

alteryx

24 BEST PRACTICES

BY THE COMMUNITY
TAKEAWAYS FROM THE COPENHAGEN BASED
ALTERYX SUPER USER GROUP

TABLE OF CONTENTS

The Basics	1
1 Remove All Browse Tools	1
2 Correct Data Types	5
3 Documentation Using Descriptive Titles	9
4 Documentation Using Tool Annotations	11
5 Documentation Using Comment Tool	15
6 Documentation Using Tool Containers	19
7 Remove Duplicates from Joins	23
8 No Errors, Warnings, or Conversion Errors	27
9 Make Use of Macros	31
Structure	35
10 Nice Folder Structure	35
11 No Automatic XML parsing	39
Performance	41
12 Central Data Source – Load Data Once	41
13 Disable All Outputs	45
14 Use of Containers for Partial Execution	47
15 Using the Performance Profiler	51
16 Investigate Data Using Subsamples	55

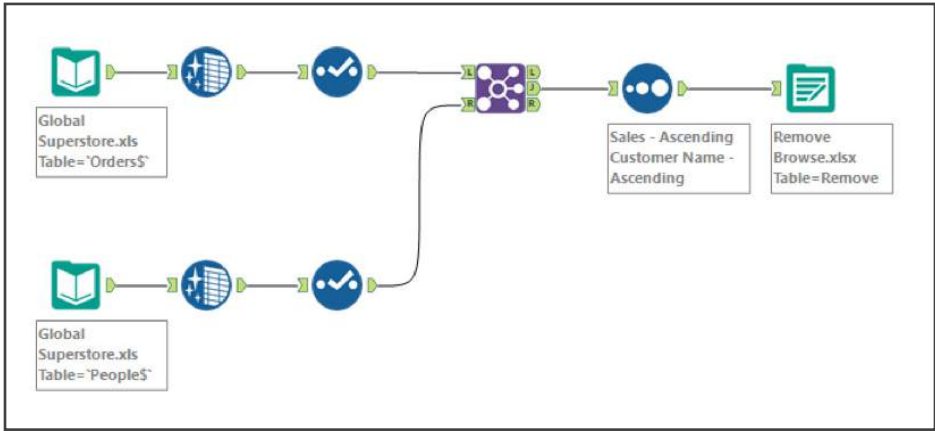
17	Using in-DB Tools	59
	When Finished	63
18	Put a Timestamp on Your Data Source	63
19	Using the List Runner to Control Execution	67
20	Clean and Well-Constructed Workflow	71
21	Relative Dependencies	75
	Analytical	79
22	Validation Checks and Testing	79
23	Calculations	83
24	Are You Getting the Right Numbers?	85

1

Remove All Browse Tools

➤ THE BASICS

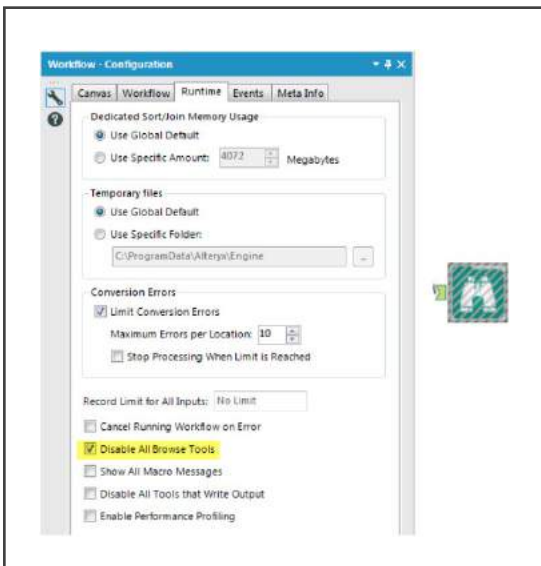
WORKFLOW – AFTER



HINT

There is an option to simply disable all Browse tools, if needed. This setting can be found here:

Click the Canvas → Runtime → Disable All Browse Tools



2

Correct Data Types

➤ THE BASICS

DESCRIPTION

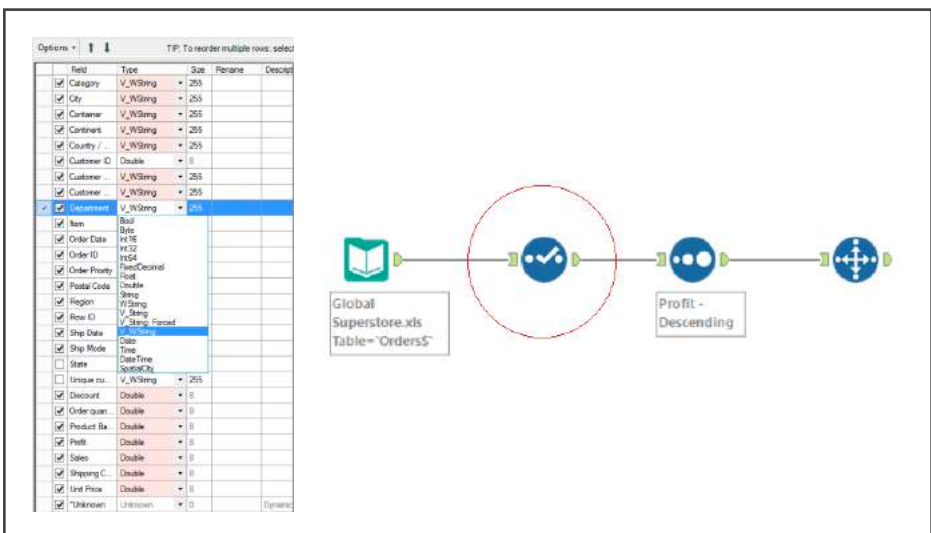
A best practice is to ensure that you are working with the correct data types and meaningful variable names as early in the workflow as possible. This can be done by using the *Select* tool in the beginning of your workflow.

Select Tool



Using this best practice will ensure that you have a good overview of your variables, and will help you avoid wasting time, e.g. by attempting to do numeric operations on strings or using incorrect data types. Some software requires specific data types depending on the data source; for example, exporting TDE-files to Tableau with unicode characters requires strings to be of the type *V_WString*. By using this best practice, you can avoid spending precious time on easily-fixed problems.

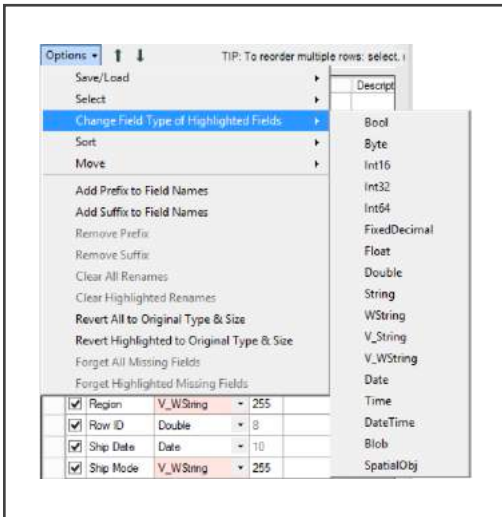
WORKFLOW AND CONFIGURATION OF SELECT TOOL



HINT

If you want to change the data type of multiple variables of the same type simultaneously, you can do so through the following steps:

Highlight the Variables of Interest → Options → Change Field Types of Highlighted Fields



You can also select or deselect multiple variables by following the steps below:

Highlight the Variables of Interest → Options → Select → (De)Select All Highlighted Variables

3

Documentation Using Descriptive Titles

➤ THE BASICS

DESCRIPTION






It is always a best practice to carefully document your workflows.

There are various reasons to do this; handing over your workflow(s) to a colleague or a client, or returning to your previous work, is much easier with adequate documentation.

When you are working with multiple workflows, it is important to name them with descriptive titles. If your workflows are dependent, it can be very useful to enumerate them as well.

Descriptive, numbered titles will make it easier to decode the workflow-dependency and to obtain an idea of what is going on, without needing to open each of the individual workflows.

DESCRIPTIVE TITLES AND ENUMERATION

-  1. Sales Cleansing and yxdb
-  2. Sales Analysis
-  3. Customer Cleansing and yxdb
-  4. Customer Analysis
-  5. Output to Tableau

4

Documentation Using Tool Annotations

➤ THE BASICS

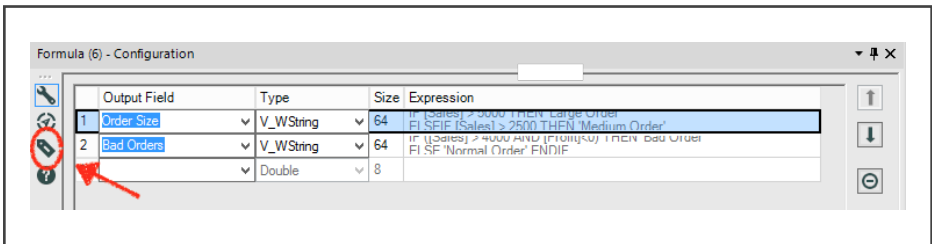
DESCRIPTION

Another aspect of good documentation involves keeping track of what is going on inside your workflow. There are several ways to keep your workflow clean and understandable. The first way you can do this is by making use of Tool Annotations, which are short and brief descriptions of what is happening within the different tools.

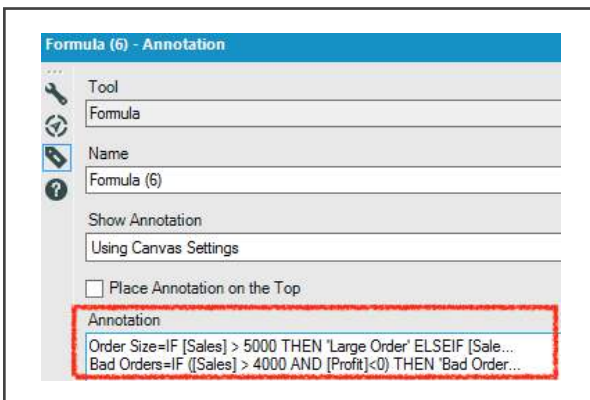
Make sure to write short and precise comments when using tool annotations. These annotations are very helpful, as they give an indication of the purpose of the tool. Additionally, the annotation will stick to the tool even though it might be moved elsewhere. You can edit existing tool annotations by clicking the tool and clicking the “Label” icon.

WHERE DO I FIND ANNOTATIONS?

Click the tool that you want to annotate and click the “Label” icon, as shown below:

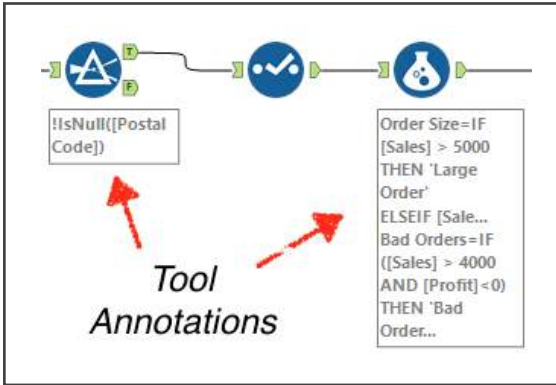


You can edit annotations in the Annotation-box, as shown below:

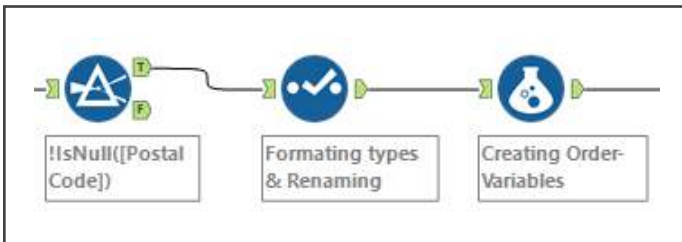


You can specify the placement of the annotation by clicking the small checkbox, "Place Annotation on the Top".

UNTIDY TOOL ANNOTATION – NOT BEST PRACTICE



CONDENSED TOOL ANNOTATION – BEST PRACTICE



5

Documentation Using Comment Tool

➤ THE BASICS

DESCRIPTION

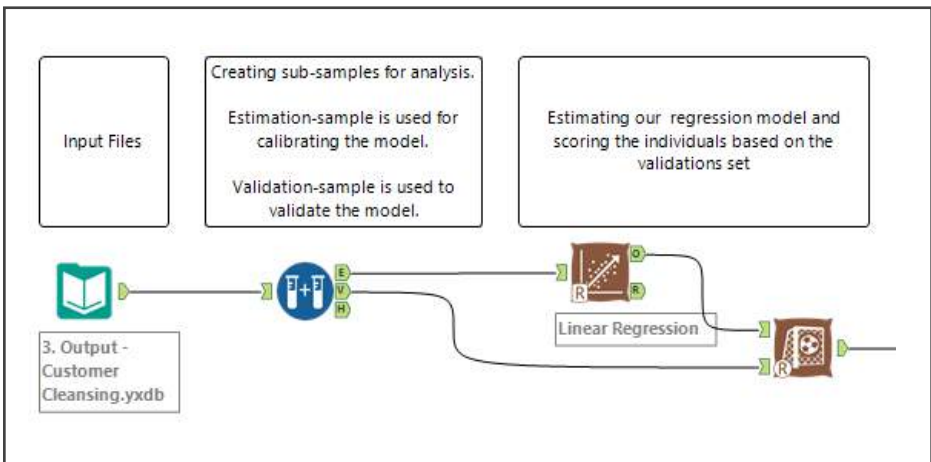
Tool annotations tend to get messy if you can't concisely explain all of your steps in only a few words. In this case, the Comment tool might be more useful. The *Comment* tool works as a text-box, and can be moved anywhere. Note that unlike annotations, text-boxes will not stay fixed to the tool.

The Comment tool also has additional formatting options that allow you to change shape, colour, alignment, font styles, and background images. You can find the Comment tool in the *Documentation* pane.

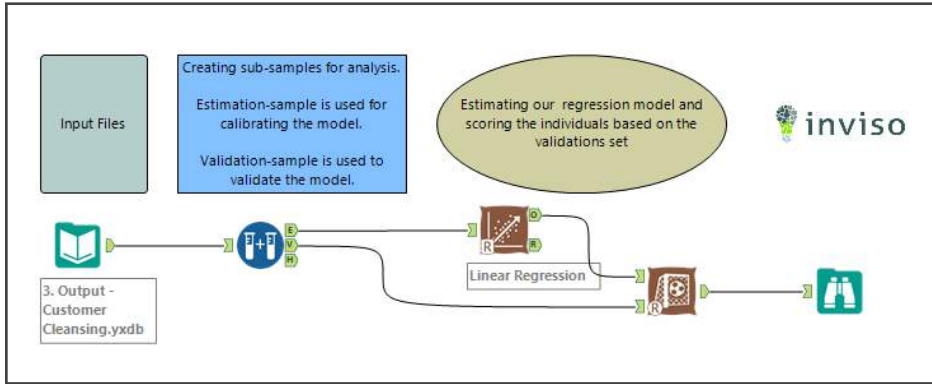
Comment Tool



COMMENT TOOL – BASICS



COMMENT TOOL – FORMATTING POSSIBILITIES



6

Documentation Using Tool Containers

➤ THE BASICS

DESCRIPTION

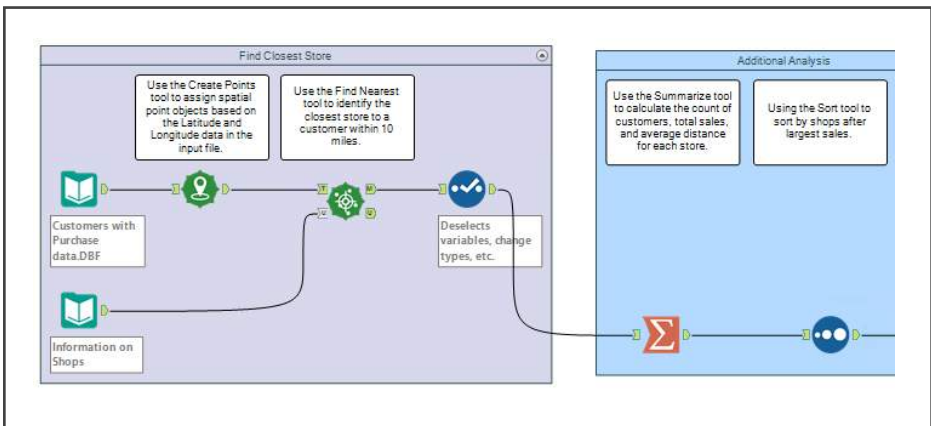
Finally, you can use *Tool Containers* to group parts of your workflow into stages. This will help you keep a clear overview of which tools are used together in each step of your analysis.

You can add names and colours to Tool Containers, which will strengthen their grouping and make the workflow easier to understand. Additionally, the layout inside of a Tool Container will be kept, even if it is moved around. For this reason, it can be particularly beneficial to use Tool Containers in conjunction with Comment tools.

Tool Container Tool



TOOL CONTAINERS



HINT

You can name tools instead of simply numbering them. This can potentially give you a clearer indication of where to look when you are debugging, because the tool name will show up in the Results pane when you run the flow.

The image shows two parts of the Alteryx interface. The top part is the configuration pane for a 'Formula (6) - Annotation' tool. It has fields for 'Tool' (Formula), 'Name' (Formula (6)), 'Show Annotation' (Using Canvas Settings), and a help icon. A red dashed box highlights the 'Name' field. The bottom part is the 'Results - Workflow - Messages' pane. It shows a summary of 1 Error, 0 Conv Errors, 0 Warnings, and 3 Messages. Below this, a list of messages is shown. The first message is 'Designer x64 Started running Y:\FJ\Alteryx Best Practises'. The second message is 'Designer x64 The Designer x64 reported: Allocating requested d'. The third message is 'Output Data (3) No Records were written because tool has been dis'. The fourth message is 'Formula (6) The formula "Order Size" resulted in a string !'. A red arrow points from the 'Name' field in the top pane to the 'Formula (6)' message in the bottom pane. A red dashed box highlights the 'Formula (6)' message.

If you are using Tool Containers, it can be useful to set margin = 0 and transparency = 50 %.

7

Remove Duplicates from Joins

➤ THE BASICS

DESCRIPTION

When you use the *Join* tool to join two data sources using specific fields that have identical names, Alteryx will assign a prefix to the variables from the right-input by default. These values are duplicates, and in most cases they only lead to slower processing and more confusion for you as a user.

Join Tool

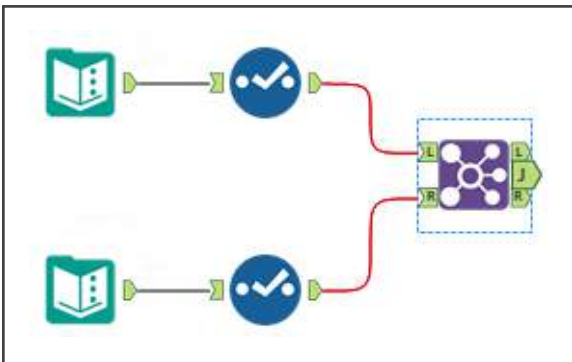


This scenario is illustrated in the figures below. In this case, a join was made on *Category*, which led to a duplicate variable called *Right_Category*.

It is a best practice to make sure to remove all duplicate variables created in a join. This can be done in two ways. First, you can simply click to remove the checkmarks beside the duplicate (right) variables. Second, if you have many duplicates you can deselect them with the following steps:

Join Configuration → *Options* → *Deselect Duplicate Fields*

JOIN ON CATEGORY



8

**No Errors,
Warnings, or
Conversion Errors**

➤ THE BASICS

DESCRIPTION

When constructing a workflow in Alteryx, sooner or later you will experience *Errors*, *Conversion Errors*, and *Warnings*. This is a wonderful feature that alerts you every time that something is wrong. So, why not make the best use of it?

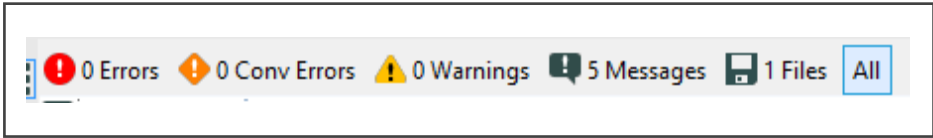
It is a best practice to deal with Errors and Warnings as soon as they occur. It's in your best interest to keep the *Results* pane clean, as shown in the figure below (*Desired View in the Results pane*). By keeping the Results pane clean, you will ensure that you have not made any mistakes in your logic, and it will be much easier to notice when something might be wrong the next time you run the flow.

Alteryx will indicate Errors by adding an exclamation mark beneath the tool where the error occurred. Conversion Errors and Warnings can be identified by looking at the *Tool Reference*, which is shown in parentheses after the tool name in the Results pane.

ERRORS AND WARNINGS IN WORKFLOW

The screenshot displays an Alteryx workflow and its corresponding Results pane. The workflow consists of three input tools (represented by book icons) feeding into a Union All tool, which then feeds into a Join on Category tool, and finally a Formula tool. The Results pane shows a summary of 1 Error, 0 Conversion Errors, and 3 Warnings. The error message is: "ParseError at char(0): formula: invalid type in subtraction operator (Expression #1)". The warning messages are: "The field 'Actual Sales' is not contained in the record." and "The field 'Actual Sales' is not present in all inputs." The Results pane also shows a list of records output by the tools, including Test Input, Select, Union, Formula, and Join.

DESIRED VIEW IN THE RESULTS PANE

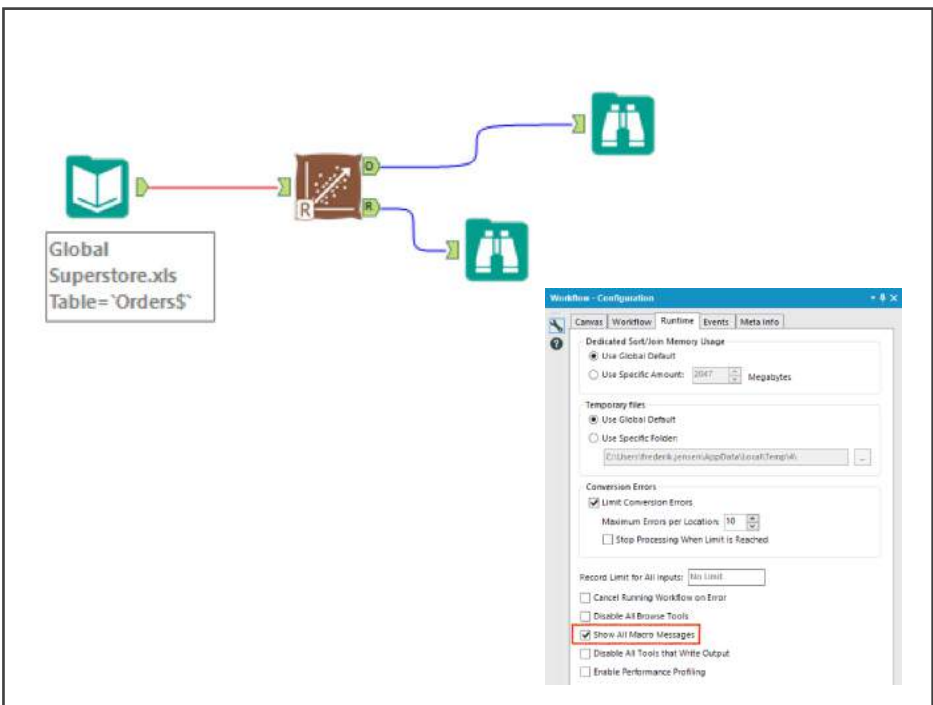


HINT

If you are working with macros (e.g. predictive tools), you might have experienced problems with debugging your workflow. This is because Alteryx only shows Errors originating from macros by default, and does not show Conversion Errors or Warnings.

Therefore, it can be helpful to make all messages visible, even within the macro. This can be done by following these steps:

Click the Canvas → Workflow—Configuration → Runtime → Show All Macro Messages



9

Make Use of Macros

➤ THE BASICS

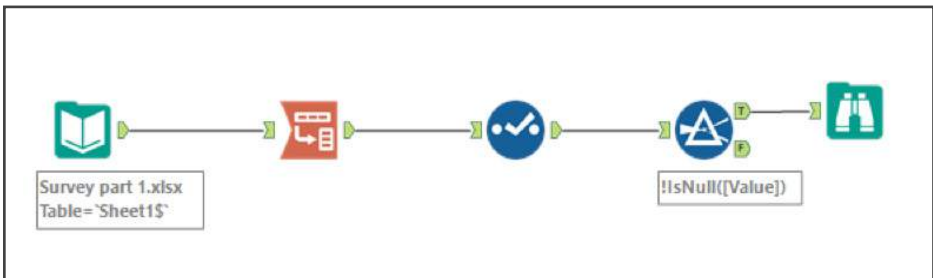
DESCRIPTION

If you have multiple workflows or multiple data-streams where you need to apply the same logic or an identical approach, it is a best practice to make use of *Macros*. You should start using macros in Alteryx if you do not already.

An Alteryx Macro Workflow is simply a workflow that has the flexibility to run as a single tool within another workflow. There are three types of macros: *Batch-*, *Iterative-*, and *Location Optimizer-Macros*. If you want to learn more about the different macro types, we suggest that you start building them and seek more information within the Alteryx Community.

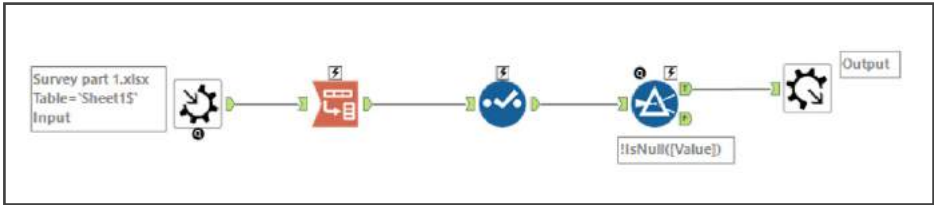
EXAMPLE OF CONSTRUCTING A BATCH-MACRO – LOGIC

Here is a small example of a batch-macro. Imagine that you receive multiple files that are more or less identical. Because you need to do the same exercise repeatedly, you build a flow like the one below to handle this process.



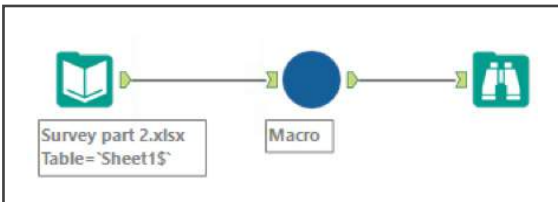
Now you want to be capable of easily applying the same logic elsewhere, and you also want to have a better overview with fewer tools. This can be done by using the *Macro Input* tool and the *Macro Output* tool.

EXAMPLE OF CONSTRUCTING A BATCH-MACRO – INPUT & OUTPUT



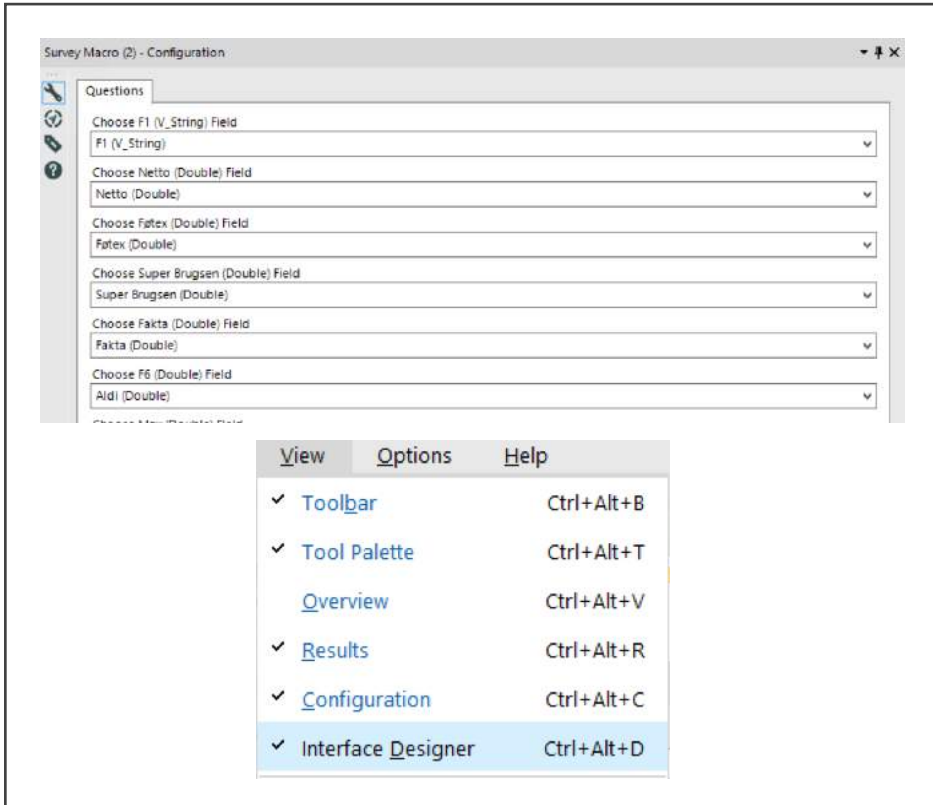
Here, the initial file has been chosen as input in the Macro Input tool. At this point, you can save the above as a macro (.yxmc), thereby creating the possibility of reusing the same logic repeatedly, while at the same time having a better overview of your workflow. By *right-clicking* the canvas and choosing *Insert*, you can now choose *Macro* to insert the macro.

EXAMPLE OF CONSTRUCTING A BATCH-MACRO – USING THE MACRO



The macro will have its own configuration, which creates a higher degree of flexibility for the end user. It can be formatted and designed within the Interface Designer. The *Interface Designer* can be found through the following steps:

Go to View → Interface Designer



HINT

You can create a *Macro Repository*, where Alteryx will look in the specified directories for Macros and display them in the Tool Palette for building workflows.



10

Nice Folder Structure

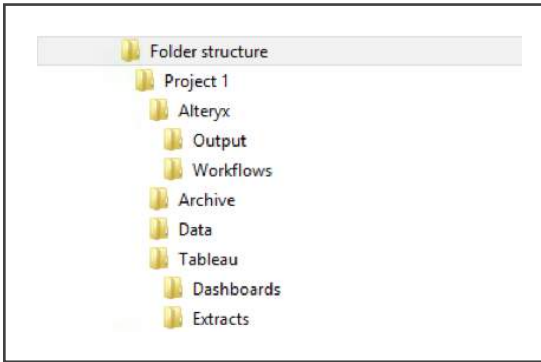
➤ STRUCTURE

DESCRIPTION

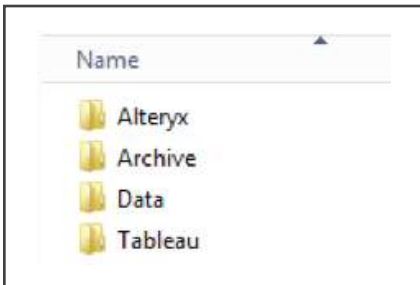
Make sure to maintain a good folder structure throughout your work. Having a logical folder structure for your data sources, workflows, output, etc. provides you a great overview of your work. Moreover, it will be make it much easier for you to hand over your work to others.

Below is an example of one way you could potentially organize your folder structure. This structure separates *Project 1* files into *Alteryx*, *Archive*, *Data* and *Tableau* folders. The raw input data goes to the *Data* folder. All Alteryx workflows and output go into their respective *Alteryx* sub-folders. Similarly, the Tableau Data Extracts and Tableau Dashboards are stored in the matching *Tableau* sub-folder. Lastly, an *Archive* folder has been created to store old workflows etc. that might be of interest in the future or that can work as backup.

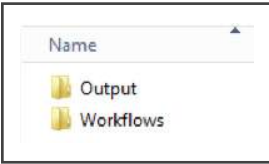
FOLDER STRUCTURE – OVERVIEW



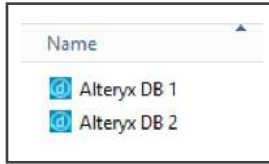
FOLDER LEVEL 1 – PROJECT 1



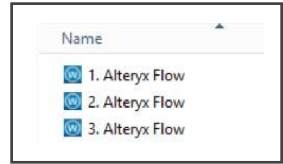
FOLDER LEVEL 2 – ALTERYX



OUTPUT



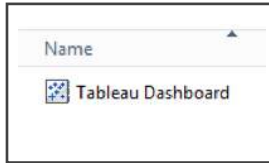
WORKFLOWS



FOLDER LEVEL 2 – TABLEAU



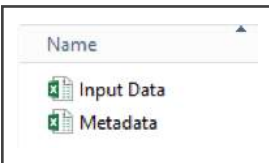
DASHBOARDS



EXTRACTS



FOLDER LEVEL 2 – DATA



11

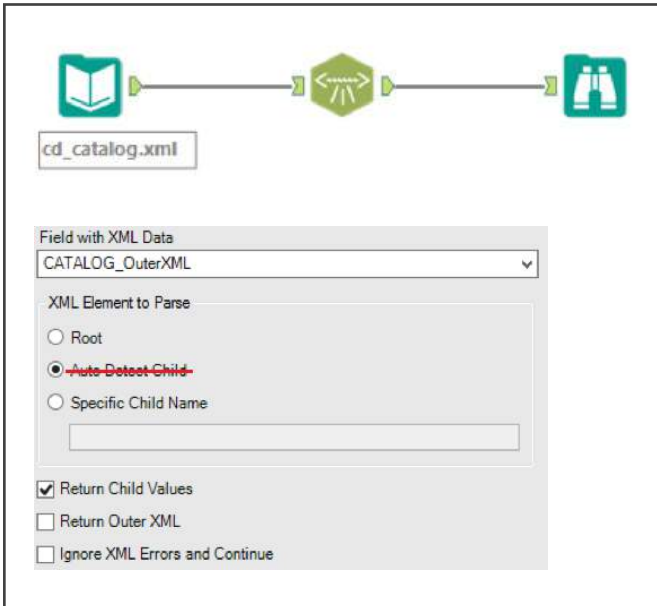
No Automatic XML Parsing

➤ STRUCTURE

DESCRIPTION

It is not a good practice to use automatic XML-parsing. This is due to the possibility that the XML-code could change and thereby break your workflow. Therefore, you should never use *Auto Detect Child* when you use the *XML Parse* tool in production.

XML Parse Tool



12

**Central Data
Source –
Load Data Once**

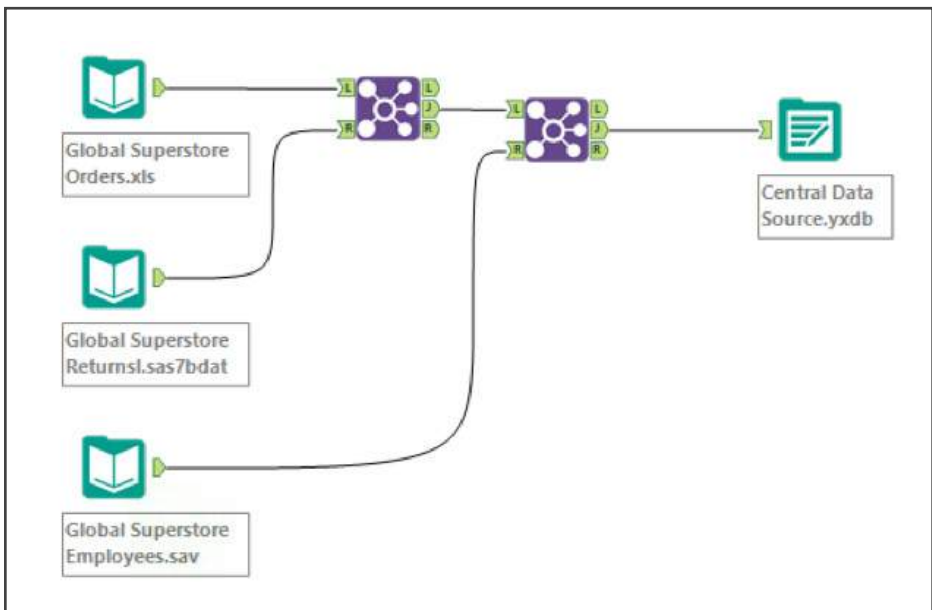
➤ PERFORMANCE

DESCRIPTION

When working with multiple data sources that need to be prepared before use, it is a best practice to separate the process of data preparation and analysis. This can be done by making one workflow that prepares and combines all your data into one central data source, which can then be used as input in your analytical workflow. In this way, you only have to load data once.

It is recommended that you combine and save your data sources in an *Alteryx Database (.yxdb)* format when working with Alteryx. There are multiple reasons for this. The .yxdb format is the most efficient file format for reading and writing in Alteryx. This is because the .yxdb format uses the exact same field types, structures, and formats as Alteryx does internally. Additionally, there is no record limit (only a record size limit of 2GB).

COMBINING DIFFERENT DATA SOURCES INTO CENTRAL DB



CENTRAL DATA SOURCE – ALTERYX DATABASE (.YXDB)



13

Disable All Outputs

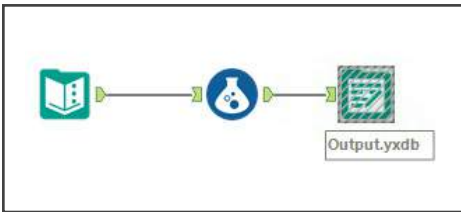
➤ PERFORMANCE

DESCRIPTION

As you are developing a workflow, you may sometimes need to make a few changes or test a new process without overwriting important output files that have already been created. One way of doing this is to delete the connection to the Output tool during your testing, and then reconnect when you are done. Another option is to *disable* the outputs.

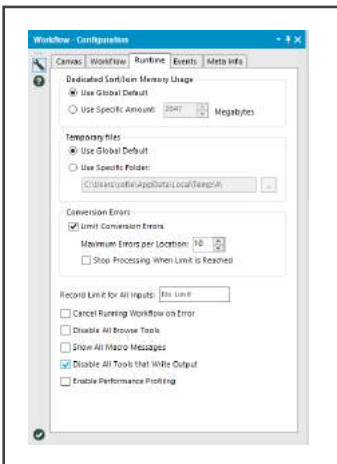
To do this, click on a blank space of your canvas and choose *Configuration*, then click on the *Runtime* tab. At the bottom, you can choose the option to *disable all tools that write output*.

WORKFLOW WITH DISABLED OUTPUT



HOW TO DISABLE ALL OUTPUT

Click the Canvas → Workflow—Configuration → Runtime → Disable All Tools That Write Output



14

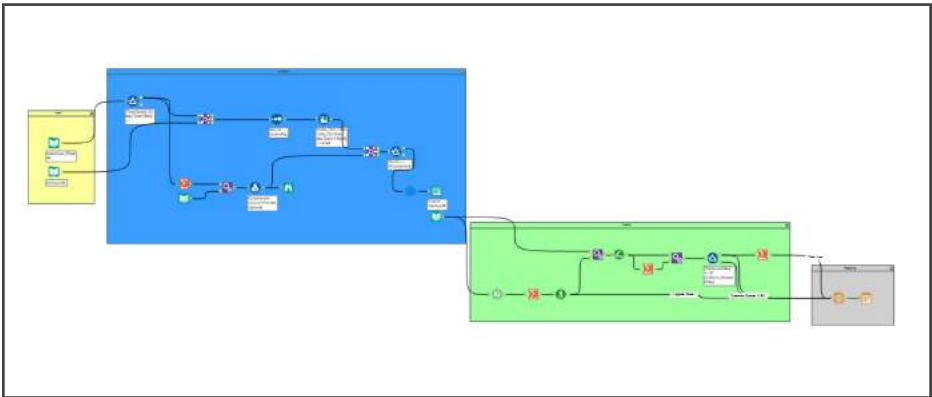
Use of Containers for Partial Execution

➤ PERFORMANCE

DESCRIPTION

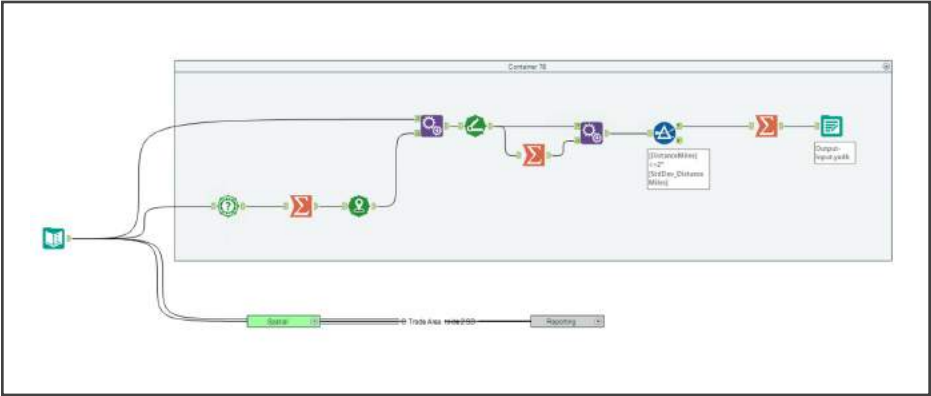
Tool Containers are useful for partial execution and organization of your module.

If anyone other than the creator of a workflow will view or use the workflow, containers can be essential for creating a clear workflow structure. By placing tools inside of the Tool Container, you are able to split data into manageable sections, and you can open and close containers without impacting the behaviour of the tools. When dealing with large workflows, this provides increased clarity of the workflow for the end user:



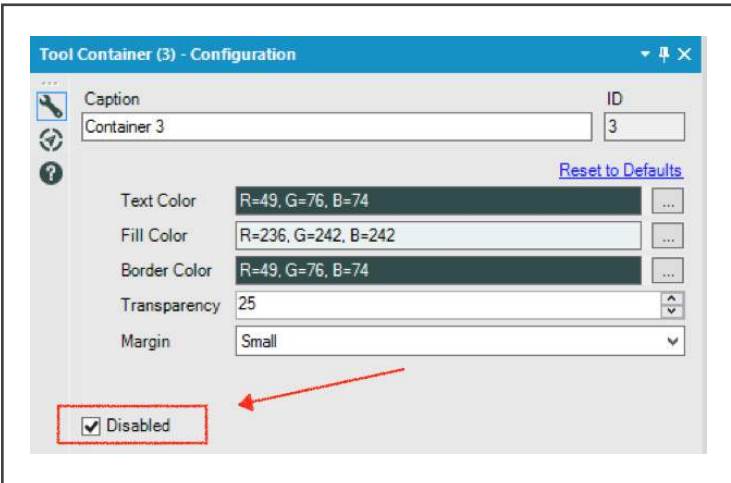
Containers are useful for testing only part of your workflow. Using containers enables you to disable the tools that are inside the container. Any tool within a disabled Tool Container will not be processed when the workflow is executed.

OPEN AND CLOSED TOOL CONTAINERS



DISABLE TOOL CONTAINER

Go to the Configuration of the Tool Container → Click Disabled



15

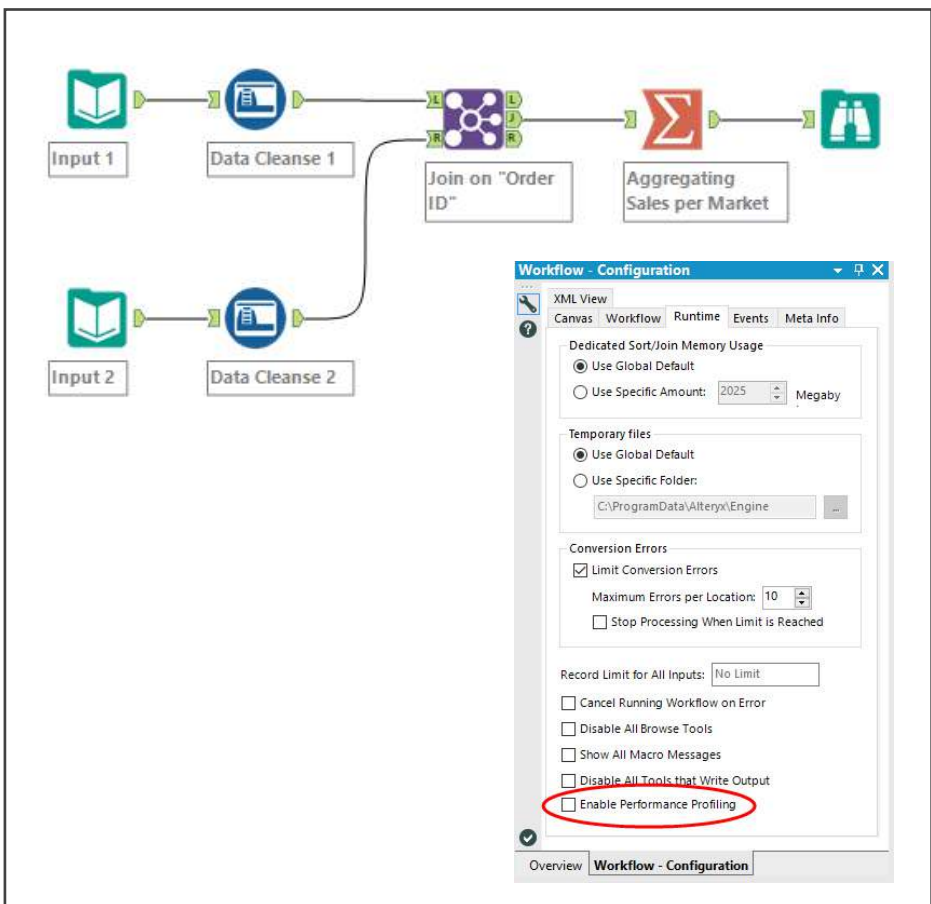
Using the Performance Profiler

➤ PERFORMANCE

DESCRIPTION

It is a best practice to optimize your workflow to run as smoothly as possible. If your workflow is slow and you would like to know what might be the cause, one option is to enable the Performance Profiler. This feature profiles each tool in the workflow, thereby letting you know which tools are taking up most of the processing.

WORKFLOW WITH PERFORMANCE PROFILER



This can be done by following these steps:

Click the Canvas → Runtime → Enable Performance Profiling.

As you can see in the following figure, Alteryx adds a *Profile Time* for each tool (in descending order) to your Output Log in the Results pane.

Results - Workflow - Messages	
0 Errors 0 Conv Errors 0 Warnings 28 Messages 3 Files All	
	Profile Time: 2629.63ms, 80.45%
	Profile Time: 331.23ms, 10.13%
	Profile Time: 169.15ms, 5.17%
	Profile Time: 110.61ms, 3.38%
	Profile Time: 20.26ms, 0.62%
	Profile Time: 6.12ms, 0.19%
	Profile Time: 1.56ms, 0.05%

HINT

Be aware that tools like *Download*, *RegEx*, and *Join* often lower performance compared to other tools. The *Download* tool lowers performance because it is usually used for downloading large amounts of data; one solution to this problem can be to run the download at night. The *RegEx* tool might impact performance, depending on the regular expression that is used. Try to optimize the search method as much as you can, e.g. by using numbers instead of text, if possible.



16

Investigate Data Using Subsamples

➤ PERFORMANCE

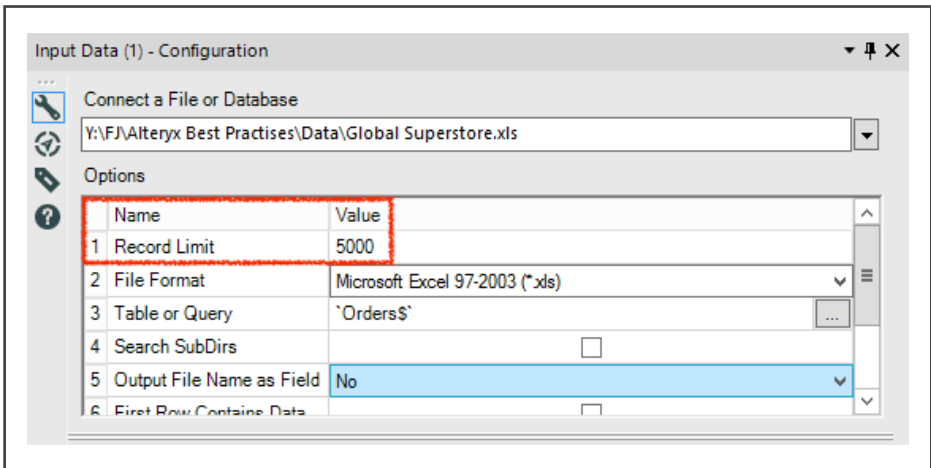
DESCRIPTION

When you initially build a workflow to execute an analysis, it is a best practice to limit the number of records that you work with. By limiting the dataset, you speed up the processing. This will save you a lot of time when you are working with large amounts of data and can help you reach your goals faster.

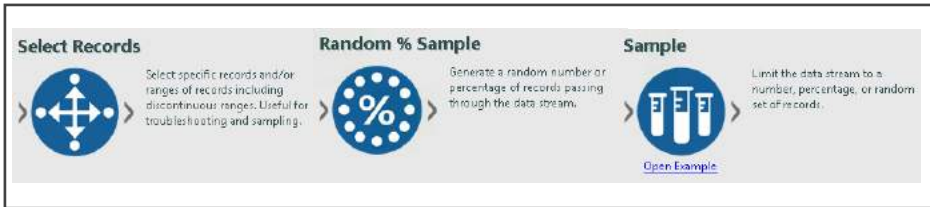
You can set a *Record Limit* in the Configuration window of your Input tool. Here you specify a value which represents the number of records upon which you initially want to develop your workflow.

SET A RECORD LIMIT IN INPUT TOOL

Input Tool



You can also make use of some of the great Alteryx tools via the *Preparation* pane. These include the *Select Records*, *Sample*, and *Random % Sample* tools. Each of these tools allows you to easily make subsamples of your dataset that you can use to start developing your analysis.

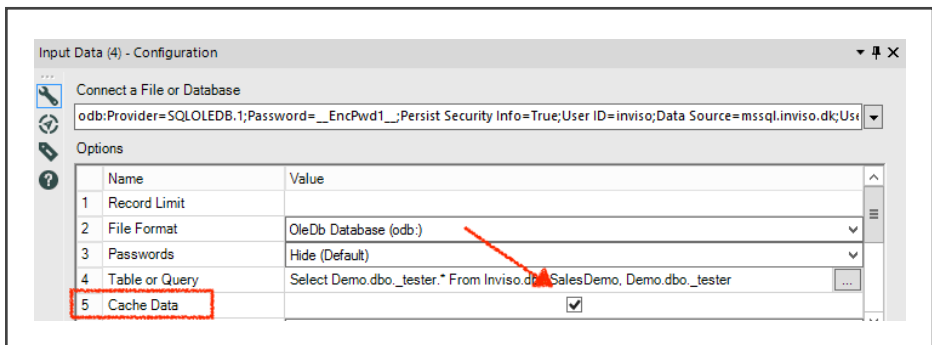


HINT

When you are working with relational database connections, you have the opportunity to use *Caching*. If you do not need live data, caching can potentially reduce the processing time by substantial margins.

When you configure the Input tool, you can choose to make use of *Cache Data* as illustrated below.

Input Tool



17

Using in-DB Tools

➤ PERFORMANCE

DESCRIPTION

When you are working with large quantities of data, it is often best practice to use *in-Database (in-DB)* tools to slice and dice your data before you drag it into your workflow. However, remember that the in-DB tools do not share the same properties of the *in-Memory* tools; the in-DB tools work only within the database to which they are connected.

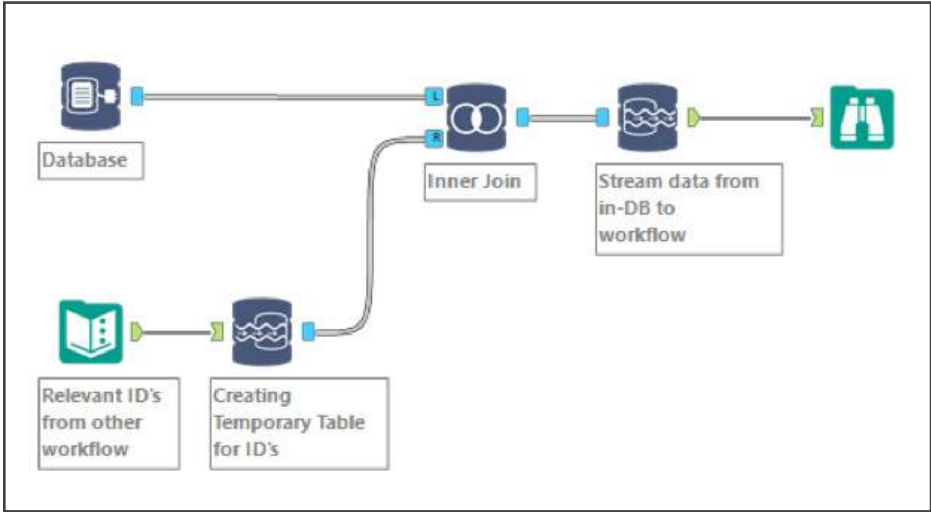
The decision to work in-DB or in-Memory is dependent upon speed and the complexity of the operations. If you gain speed by doing operations in-DB, then use in-DB tools. If you don't gain speed, or if you require the flexibility of the in-Memory tools, ignore the in-DB tools and stay in-Memory.

In-Database Pane



In-DB tools are characterized by a dark-blue ground colour, blue inputs and outputs, and “fat” connections. The green-coloured inputs and outputs indicate data from Memory to and from the database.

CUTTING DOWN DATA BY DOING IN-DB JOIN



18

Put a Timestamp on Your Data Source

➤ WHEN FINISHED

DESCRIPTION

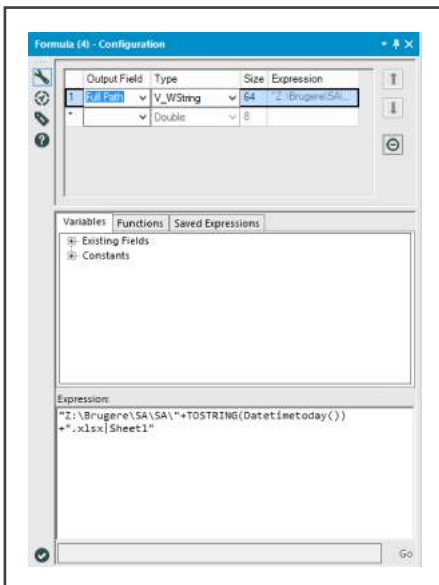
When you create reports and workflows, it is often important to inform other users and yourself of the last time that the workflow was run. One simple way to accomplish this is to append a timestamp either to the output file name or within the data source itself. To do so, you can use the *Formula* tool and the function *DateTimeToday()*.

Formula Tool

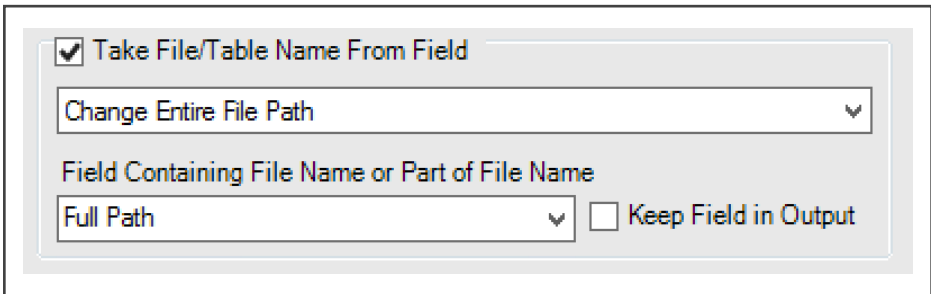


The function *DateTimeToday()* returns the date and time of the present day. When you are creating an Excel output file, first use the Formula tool to create a field with the full file path containing today's date, as shown in the following figures.

EXAMPLE OF CREATING EXCEL DATE STAMP

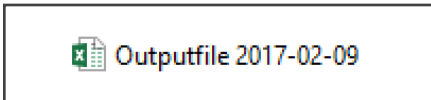


At the bottom of the configuration for the Output tool, there is an option that allows you to *take the file or table name from a field*. By checking this box, you can update or replace the filename/path using information from fields in your data stream. In the current example, choosing this option will change the entire file path using the new field, *Full Path*, that was created in the previous step.



The screenshot shows a configuration panel for an Output tool. At the top, the checkbox 'Take File/Table Name From Field' is checked. Below it is a dropdown menu set to 'Change Entire File Path'. Underneath, the label 'Field Containing File Name or Part of File Name' is followed by a dropdown menu set to 'Full Path' and an unchecked checkbox labeled 'Keep Field in Output'.

The output field will now contain the date of the workflow's last update.



19

Using the List Runner to Control Execution

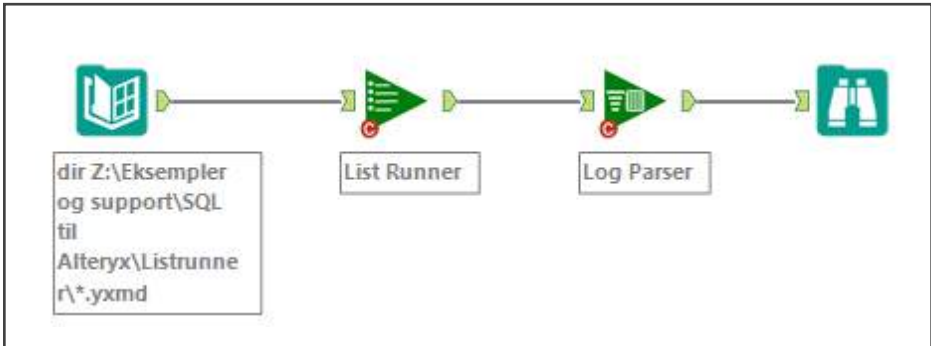
➤ WHEN FINISHED

DESCRIPTION

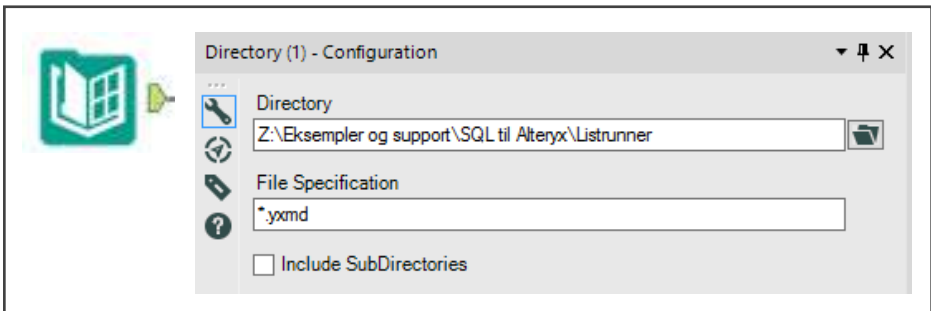
When you are working with multiple workflows, it is a best practice to use the *List Runner* tool to control the order of execution. The List Runner tool takes a list of Alteryx modules as data input and runs them one after another. It will then output all the logs to a single output, and then indicate the existence of any *Errors*, *Conversion Errors*, or *Warnings*.

By using the List Runner tool, you can run multiple workflows with a single click. At the same time, the process of debugging becomes much easier because the tool will indicate which module(s), if any, have failed.

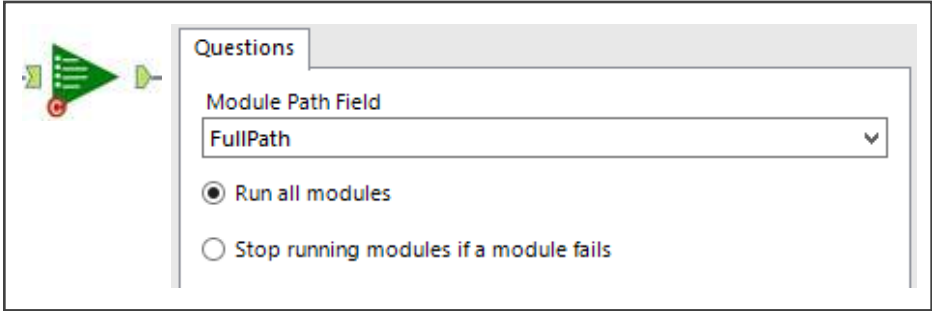
USING THE LIST RUNNER – WORKFLOW



USING THE LIST RUNNER – DIRECTORY CONFIGURATION



USING THE LIST RUNNER – *LIST RUNNER* CONFIGURATION



HINT

To improve debugging, use the List Runner tool in combination with the *Log Parser* tool. The Log Parser tool will parse the log into an Alteryx data table. The table will indicate not only the specific module, but also the tool ID and its related description at the source of the error.

20

**Clean and
Well-Constructed
Workflow**

➤ WHEN FINISHED

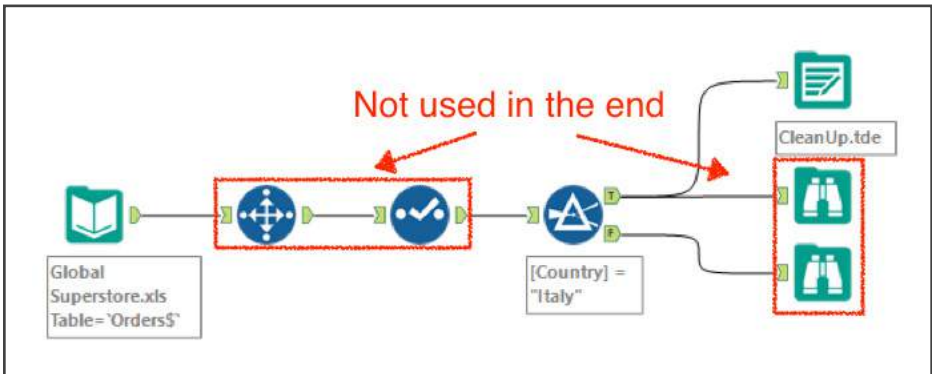
DESCRIPTION

When you have finished your workflow, make sure that you don't retain any unnecessary tools. For instance, this could be a Select tool that was used to ensure your data set contained the right data types, or a Sort tool that was used to validate your workflow, or any tool that is no longer relevant once the workflow is complete.

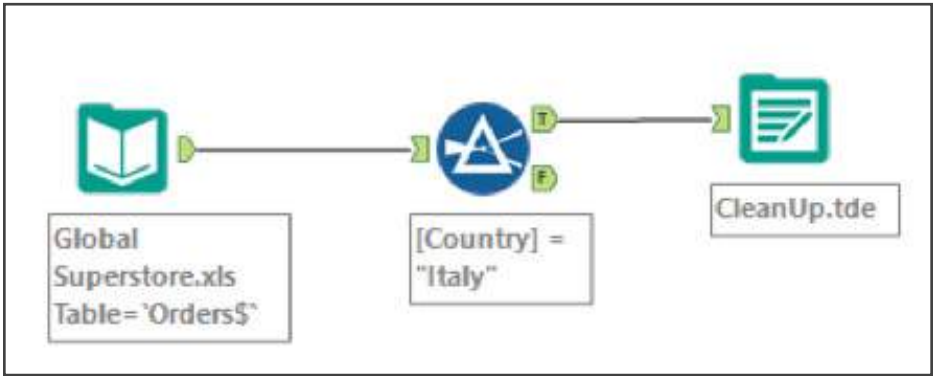
Always try to find the shortest route from input to output. Once you have solved your problem, you may find it beneficial to construct your workflow all over again. Often, you will find that you can accomplish the same result with fewer steps. This review process will also help ensure that you are not dragging along data that you are not using in the end.

By cleaning up your workflow, you will improve both the efficiency and the overview of your workflow. Your analysis will run faster, you will be able to easily hand over your workflow to a colleague or client, and it will be simpler to return to your old workflows.

INITIAL WORKFLOW



FINAL WORKFLOW



21

Relative Dependencies

➤ WHEN FINISHED

DESCRIPTION

Relative paths define relations to the location of the workflow on the user's system. There are relative paths for inputs as well as outputs. If you are using *Absolute* paths and you move your workflows around, they tend to break as the paths are broken.

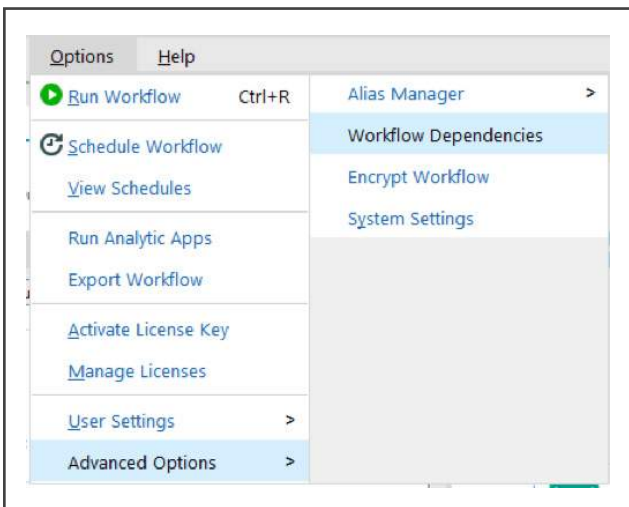
Relative Dependencies allow for great flexibility when sharing, distributing, and deploying workflows across an organization because you can move your flows around easily.

When you open the Workflow Dependencies dialog, you will be able to change one (click *Edit*) or all (click *All Relative*) of the workflow dependencies.

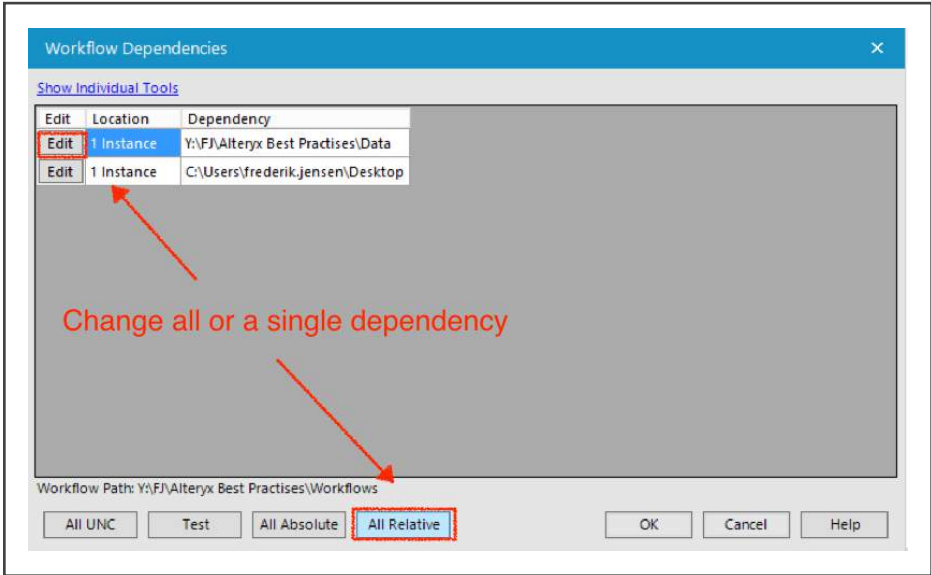
You can change the dependencies of your workflow by going to the toolbar and following these steps:

Options → *Advanced Options* → *Workflow Dependencies*

HOW TO CHANGE RELATIVE DEPENDENCIES



You can change either *All Dependencies* or *Single Dependencies*.



22

Validation Checks and Testing

➤ ANALYTICAL

DESCRIPTION

It is a best practice to conduct regular validation checks, and to use these in order to warn you that something is potentially wrong. A few examples of logic checks that could be used for testing might include:

You know that all of your ratios are supposed to sum to 100 %.

You know that no data is supposed to fall out from your join.

You know there should not be any Errors, Warnings, or Conversion Errors in your flow.

These examples could be used to set up tests that will advise you of any mistakes in your workflow. The Alteryx *Message* tool is perfect for setting up tests.

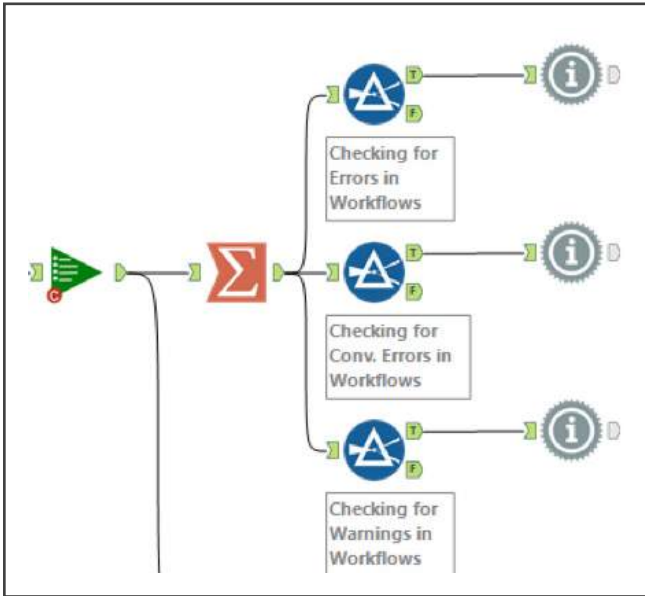
Message Tool



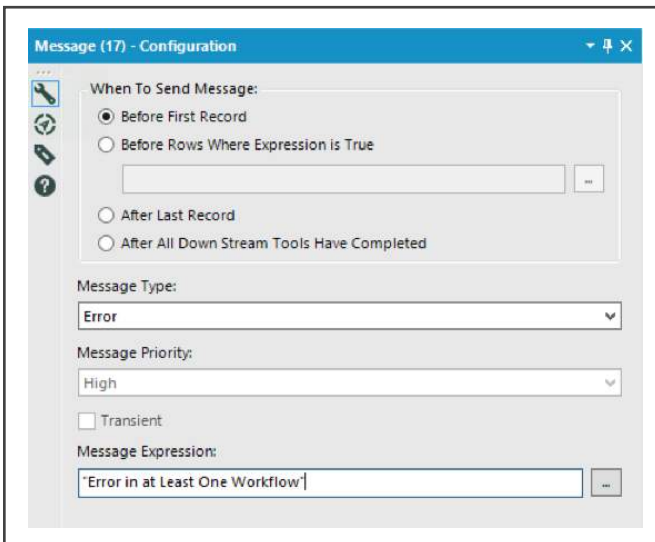
In the Message tool you can specify your own test criteria or rules, which will be reported to you in the output window when they are broken. Additionally, you can specify the *Message Expression* yourself, which will increase the level of information that is returned in the Message window.

The Message tool works very well with the List Runner tool, as you can check for the existence of Errors or Warnings in any of your workflows and the Message tool will alert you.

WORKFLOW TESTING FOR ERRORS AND WARNINGS



CONFIGURATION OF MESSAGE TOOL



23

Calculations

➤ ANALYTICAL

DESCRIPTION

It is a best practice to walk through your calculations and, ideally, to have a colleague double-check them. Many times, it is best to simply do the calculations in the front-end (e.g. Tableau), which is preferable because it will give all users direct access to the calculations.

24

**Are You Getting
the Right
Numbers?**

➤ ANALYTICAL

DESCRIPTION

Whenever possible, it is a best practice to validate the numbers of your analysis. If you have used Alteryx to reconstruct a previous analysis, it is important to compare the Alteryx results to the prior results. In this way, you can determine if the earlier analysis was mistaken or if you have done something wrong in your workflow.

If you have no certainty about the numbers, try to think carefully about the results and evaluate whether they seem reasonable.

CREDITED TO

**GREAT PEOPLE AND ALTERYX USERS FROM THE
COPENHAGEN BASED ALTERYX SUPER USER GROUP**

CREATED BY

