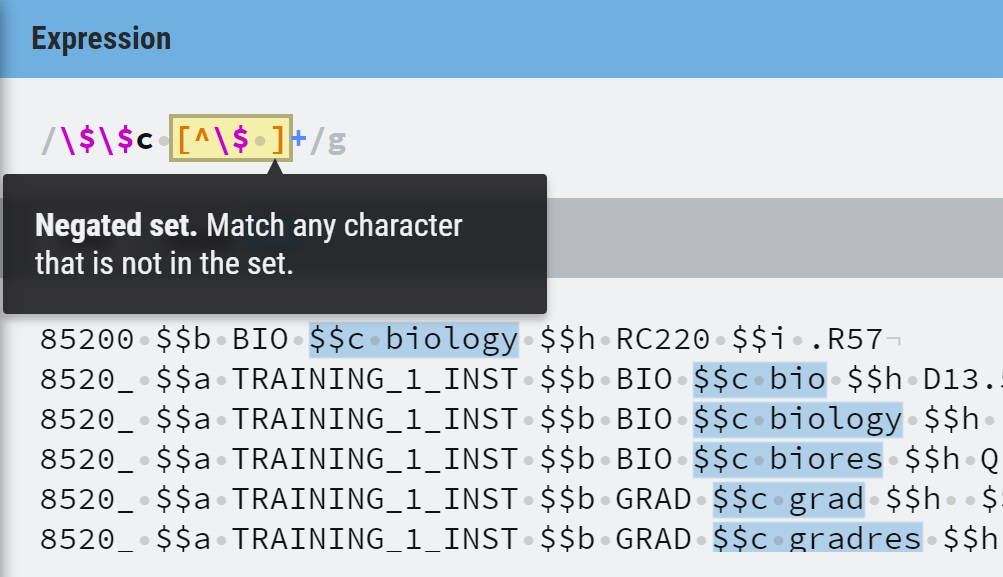
Many punctuation characters and symbols are special operators in regular expressions, for example $ matches the end of a line. Thus, if you want to match those characters you need to “escape” them using a backslash (e.g. \$ matches a dollar sign instead of the end of the line. Backslash itself must be also be escaped, e.g. \\ matches “\” instead of escaping the next character). Square brackets are used to define a set of characters that you want to match and can be entered individually (e.g. [abc] matches “a”, “b”, or “c”) or a dash can be used to specify a range. In the example below, [a-z] matches any lower-case alphabetical character:



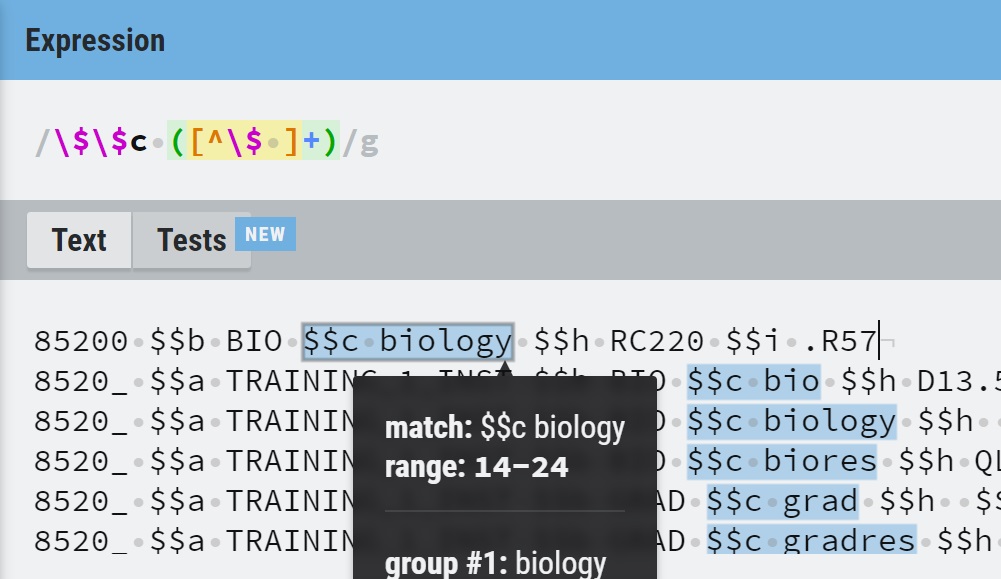
Quantifiers let you specify how many of the preceding pattern you would like to match. For example, [a-z] matches only one alphabetical character but [a-z]+ matches one or more of these characters (? would match zero to one, \* would match zero to many, and curly brackets can be used to specify a range e.g. {2,5} would match between two and five of these characters):



Adding a caret (^) after the opening bracket in a set will negate it, meaning that the regular expression will match any characters except those specified:



Finally, parentheses can be used to define a capture group. This allows us to match text that may be around the data we want but that we do not want to include in the data we are extracting. In this case, we need to match $$c to identify the correct subfield but we do not want “$$c “ in the results so we can use a capture group to exclude it:

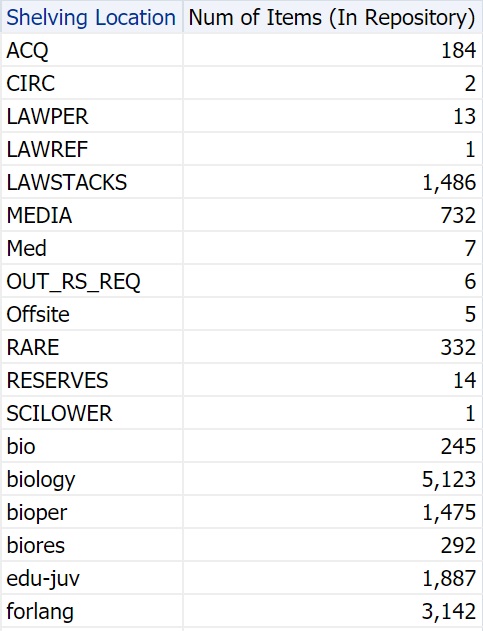


## Putting It All Together

At last, here is our completed function:

Evaluate('regexp\_substr(%1,%2,%3,%4,%5,%6)',"Holding Details"."852 MARC", '\$\$c ([^\$ ]+)', 1, 1, NULL, 1)

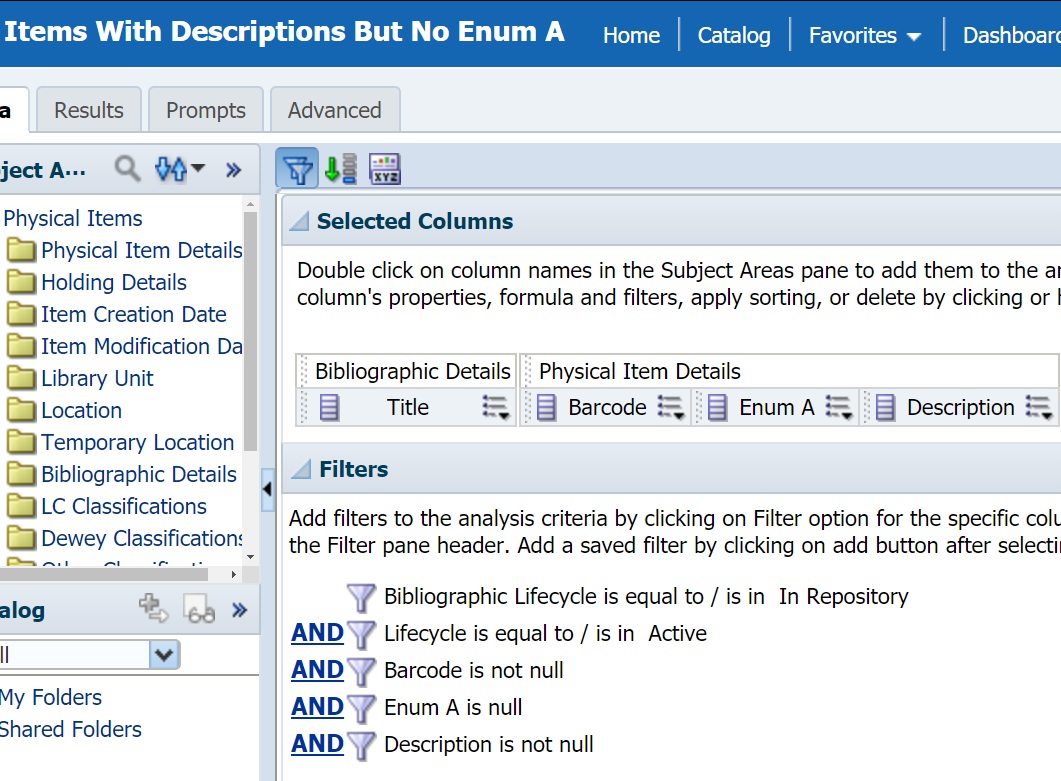
And the results. With the other 852 MARC data removed, we now see item totals for each 852$c shelving location as intended:



# **Bonus: Using Advanced Functions to Work With Messy Data**

## Extracting Volume Data From Messy Descriptions

With the extended functionality of REGEXP\_SUBSTR we can now work with much messier data in Analytics. Consider the “Items With Descriptions But No Enum A” report below, we are looking for items that do not have any Enum A value but might have volume information somewhere in the Description field:



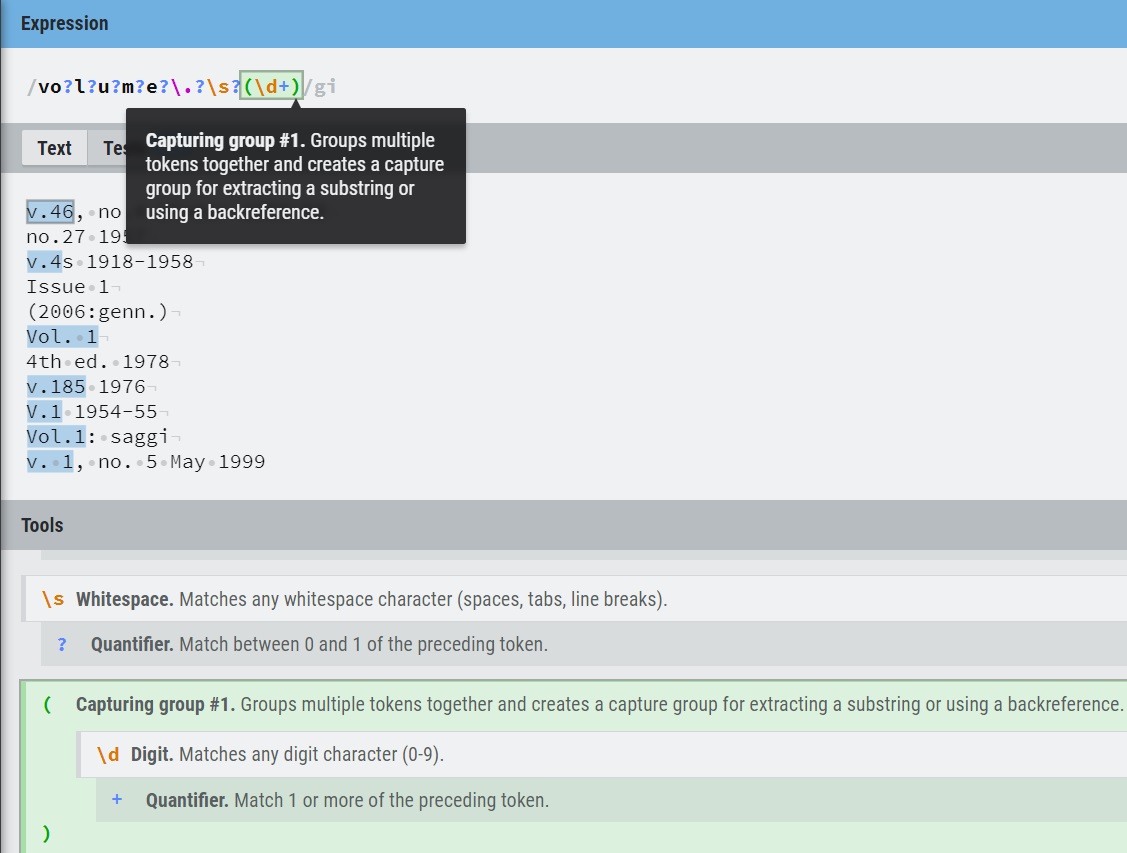
Unfortunately, the data in the Description field is a bit of a mess. Volume information does appear, but so does publication years, issues, and other data. Meanwhile the volume information is not very consistent: “v.”, “v. “, “vol.”, “Vol.”, “Volume”, and other variations were all found in the full results:



To determine what regular expression to use for this data, we can again rely on a tool like RegExr to test our patterns on some example data from the report. Since we know that we want to match both upper and lower case variations like “vol” and “Vol”, we can enable the case insensitive flag from the Flags menu before starting and later use 'i' in match\_param to enable case-insensitive searching (this is not necessary as we could use sets for each character like [Vv] but does make our regular expression smaller and easier to read):



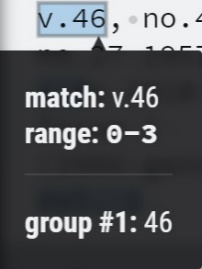
Copy some of the Description values from the report into the Text section so that we have some data to test with. Try to include a variety of volume formats as well as some examples that do not have any volume information at all (and thus should not display any result in the report) to make sure that the regular expression is not overmatching:



After some trial and error, we found that this regular expression worked well for our data, depending on your data you might need to make some adjustments:

vo?l?u?m?e?\.?\s?(\d+)

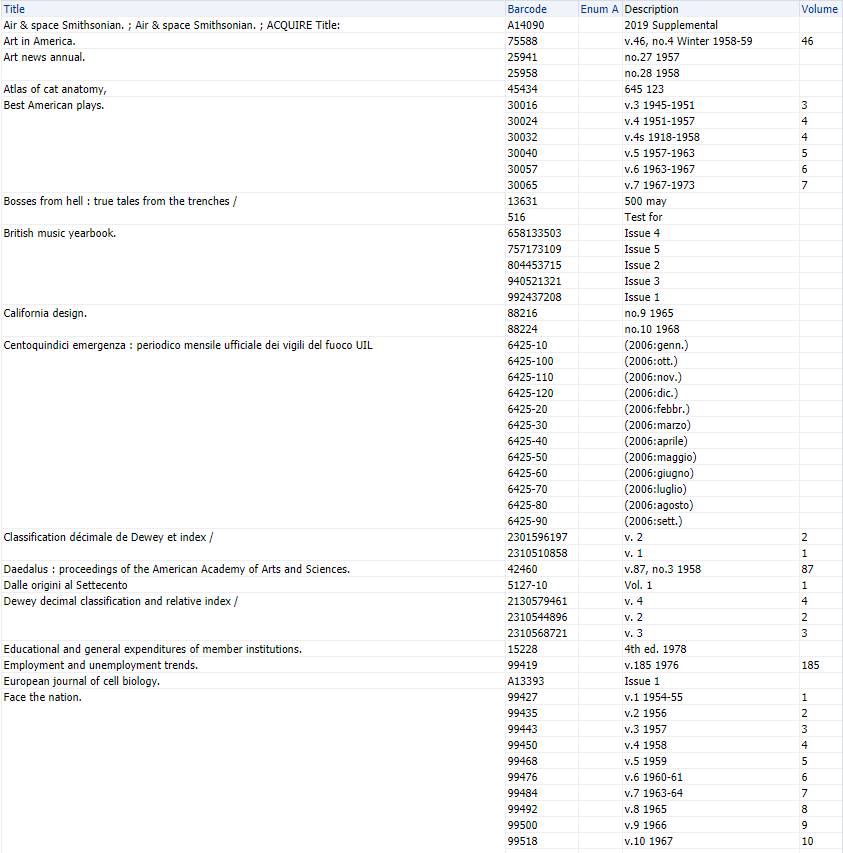
The question marks are quantifiers that mean the character that comes before it is optional, so “v”, “vo”, “vol”, etc. will all match if the rest of the pattern matches. Some of the data had periods and/or spaces after the volume designator, so these were included with question marks as well (\s matches not only a space but other whitespace characters as well such as tabs). \d+ is used to match one to many digits and this is put in parentheses to form a capture group, which is the only part of the matched data that we want to return. You can see these groups in RegExr by hovering over the matched data, in this case even though “v.46” is matched if we use subexpr to return only the first capture group the result will be “46”:



Here is our function. Note that it was possible to almost completely reuse the previous REGEXP\_SUBSTR function, the only parts that have changed are the field, regular expression and the match\_param was changed from NULL to 'i':

EVALUATE('regexp\_substr(%1, %2, %3, %4, %5, %6)', "Physical Item Details"."Description", 'vo?l?u?m?e?\.?\s?(\d+)', 1, 1, 'i', 1)

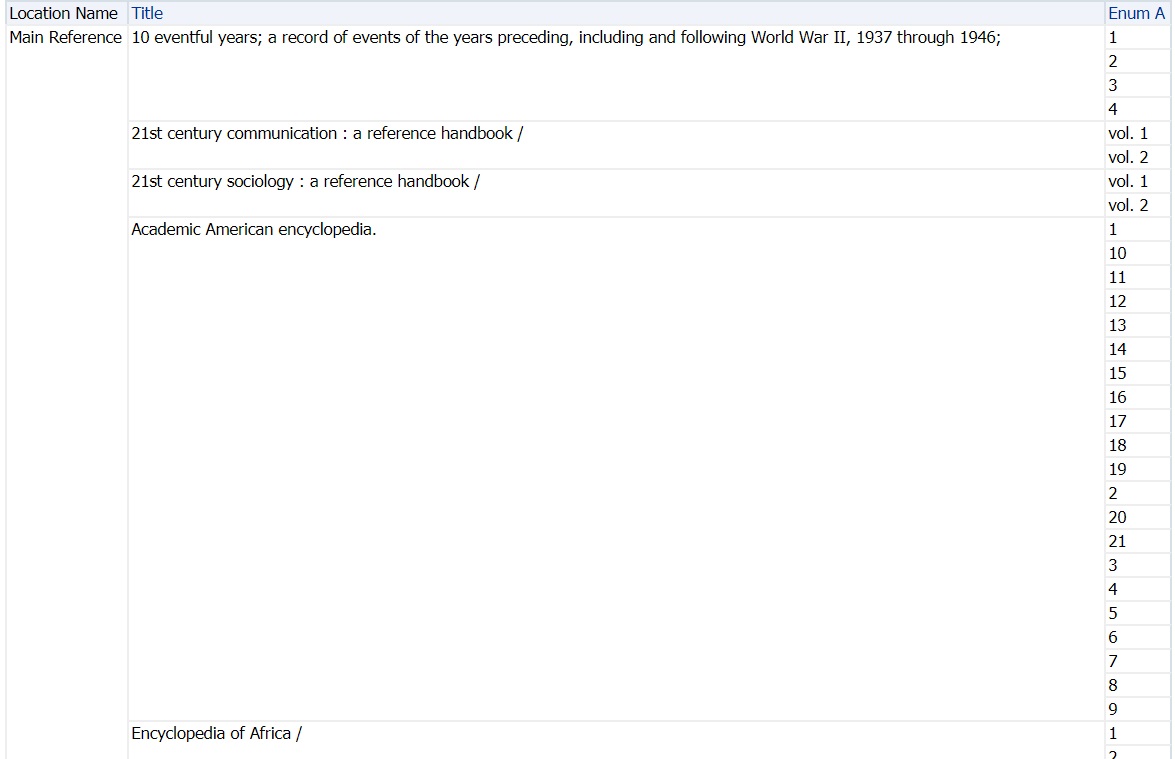
Here is the result of this function in the report (an unmodified copy of the Description field was kept in the report for comparison):



Matching was successful on all the data in the report in our training environment, although there were some examples of supplementary volumes like “v.4s” and “Suppl. to v.62” that returned only the volume number. Depending on how you treat supplemental volumes this might be desired, or you might want to adjust the regular expression to not match these or include some indication that it is a supplementary volume. In your environment it is possible you will encounter outliers where this regular expression did not match the data as expected. In some cases, it would be reasonable to modify the regular expression a bit to also match these outliers, but sometimes making these adjustments can result in a significantly more complicated regular expressions, which would start becoming difficult to maintain. At some point you must ask yourself whether it makes more sense to clean up the data itself rather than continue adding exceptions to your regular expression.

## REGEXP\_REPLACE: The Regular Expression Replace Function

Recall the “Items With Volumes By Location” report where we wanted to sort the volumes in Enum A, but they were sorting alphanumerically and inconsistent data such as “1” and “vol. 1” was preventing us from just using the CAST function to sort it numerically:



REGEXP\_SUBSTR would solve this issue neatly and we would not need to use the optional parameters to do so:

EVALUATE('regexp\_substr(%1, %2)', "Physical Item Details"."Enum A", '\d+')

However, another option to consider would be the [REGEXP\_REPLACE](https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/REGEXP_REPLACE.html) function, which extends the REPLACE function with regular expression matching (similar to how REGEXP\_SUBSTR extends SUBSTRING). Here is the syntax for this function:

regexp\_replace(source\_char, pattern, replace\_string, *position*, *occurrence*, *match\_param*)

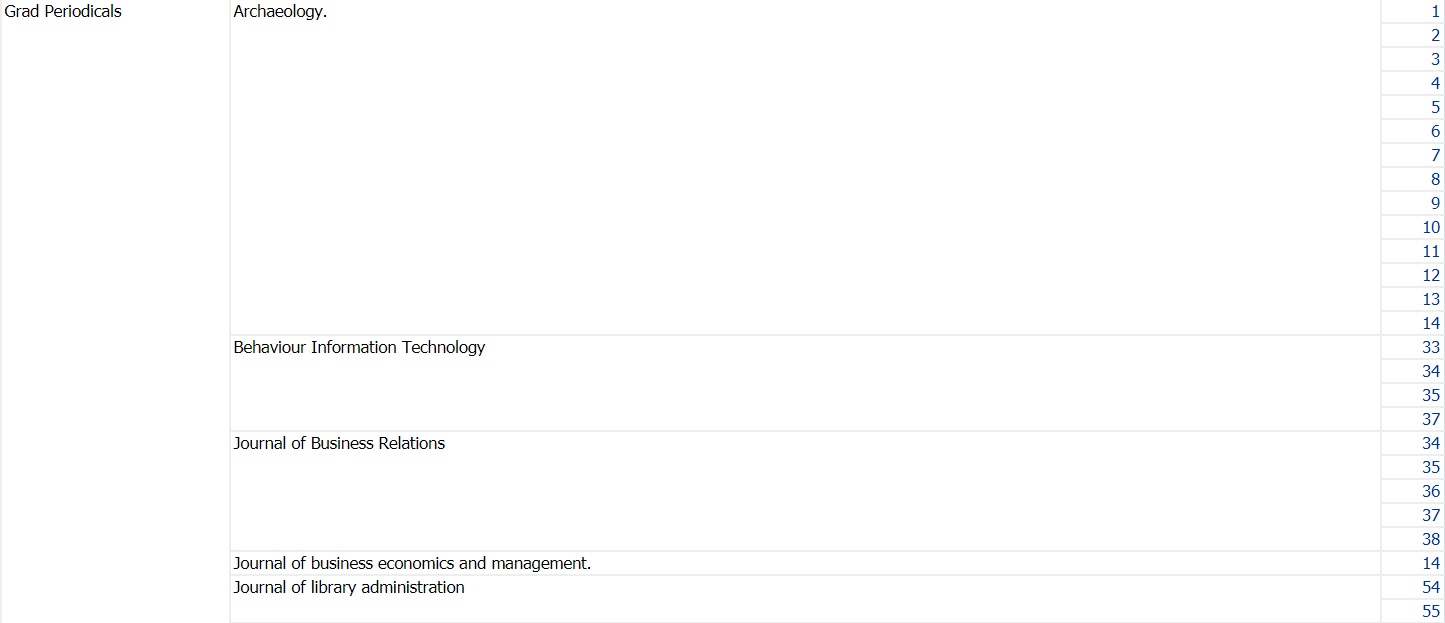
Just as with REGEXP\_SUBSTR, the source\_char is an expression that serves as the search value and we can use %1 as a reference for the field in the Evaluate function (Enum A). The pattern is where our regular expression will go and the replace\_string will be what replaces the matched text (in this example we will replace it with an empty string as before). The rest of the parameters are optional and we do not need to use them in this example, but position is where the search starts (default 1, the beginning of the Enum A data in this case), occurrence can be used when the regular expression matches multiple times in the data (default 1), and the match\_param can be used to change the default matching behavior (e.g. if you wanted a case-insensitive replacement you could use 'i' as before).

There is a regular expression operator for matching non-digit characters \D which is supported by Analytics (however [^0-9]+ would work as well). Here is how we would use the function to remove all non-digit characters from Enum A:

Evaluate('regexp\_replace(%1, %2, %3)',"Physical Item Details"."Enum A", '\D', '')

It is now safe to CAST this function and know that regardless of the data in Enum A it will be converted to a number and sorted numerically without displaying an error:

CAST(Evaluate('regexp\_replace(%1, %2, %3)',"Physical Item Details"."Enum A", '\D', '') AS INT)



We have used this function in the “completed” version of this report under the Shared Folders/Community/Reports/Shared Reports/Knowledge Days 2020/Resource Management/Completed Exercises folder. Since we did not know what data might appear in the Enum A fields of all of your Alma environments, we could not rely on any number of REPLACE functions to prevent an error from appearing when using CAST, but with this function it should not be possible for any non-digit data to be sent to CAST.