BEREXIA

GLOBAL ADVISOR FOR CHANGE

Berexia Hackathon 2018

Problem Statement

25/10/18

RULES:

1- The groups must be composed of two people.

2- The end of the competition is scheduled for October 28 at 9:00.

3- Each group must provide at the end of the contest:

  A. A presentation (with a demo).

  B. The source code (on Github or bitbucket).

4- Only the use of Java, Scala, Angular, React or Spark is allowed.

Problem Statement

Introduction

The Extract-Transform-Load(ETL) system is the foundation of data engineering. A properly designed ETL system extracts data from the source systems, enforces data quality and consistency standards, conforms data so that separate sources can be used together, and finally delivers data in a presentation-ready format so that the application developers can build high-end applications and end users can make decisions.

In this competition, the candidates are asked to build a simplified Proof-of-Concept of the transformation stage of an ETL system. We are going to assume the following hypothesis:

- The system only receives CSV and XSLX files.
- The first line of the file contains the headers and all the other lines are data.
- There are not blank columns or rows in the test files.
- The testing files labelled into four categories (XS, S, M, L, XL) based on their size.
- The (M, L, XL) testing data will only be available at the late game.

The Proof-of-Concept contains four main parts that will be explained in the sections below. However, keep in mind that this competition is in here to measure the best effort. If it's hard for you, it is probably hard for everyone else.

Each part is crafted to test specific set of skills that should be mastered by data engineers. The score will be calculated by a biased average taking in consideration the difficulty of each part.

Transformations

You are required to build an application that is able to do the following:

1. **Model a flowchart of transformation using drag and drop.**

As shown in the Figure 1, the main interface allows the building of a flowchart using drag and drop. The User will drag the dataset files and transformations then link them together.

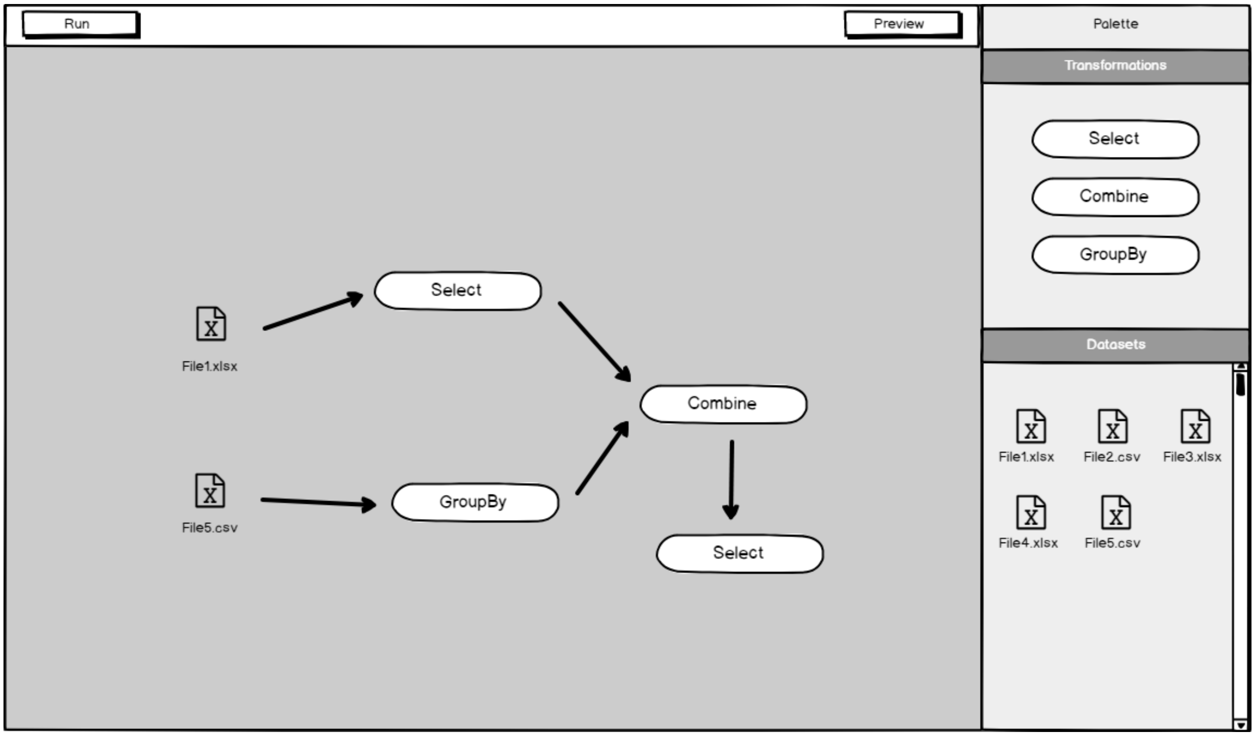


Figure 1 : Main Screen

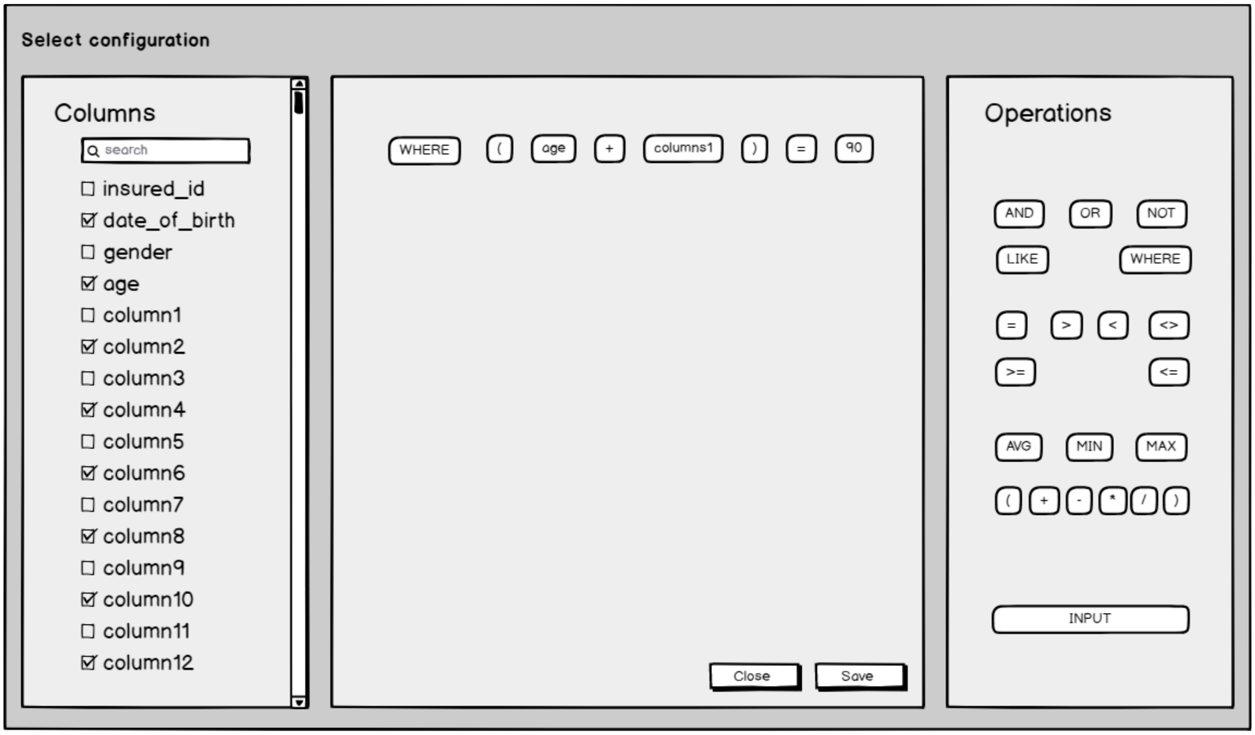
Keep in mind that at each step you need to store the output columns in-order to be able to configure the chained transformations.

2. Configure each node of the flowchart that represents a transformation.

When the user double clicks on a transformation node on the flowchart. A modal window appears and its contents are relative to the transformation type.

A. Case of Select

The User should be able to select the columns he want and apply a filter on the input columns as clearly shown in the following figure.


Select configuration

Columns

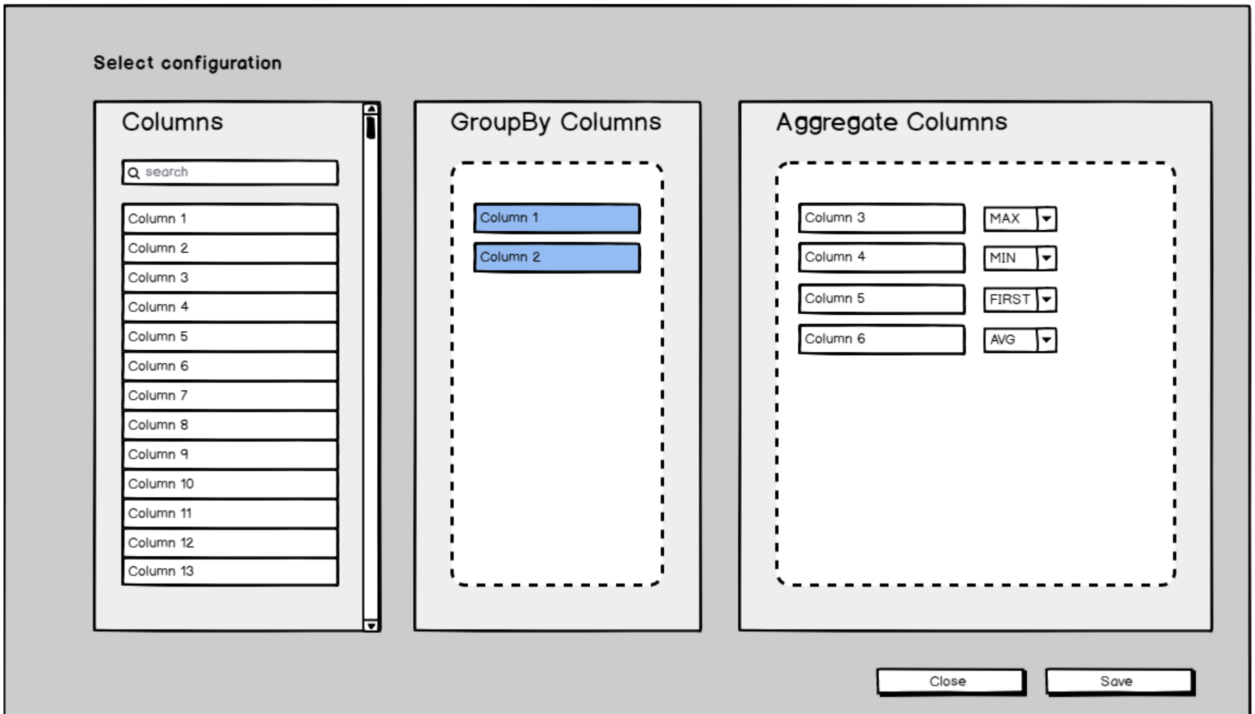
- □ insured_id
- ☑ date_of_birth
- □ gender
- ☑ age
- □ column1
- ☑ column2
- □ column3
- ☑ column4
- □ column5
- ☑ column6
- □ column7
- ☑ column8
- □ column9
- ☑ column10
- □ column11
- ☑ column12

WHERE (age + columns1) = 90

The output columns in this case are the selected columns only.

B. Case of GroupBy

The User should be able to group the rows in the file with the aggregations that are possible (MAX,MIN,FIRST,LAST,AVG).


Select configuration

Columns

- Column 1
- Column 2
- Column 3
- Column 4
- Column 5
- Column 6
- Column 7
- Column 8
- Column 9
- Column 10
- Column 11
- Column 12
- Column 13

GroupBy Columns

- Column 1
- Column 2

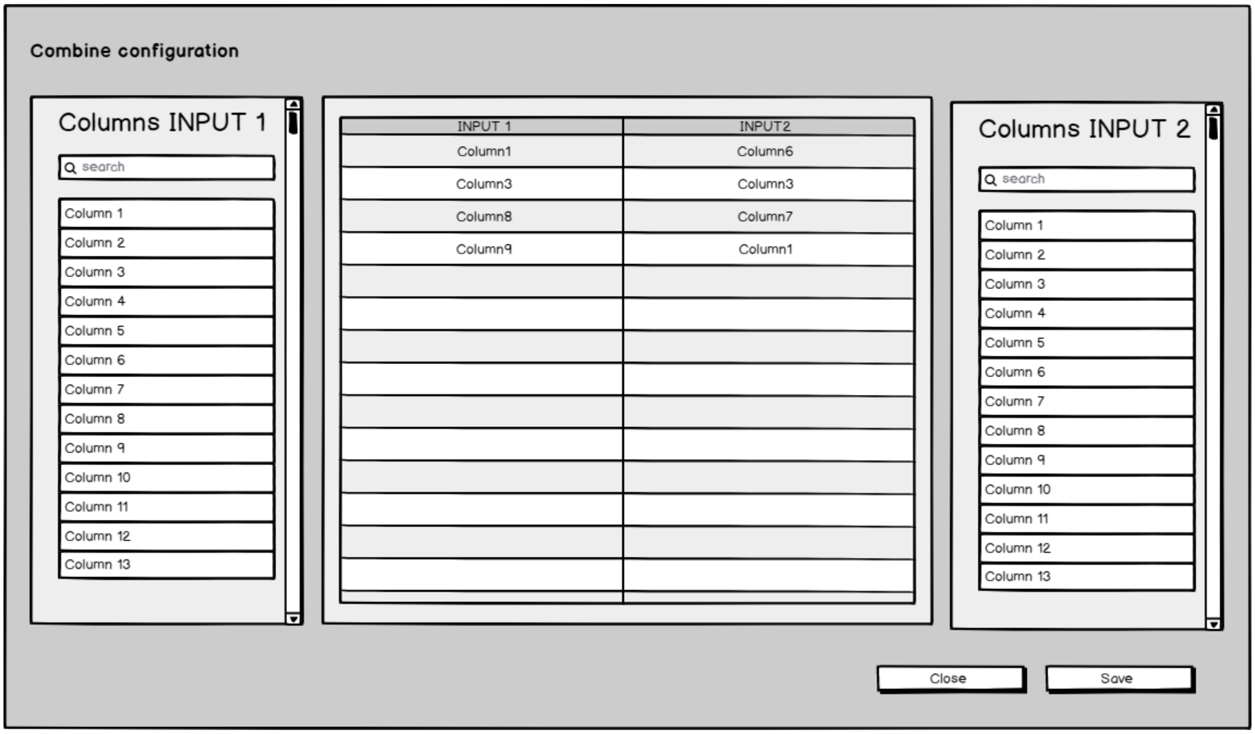
Aggregate Columns

Column 3	MAX
Column 4	MIN
Column 5	FIRST
Column 6	AVG

The output columns are the key columns (blue ones) and the aggregated columns.

C. Case of Combine

The User can only combine two columns. For the ease of this exercise, we will consider the combine an equivalent of an inner join between input 1 and input 2.



Combine configuration

Columns INPUT 1

Column 1
Column 2
Column 3
Column 4
Column 5
Column 6
Column 7
Column 8
Column 9
Column 10
Column 11
Column 12
Column 13

INPUT 1	INPUT2
Column1	Column6
Column3	Column3
Column8	Column7
Column9	Column1

Columns INPUT 2

Column 1
Column 2
Column 3
Column 4
Column 5
Column 6
Column 7
Column 8
Column 9
Column 10
Column 11
Column 12
Column 13

The output columns are those of Input 1 that are shown in the mapping.

3. Run the transformations defined by the flowchart.

After modeling the flowchart, the user will have a list of transformation to execute and should find out an order of execution.

Based on the structuring and how one wants to attack this problem (because there are many ways to view the problem at hand). You are required to implement the code of the transformations and their scheduling.

In general, you either going treat it as a big data problem or a parallel processing problem. Or, you want to just treat the XS files.

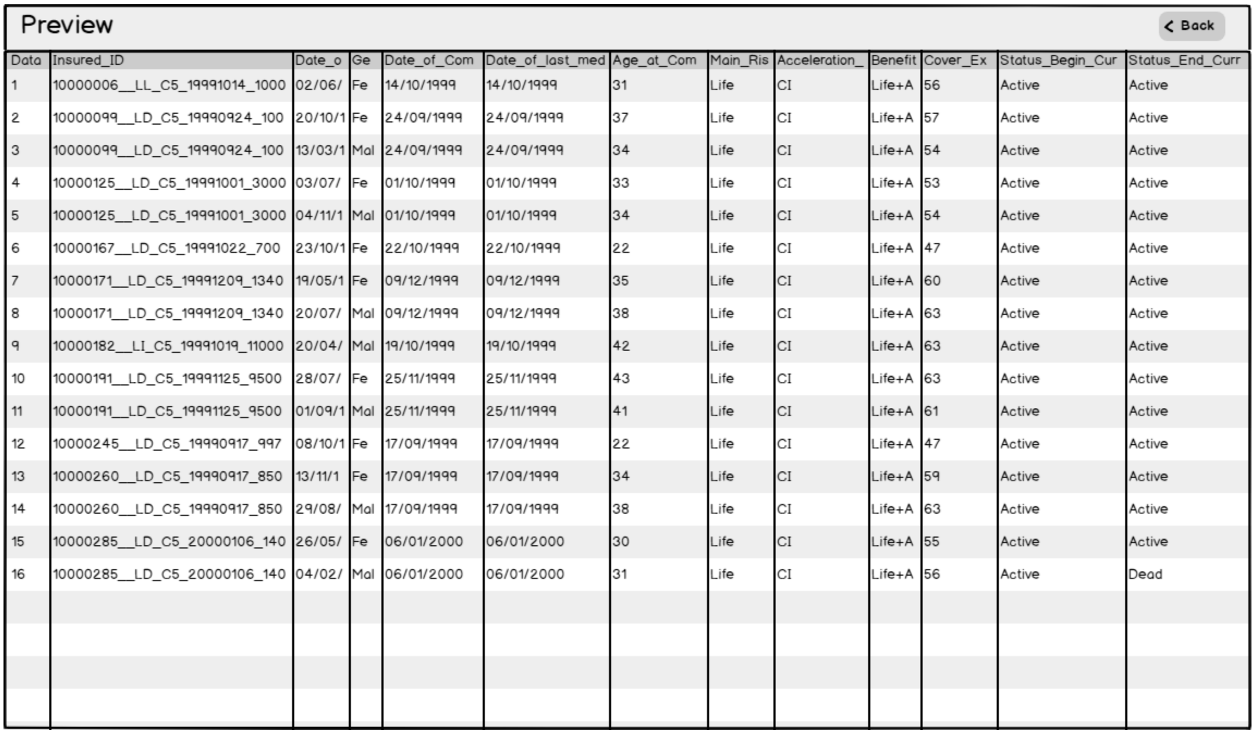
The choice of data structure and the processing model is very important and is not a direct answer as it may seem to the casual reader.

It is also to note that in terms of this competition, the flowchart is simply a directed acyclic graph. Hence, your task is to traverse the graph in the best way and at each node to do the corresponding transformation.

Select and Combine are the only two required transformation. The groupBy is added as a bonus.

4. Preview the result in a data-grid.

After the run of the transformation is done. The User can visualize the results in a Preview data-grid.



Data	Insured_ID	Date_o	Ge	Date_of_Com	Date_of_last_med	Age_at_Com	Main_Ris	Acceleration_	Benefit	Cover_Ex	Status_Begin_Cur	Status_End_Curr
1	10000006__LL_C5_19991014_1000	02/06/	Fe	14/10/1999	14/10/1999	31	Life	CI	Life+A	56	Active	Active
2	10000099__LD_C5_19990924_100	20/10/1	Fe	24/09/1999	24/09/1999	37	Life	CI	Life+A	57	Active	Active
3	10000099__LD_C5_19990924_100	13/03/1	Mal	24/09/1999	24/09/1999	34	Life	CI	Life+A	54	Active	Active
4	10000125__LD_C5_19991001_3000	03/07/	Fe	01/10/1999	01/10/1999	33	Life	CI	Life+A	53	Active	Active
5	10000125__LD_C5_19991001_3000	04/11/1	Mal	01/10/1999	01/10/1999	34	Life	CI	Life+A	54	Active	Active
6	10000167__LD_C5_19991022_700	23/10/1	Fe	22/10/1999	22/10/1999	22	Life	CI	Life+A	47	Active	Active
7	10000171__LD_C5_19991209_1340	19/05/1	Fe	09/12/1999	09/12/1999	35	Life	CI	Life+A	60	Active	Active
8	10000171__LD_C5_19991209_1340	20/07/	Mal	09/12/1999	09/12/1999	38	Life	CI	Life+A	63	Active	Active
9	10000182__LI_C5_19991019_11000	20/04/	Mal	19/10/1999	19/10/1999	42	Life	CI	Life+A	63	Active	Active
10	10000191__LD_C5_19991125_9500	28/07/	Fe	25/11/1999	25/11/1999	43	Life	CI	Life+A	63	Active	Active
11	10000191__LD_C5_19991125_9500	01/09/1	Mal	25/11/1999	25/11/1999	41	Life	CI	Life+A	61	Active	Active
12	10000245__LD_C5_19990917_997	08/10/1	Fe	17/09/1999	17/09/1999	22	Life	CI	Life+A	47	Active	Active
13	10000260__LD_C5_19990917_850	13/11/1	Fe	17/09/1999	17/09/1999	34	Life	CI	Life+A	59	Active	Active
14	10000260__LD_C5_19990917_850	29/08/	Mal	17/09/1999	17/09/1999	38	Life	CI	Life+A	63	Active	Active
15	10000285__LD_C5_20000106_140	26/05/	Fe	06/01/2000	06/01/2000	30	Life	CI	Life+A	55	Active	Active
16	10000285__LD_C5_20000106_140	04/02/	Mal	06/01/2000	06/01/2000	31	Life	CI	Life+A	56	Active	Dead

BONUS / ADVANCED

Having a transformation scheme that runs from end to end is the goal of our application. But to ensure the reliability and maintainability of the solution a debug feature must exist.

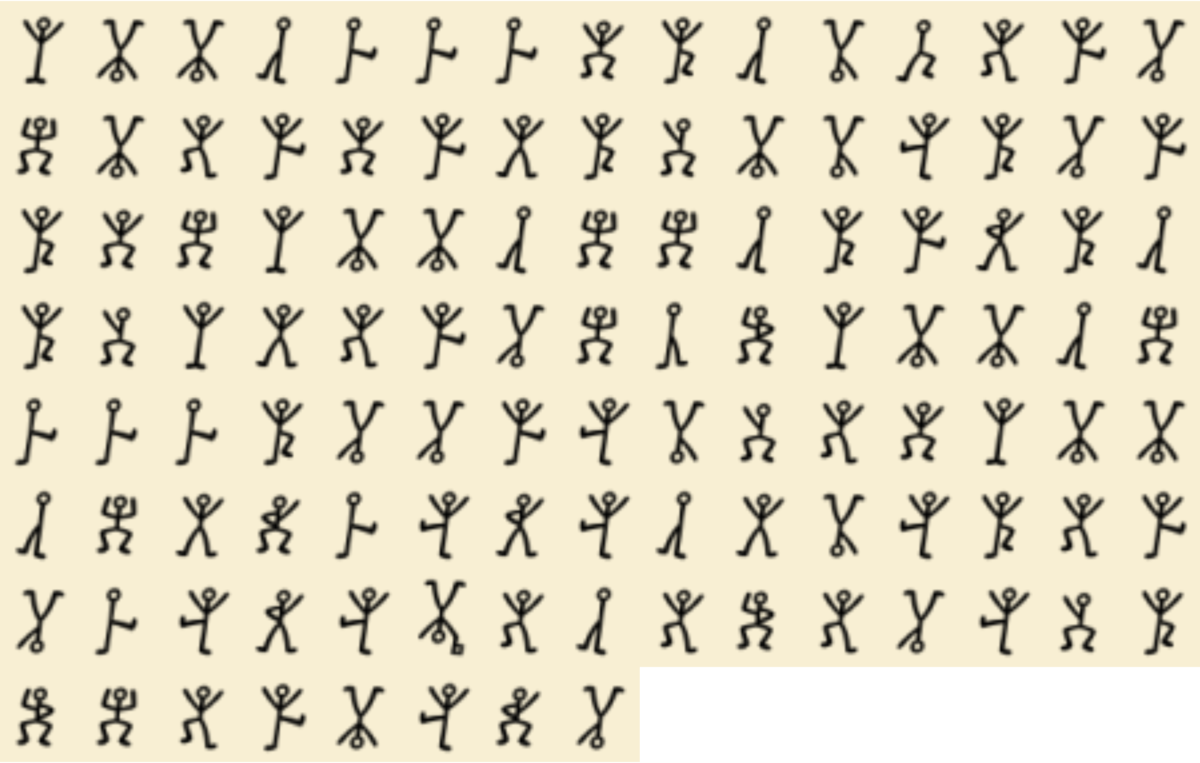
Debug functionality is a feature that allows you to choose a transformation on the schema and define it as a breakpoint.
The objective of this breakpoint and execute the schema of transformations and stop at the transformation defined as breakpoint, and therefore our result will not be the one of the last transformation on the schema but that of the breakpoint.

To ensure this, it will be necessary to ensure:

- Interconnectivity: The transformations forming the schema must be well connected to allow the detection of the transformations that directly or indirectly impact the result of our breakpoint.

- Analysis of the links: Having related transformations is not enough, but to ensure the breakpoint, it will be necessary to ensure an analysis of the links to be able to execute only the transformations necessary to have the result.

An analysis of links will avoid the execution of transformations that have no impact on the expected result.



—
GK