OpenSchoolMaps: Data cleansing and integrating Apachehop

OpenSchoolMaps.ch – Free learning materials on free geodata and maps and open source.

This is a worksheet with exercises for self-learners and students.

Overview

Goals

This worksheet demonstrates how Apache Hop can be used to read, cleanse, filter, merge and store data.

The tasks consist of realistic examples that involve working with heterogeneous data. The aim is to show how to deal with this data when confronted with problems in this area.

After completing this worksheet, you will be able to:

- Understand and operate **Apache Hop**.
- Create simple data processing workflows in Apache Hop.
- Merge multiple data sources using Apache Hop.
- Transform column values and save them in a new column.

Time Required

The time required to complete this worksheet is about one hour for the reading part (without exercises), plus about five quarters of an hour for the exercises part—both depending on your previous knowledge and skills.

Prerequisites

In order to complete this worksheet, you need the following prerequisites:

- Internet access to download the required software and data.
- Java 17, available at oracle.com.
- Software: Apache Hop, available for Windows, Mac and Linux, and to be installed as described below.
- Data: File `Daten_OpenRefine.zip` from OpenSchoolMaps

The following topics may be helpful as preparation for this topic:

- Basic understanding of working with data.
- The worksheet `Viewing, cleaning and integrating data with OpenRefine` on OpenSchoolMaps.

Installation of Hop

Apache Hop can be downloaded from the Apache Hop website. It is important to note that **Java 17** is required to start the application. Both the source code and the finished programme (binaries) are offered on the site. Only the binaries are required to run the application. The downloaded file contains everything you need to start Apache Hop on the most common operating systems.

If you run into problems, there is GitHub Discussions, which contains information about a lot of the most common issues.

Structure of this Worksheet

- Introduction: Overview of Apache Hop, its functions and areas of application.
- User interface (GUI): Description of the GUI and its elements for creating pipelines and workflows.
- Apache Hop functions: Detailed explanations of various transformation options and data operations.
- Exercises: Practical tasks on using Apache Hop, including data integration, cleansing and transformation.
- Conclusion: Summary of concepts learnt and their practical relevance.
- What was learnt: Overview of key lessons learnt from the exercises.
- Appendix: Definitions: An overview over the most important keywords.

Apache Hop-Basics

Apache Hop (**H**op **O**rchestration **P**latform) is an open source data orchestration platform, used for the creation of data integration processes. The application was developed in Java based on a platform-independent and modular architecture. This means that the application can be flexibly extended with plugins for a specific use case.

Hop uses a visual user interface to visualise the processes as clearly as possible. Local files as well as external databases can be used as a means of reading and saving data. Apache Hop findet Verwendung bei z.B.:

- Data integration
- Data migration
- Automation of workflows and data processes
- Data cleansing

The Hop Engine is the core of Apache Hop. It is accessible via three clients: Hop GUI, Hop Run and Hop Server. Hop GUI is the visual development environment in which data teams develop, test, execute and debug workflows and pipelines. Hop Run is the command line interface (CLI) for executing workflows and pipelines. Hop Server is a lean web server for executing workflows and pipelines as web services (HTTP REST API).

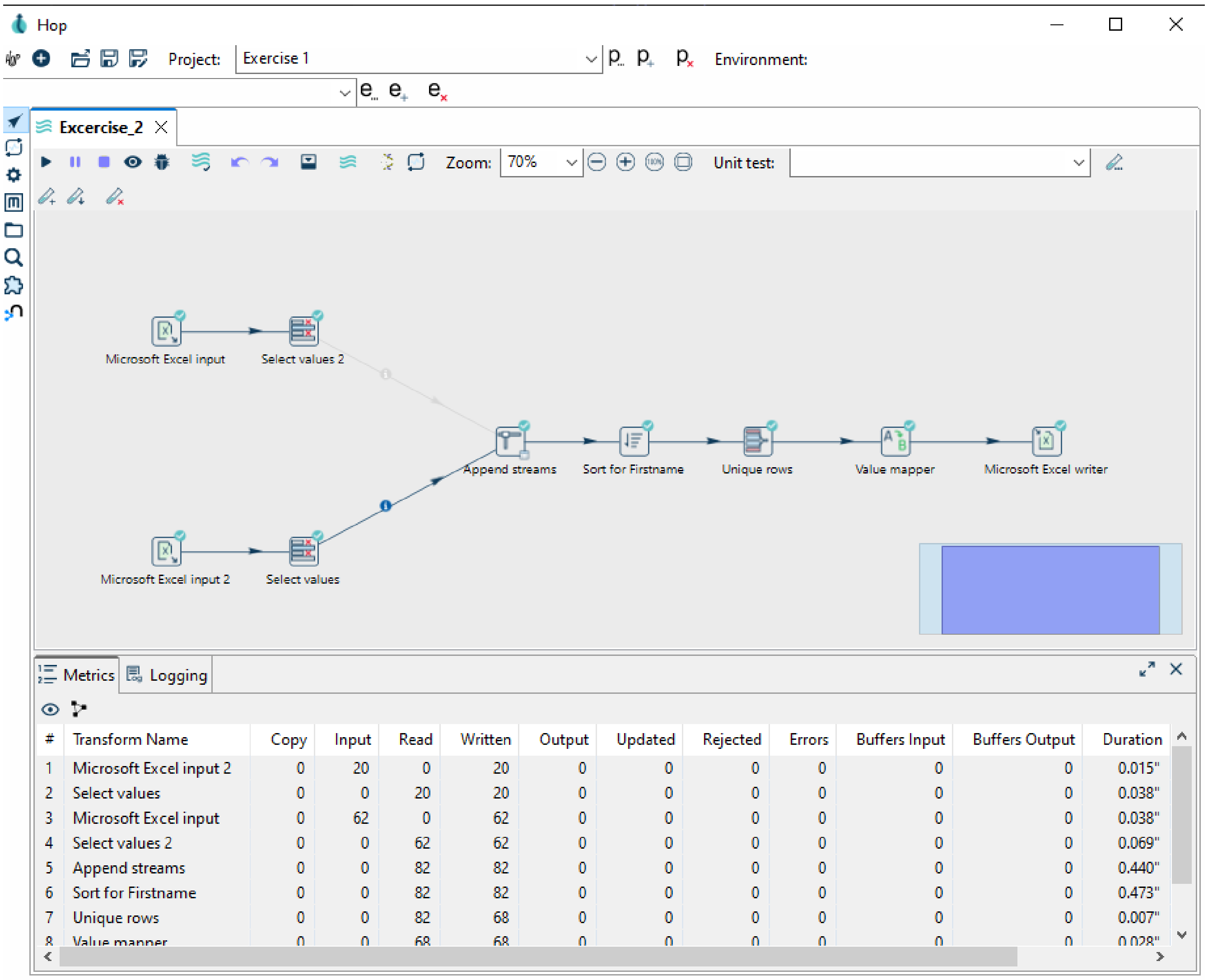
The User Interface (GUI)

Apache Hop offers a graphical user interface (GUI) in which the projects are created and displayed. It consists of a main window that serves as a workspace. The current project and its content are displayed here.

In Apache Hop, transforms, actions and hops are central concepts for processing and automating data.

- **Transforms** are the individual processing steps within a pipeline. They perform tasks such as loading, converting or saving data.
- **Actions** are the building blocks of a workflow and control processes, e.g. starting a pipeline, sending a notification or accessing external systems.
- **Hops** connect transforms and display the data flow in a pipeline. In workflows, you define the sequence of actions.

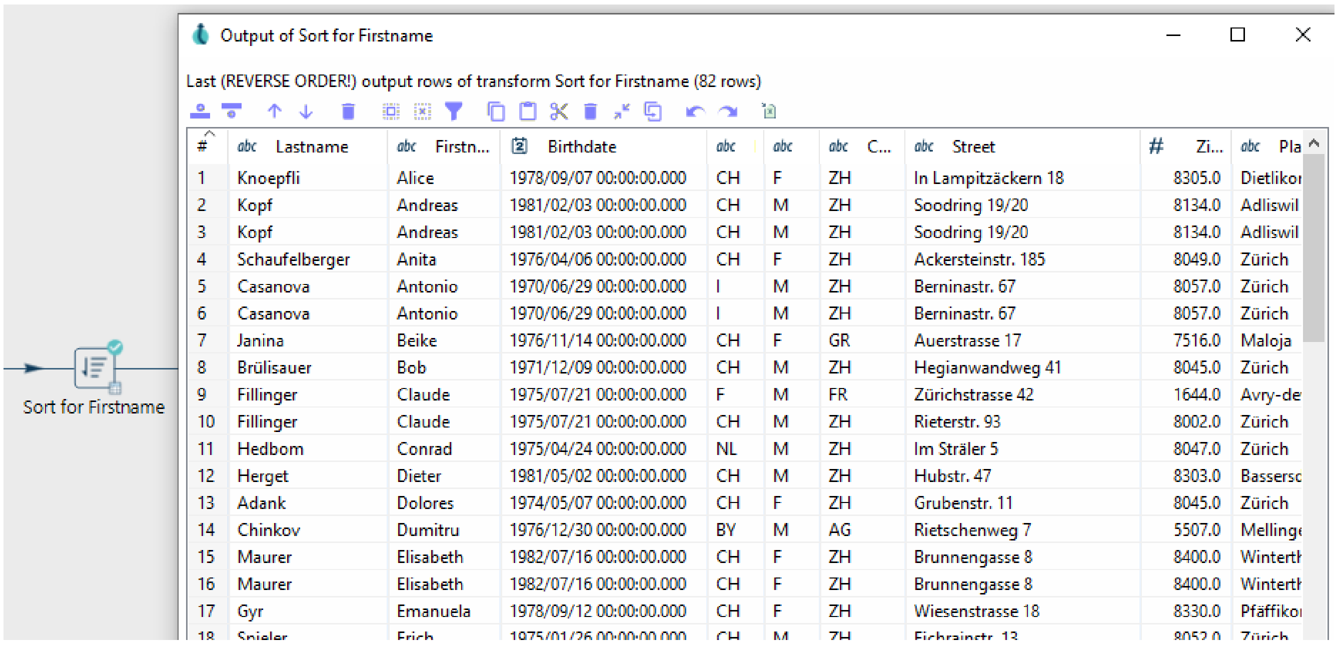
These three elements enable a flexible and visual design of data processing and automation processes in Apache Hop.



#	Transform Name	Copy	Input	Read	Written	Output	Updated	Rejected	Errors	Buffers Input	Buffers Output	Duration
1	Microsoft Excel input 2	0	20	0	20	0	0	0	0	0	0	0.015"
2	Select values	0	0	20	20	0	0	0	0	0	0	0.038"
3	Microsoft Excel input	0	62	0	62	0	0	0	0	0	0	0.038"
4	Select values 2	0	0	62	62	0	0	0	0	0	0	0.069"
5	Append streams	0	0	82	82	0	0	0	0	0	0	0.440"
6	Sort for Firstname	0	0	82	82	0	0	0	0	0	0	0.473"
7	Unique rows	0	0	82	68	0	0	0	0	0	0	0.007"
8	Value mapper	0	0	68	68	0	0	0	0	0	0	0.028"

Figure 1. Apache Hop GUI.

After a pipeline has been executed, a small icon appears next to the individual transforms, which you can click on to view the data at this point in the execution.



Output of Sort for Firstname

Last (REVERSE ORDER!) output rows of transform Sort for Firstname (82 rows)

#	Lastname	Firstn...	Birthdate			C...	Street	Zi...	Pla
1	Knoepfli	Alice	1978/09/07 00:00:00.000	CH	F	ZH	In Lampitzäckern 18	8305.0	Dietlikor
2	Kopf	Andreas	1981/02/03 00:00:00.000	CH	M	ZH	Soodring 19/20	8134.0	Adliswil
3	Kopf	Andreas	1981/02/03 00:00:00.000	CH	M	ZH	Soodring 19/20	8134.0	Adliswil
4	Schaufelberger	Anita	1976/04/06 00:00:00.000	CH	F	ZH	Ackersteinstr. 185	8049.0	Zürich
5	Casanova	Antonio	1970/06/29 00:00:00.000	I	M	ZH	Berninastr. 67	8057.0	Zürich
6	Casanova	Antonio	1970/06/29 00:00:00.000	I	M	ZH	Berninastr. 67	8057.0	Zürich
7	Janina	Beike	1976/11/14 00:00:00.000	CH	F	GR	Auerstrasse 17	7516.0	Maloja
8	Brülisauer	Bob	1971/12/09 00:00:00.000	CH	M	ZH	Hegianwandweg 41	8045.0	Zürich
9	Fillinger	Claude	1975/07/21 00:00:00.000	F	M	FR	Zürichstrasse 42	1644.0	Avry-de
10	Fillinger	Claude	1975/07/21 00:00:00.000	CH	M	ZH	Rieterstr. 93	8002.0	Zürich
11	Hedbom	Conrad	1975/04/24 00:00:00.000	NL	M	ZH	Im Sträler 5	8047.0	Zürich
12	Herget	Dieter	1981/05/02 00:00:00.000	CH	M	ZH	Hubstr. 47	8303.0	Bassersc
13	Adank	Dolores	1974/05/07 00:00:00.000	CH	F	ZH	Grubenstr. 11	8045.0	Zürich
14	Chinkov	Dumitru	1976/12/30 00:00:00.000	BY	M	AG	Rietschenweg 7	5507.0	Mellinge
15	Maurer	Elisabeth	1982/07/16 00:00:00.000	CH	F	ZH	Brunnengasse 8	8400.0	Winterth
16	Maurer	Elisabeth	1982/07/16 00:00:00.000	CH	F	ZH	Brunnengasse 8	8400.0	Winterth
17	Gyr	Emanuela	1978/09/12 00:00:00.000	CH	F	ZH	Wiesenstrasse 18	8330.0	Pfäffikor
18	Spieler	Erich	1975/01/26 00:00:00.000	CH	M	ZH	Eichrainstr. 13	8052.0	Zürich

Figure 2. Display of the result after Apache Hop has executed a pipeline.

Pipelines can also be combined into larger constructs using **workflows**. Here, various pipelines, also called **actions**, are connected with **hops**. In this case, however, these represent the sequential execution of the individual actions.

Projects

In HOP, projects are the basis for all kinds of tasks.

Environments in Apache Hop are predefined configurations that make it possible to run projects in different contexts without having to make manual adjustments. They contain specific settings such as variables, connection details or paths and can be used for development, test and production environments, for example. For example, a pipeline in the development environment can work with local files, while in the production environment it automatically uses a database as a source. Environments help to make workflows flexible and reusable.

In addition, each project is assigned its own project folder in which the project files are saved. Apache Hop does not usually change the original data unless it is configured to do so in the pipeline.

Apache Hop-Functions

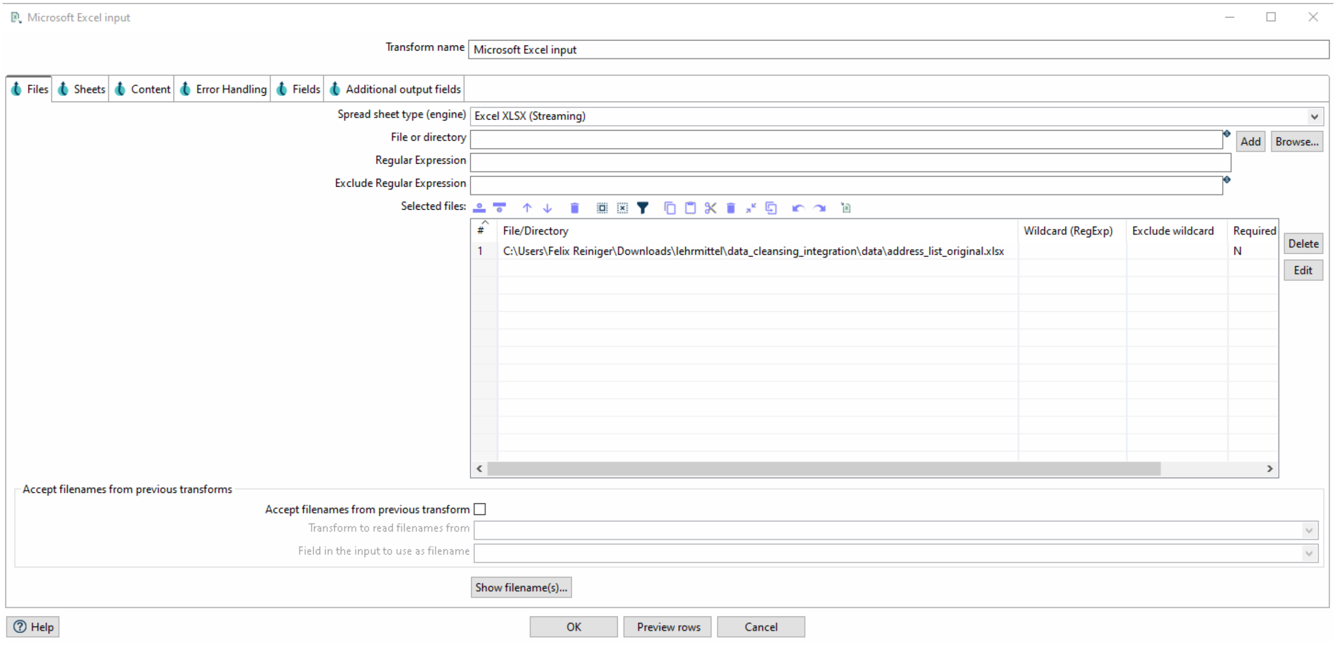
Apache Hop offers a wide range of powerful functions for efficient control of the entire data flow. Data can be easily imported and exported as well as transformed, merged and validated in complex workflows. The modular and visual interface enables flexible integration and customization of processes, which offers decisive advantages, particularly when importing data, combining data streams and cleansing data records. Some of these functions are explained in more detail below.

In- and Output

In Apache Hop there is a large selection of input and output transforms that can be used to read and write data. A few selected transforms are described below.

Excel Input

Excel files can be integrated directly into a pipeline as a data source with the `Microsoft Excel Input`.



Transform name: Microsoft Excel input

Spread sheet type (engine): Excel XLSX (Streaming)

File or directory:

Regular Expression:

Exclude Regular Expression:

Selected files:

#	File/Directory	Wildcard (RegExp)	Exclude wildcard	Required
1	C:\Users\Felix Reiniger\Downloads\lehrmittel\data_cleansing_integration\data\address_list_original.xlsx			N

Accept filenames from previous transforms

Accept filenames from previous transform ☐

Transform to read filenames from

Field in the input to use as filename

Figure 3. The parameter menu of the Excel Input Transform

In the **Files › Browse...** tab, the file must first be referenced and then added to the list of **Selected Files** by clicking on **[Add]**.

The corresponding `sheets` to be read can then be selected in the **Sheets › List of sheets** tab under **[Get sheetnames(s)...]**. Finally, the header can be read in in the **Fields › Define fields schema:** tab via **[Get fields from header row...]**. If desired, the data types of the individual columns can now be customised.

Excel Writer

The `Microsoft Excel Writer` can be used to save a data flow as an Excel file (`.xlsx`, `.xls`). To Export the data as a Excel file you'll have to reference a saving location.

Transforms

Apache Hop is particularly powerful when it comes to transforming data. Thanks to the visual representation, pipelines can be efficiently compiled and analysed. As Hop has a flexible structure, it can also be easily integrated into existing processes and infrastructures. Some of the available transformations are listed and explained below.

Merging data records from different sources

Apache Hop offers various functions for merging different data. Depending on what is to be achieved, different transforms can be used.

Append Streams Transform

The `Append Streams` transform can be used to append one data flow to another. The parameters refer to `Tail` and `Head`. The `tail` is placed at the end of the `head`. The schema, i.e. the columns of the tables and their data types, of the two sources must be identical, otherwise an error will occur.

Merge Join Transform

The `Merge Join` transform in Apache Hop enables two data streams to be linked using a common key. A join type such as `INNER`, `LEFT OUTER` or `FULL OUTER` is selected, similar to SQL joins. It is important that the input data is sorted in advance, as Merge Join does not perform automatic sorting. After the join, the combined data records are processed as a single data stream.

Validating and Deduplication

Sorting Transform

The `Sorting` Transform sorts a data flow alphabetically/numerically in ascending or descending order. The corresponding columns must be selected in the parameters. Several columns can also be selected. In this case, the order in which the columns are specified is decisive.

Unique Rows Transform

Duplicates can be removed with the `Unqiue Rows` Transform. Columns relevant for checking can be defined in the parameters. However, this transform only works with sorted data flows. Accordingly, a `Sorting` transform must be applied beforehand.

Value Mapper Transform

The Value Mapper Transform is used to convert values within a data stream. It makes it possible to replace certain input values with defined target values.

To use it, add a `Value Mapper` transform in a pipeline. Open the parameters and enter the original value under `Source value` and the desired target value under `Target value`. After saving, the defined values are automatically replaced when running through the pipeline. This is particularly useful for data cleansing or the standardisation of values.

Exercise 1: First Apache hop pipeline

In this exercise, you will build a simple pipeline with Apache Hop. An Excel file is read in, filtered, the data is cleansed and finally saved again.

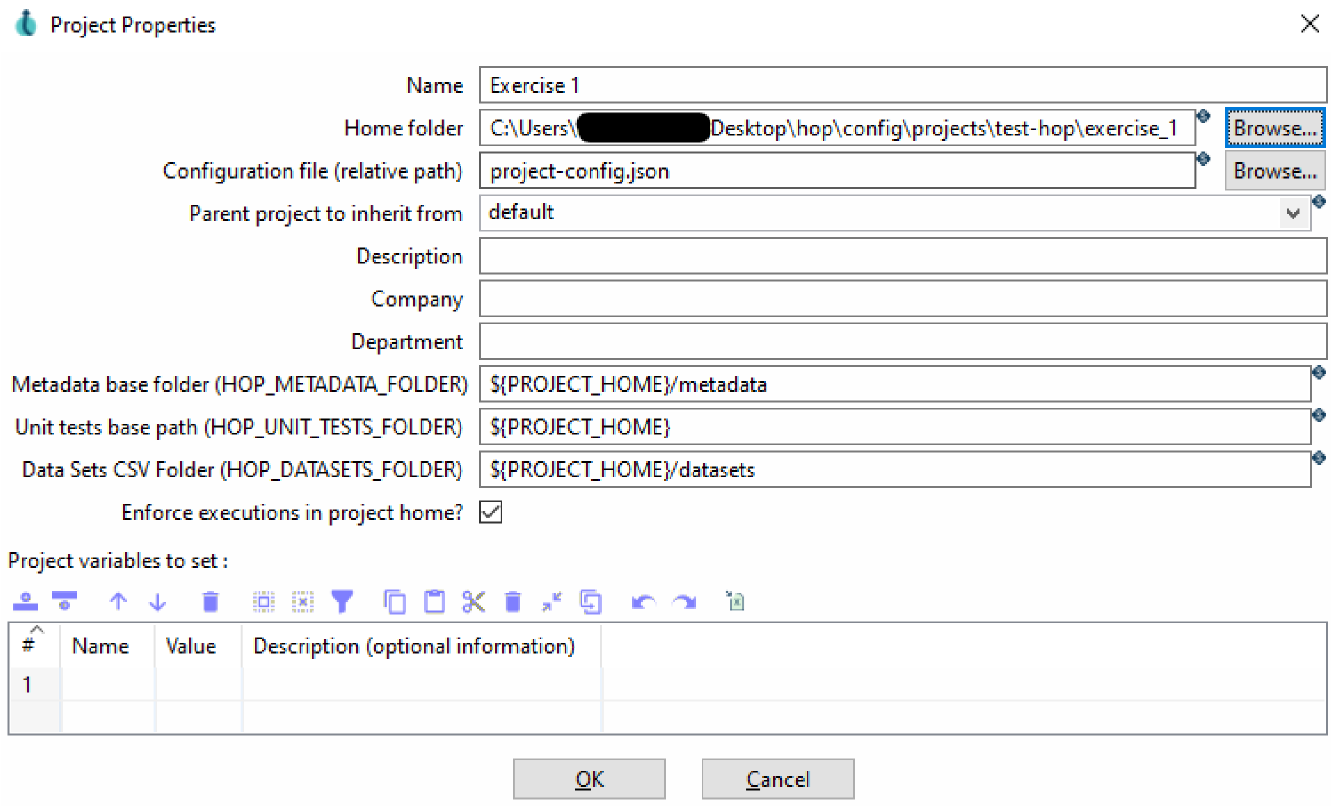
Data

For this excersise `address_list_original.xlsx` is used. This is part of the zip archive `Daten_OpenRefine.zip`, which you can download from OpenSchoolMaps (same section as this worksheet).

Step 1: Create a Project

To create a project - the actual workspace in which you work with the data - you must do the following:

1. start Apache Hop on your local machine.
2. click on **[Add a new Project]** in the top bar.
3. in the new window you must give the project a name and define a `Home folder`. You can confirm the entry with **[OK]**.
4. a new window then appears in which you can define enviorment variables. You can click these away with **[No]**.



Name	Exercise 1
Home folder	C:\Users\[redacted]Desktop\hop\config\projects\test-hop\exercise_1
Configuration file (relative path)	project-config.json
Parent project to inherit from	default
Description	
Company	
Department	
Metadata base folder (HOP_METADATA_FOLDER)	${PROJECT_HOME}/metadata
Unit tests base path (HOP_UNIT_TESTS_FOLDER)	${PROJECT_HOME}
Data Sets CSV Folder (HOP_DATASETS_FOLDER)	${PROJECT_HOME}/datasets
Enforce executions in project home?	✓

Project variables to set :

#	Name	Value	Description (optional information)
1			

Figure 4. New Projects.

Step 2: Load and Validate Data

Filter out all Customers that are not from Zurich, so the dataset only includes customers from Zurich.

1. load the data from the file `address_list_original.xlsx` with the help of `Microsoft Excel Input`.
2. apply the transform `Filter rows` to the column `Canton` to filter only customers who live in the

canton ZH.

3. use the separate transform Standardize phone number to add the country code and bring the phone number into a standardised format.

> Standardise Phone Number is a transform that is exemplary for transforms added by plugins. Alternatively, you can also use a String Operation to convert the data into the desired form.

Step 3: Export Data

Once you have completed the above tasks, you can add an export to the pipeline. Use the Microsoft Excel Writer for this. In the options, select xlsx [Excel 2007 and above] as the format and choose a suitable file name.

Exercise 2: Integrating another data set

In this exercise, you integrate a data record into a predefined target data record. Some column names and structures are changed, otherwise the data set remains unchanged. Such scenarios are common in practice, as data from various sources have to be merged into a comprehensive data set. With Apache Hop, you can use various techniques - such as mapping attributes, merging and splitting fields, and deduplication. and deduplication - to successfully standardise the different data streams.

Data

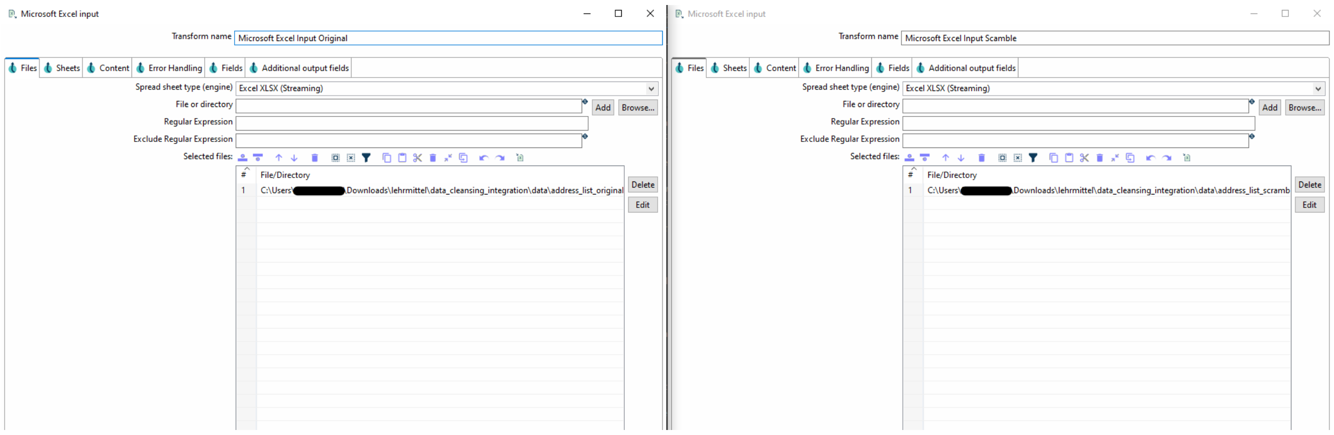
The data from the files address_list_original.xlsx & address_list_scrambled.xlsx are used for this task.

Step 1: Create a Project

Once you have downloaded both Excel files, create a new pipeline by clicking on the **[+]** in the top bar and selecting **[Pipeline]**.

Step 2: Integrate the Data

To import both data into one pipeline, two separate Microsoft Excel inputs must be created. Reference one Excel file for each transform. You will then first convert the data into a standardised format, before you merge the two data flows.


Microsoft Excel input

Transform name: Microsoft Excel Input Original

Files | Sheets | Content | Error Handling | Fields | Additional output fields

Spread sheet type (engine): Excel XLSX (Streaming)

File or directory:

Regular Expression:

Exclude Regular Expression:

Selected files:

#	File/Directory
1	C:\Users\[redacted]\Downloads\lehrmittel\data_cleansing_integration\data\address_list_original

Microsoft Excel input

Transform name: Microsoft Excel Input Scamble

Files | Sheets | Content | Error Handling | Fields | Additional output fields

Spread sheet type (engine): Excel XLSX (Streaming)

File or directory:

Regular Expression:

Exclude Regular Expression:

Selected files:

#	File/Directory
1	C:\Users\[redacted]\Downloads\lehrmittel\data_cleansing_integration\data\address_list_scramb

Figure 5. Parameters for both Excel inputs

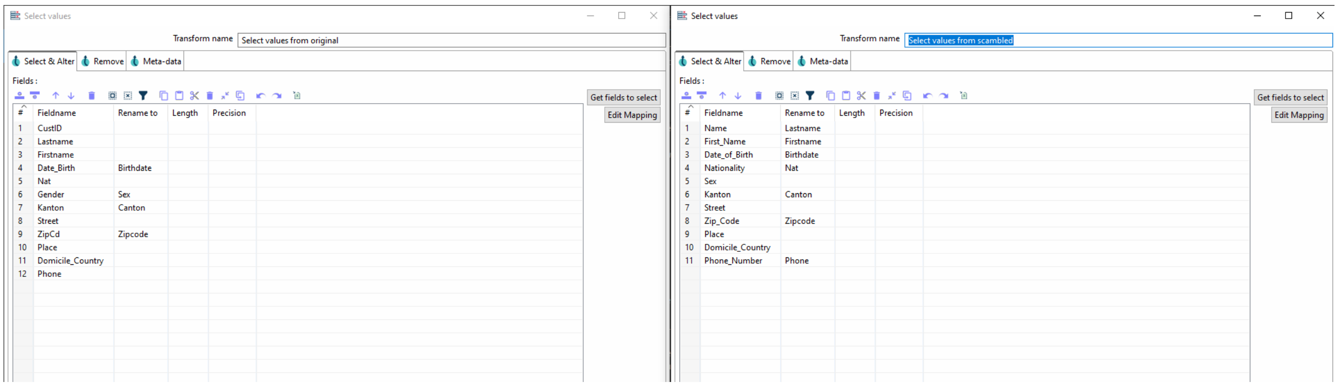
Step 2.1 Merging and standardising the columns from both files

First of all, you need to look at the columns in both files and find out which columns belong together.

Once you have found matching columns, they must first be matched using `Select Values` and then merged using `Append Streams`. Append Streams only works with data that has an identical structure. As soon as the tables have been merged, they can be processed further.

In the parameters for `Select Values` you can rename the column under `rename`.

You can also use `Remove` to remove columns such as `CustID` that only occur in one table.


Select values

Transform name: Select values from original

Select & Alter | Remove | Meta-data

Fields :

#	Fieldname	Rename to	Length	Precision
1	CustID			
2	Lastname			
3	Firstname			
4	Date_Birth	Birthdate		
5	Nat			
6	Gender	Sex		
7	Kanton	Canton		
8	Street			
9	ZipCd	Zipcode		
10	Place			
11	Domicile_Country			
12	Phone			

Select values

Transform name: Select values from scambled

Select & Alter | Remove | Meta-data

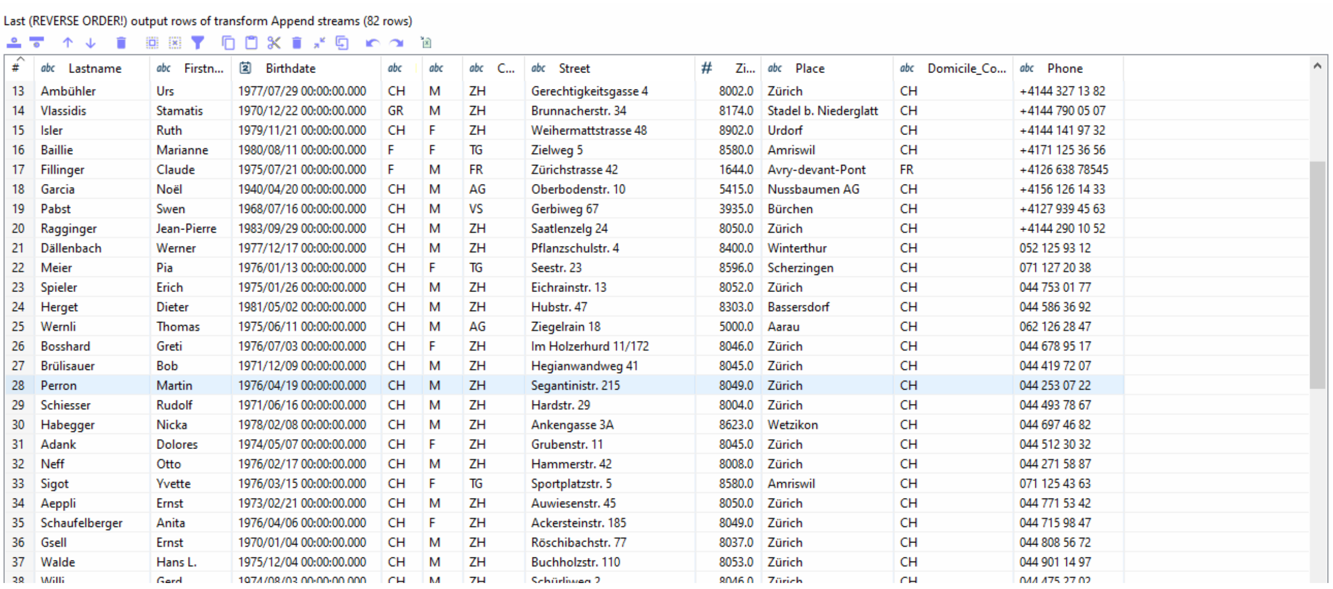
Fields :

#	Fieldname	Rename to	Length	Precision
1	Name	Lastname		
2	First_Name	Firstname		
3	Date_of_Birth	Birthdate		
4	Nationality	Nat		
5	Sex			
6	Kanton	Canton		
7	Street			
8	Zip_Code	Zipcode		
9	Place			
10	Domicile_Country			
11	Phone_Number	Phone		

Figure 6. Select Values parameters of both streams.

If both streams generate an identical schema, they can be merged with `Append Streams`. You must define `head` and `tail` in the parameters. However, the order does not play a role in this example.

If you now execute the pipeline and an error occurs, it may well be that the two streams do not yet have an identical schema.



Last (REVERSE ORDER!) output rows of transform Append streams (82 rows)

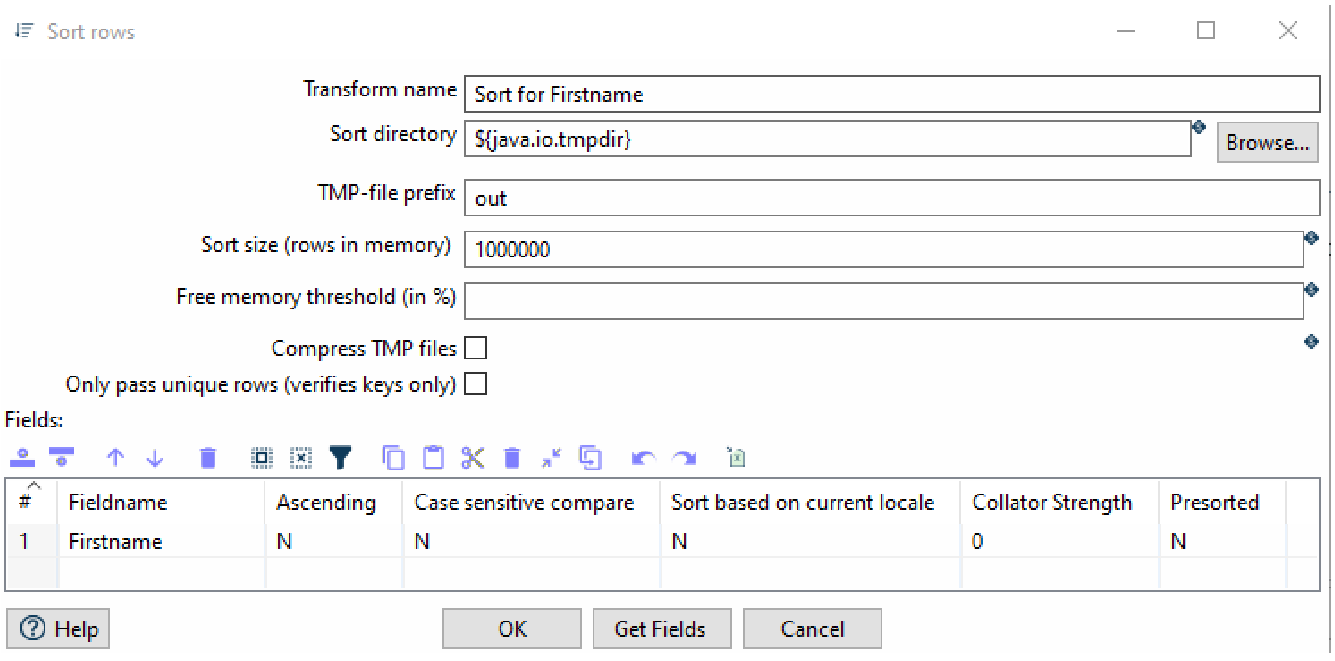
#	Lastname	Firstn...	Birthdate			C...	Street	Zi...	Place	Domicile_Co...	Phone
13	Ambühler	Urs	1977/07/29 00:00:00.000	CH	M	ZH	Gerechtigkeitsgasse 4	8002.0	Zürich	CH	+4144 327 13 82
14	Vlassidis	Stamatis	1970/12/22 00:00:00.000	GR	M	ZH	Brunnacherstr. 34	8174.0	Stadel b. Niederglatt	CH	+4144 790 05 07
15	Isler	Ruth	1979/11/21 00:00:00.000	CH	F	ZH	Weihermattstrasse 48	8902.0	Urdorf	CH	+4144 141 97 32
16	Baillie	Marianne	1980/08/11 00:00:00.000	F	F	TG	Zielweg 5	8580.0	Amriswil	CH	+4171 125 36 56
17	Fillinger	Claude	1975/07/21 00:00:00.000	F	M	FR	Zürichstrasse 42	1644.0	Avry-devant-Pont	FR	+4126 638 78545
18	Garcia	Noël	1940/04/20 00:00:00.000	CH	M	AG	Oberbodenstr. 10	5415.0	Nussbaumen AG	CH	+4156 126 14 33
19	Pabst	Swen	1968/07/16 00:00:00.000	CH	M	VS	Gerbiweg 67	3935.0	Bürchen	CH	+4127 939 45 63
20	Ragginger	Jean-Pierre	1983/09/29 00:00:00.000	CH	M	ZH	Saatlenzelg 24	8050.0	Zürich	CH	+4144 290 10 52
21	Dällenbach	Werner	1977/12/17 00:00:00.000	CH	M	ZH	Pflanzschulstr. 4	8400.0	Winterthur	CH	052 125 93 12
22	Meier	Pia	1976/01/13 00:00:00.000	CH	F	TG	Seestr. 23	8596.0	Scherzingen	CH	071 127 20 38
23	Spieler	Erich	1975/01/26 00:00:00.000	CH	M	ZH	Eichrainstr. 13	8052.0	Zürich	CH	044 753 01 77
24	Herget	Dieter	1981/05/02 00:00:00.000	CH	M	ZH	Hubstr. 47	8303.0	Bassersdorf	CH	044 586 36 92
25	Wernli	Thomas	1975/06/11 00:00:00.000	CH	M	AG	Ziegelrain 18	5000.0	Aarau	CH	062 126 28 47
26	Bosshard	Greti	1976/07/03 00:00:00.000	CH	F	ZH	Im Holzerhurd 11/172	8046.0	Zürich	CH	044 678 95 17
27	Brülisauer	Bob	1971/12/09 00:00:00.000	CH	M	ZH	Hegianwandweg 41	8045.0	Zürich	CH	044 419 72 07
28	Perron	Martin	1976/04/19 00:00:00.000	CH	M	ZH	Segantinistr. 215	8049.0	Zürich	CH	044 253 07 22
29	Schiesser	Rudolf	1971/06/16 00:00:00.000	CH	M	ZH	Hardstr. 29	8004.0	Zürich	CH	044 493 78 67
30	Habegger	Nicka	1978/02/08 00:00:00.000	CH	M	ZH	Ankengasse 3A	8623.0	Wetzikon	CH	044 697 46 82
31	Adank	Dolores	1974/05/07 00:00:00.000	CH	F	ZH	Grubenstr. 11	8045.0	Zürich	CH	044 512 30 32
32	Neff	Otto	1976/02/17 00:00:00.000	CH	M	ZH	Hammerstr. 42	8008.0	Zürich	CH	044 271 58 87
33	Sigot	Yvette	1976/03/15 00:00:00.000	CH	F	TG	Sportplatzstr. 5	8580.0	Amriswil	CH	071 125 43 63
34	Aeppli	Ernst	1973/02/21 00:00:00.000	CH	M	ZH	Auwiesenstr. 45	8050.0	Zürich	CH	044 771 53 42
35	Schaufelberger	Anita	1976/04/06 00:00:00.000	CH	F	ZH	Ackersteinstr. 185	8049.0	Zürich	CH	044 715 98 47
36	Gsell	Ernst	1970/01/04 00:00:00.000	CH	M	ZH	Röschibachstr. 77	8037.0	Zürich	CH	044 808 56 72
37	Walde	Hans L.	1975/12/04 00:00:00.000	CH	M	ZH	Buchholzstr. 110	8053.0	Zürich	CH	044 901 14 97
38	Willi	Gerd	1974/08/03 00:00:00.000	CH	M	ZH	Schürliweg 2	8046.0	Zürich	CH	044 475 27 02

Figure 7. Both streams sucessfully merged.

Step 2.2 Deduplication of the data

The next step is to deduplicate the data. Some data from the second file is also present in the first file, which means that these are duplicates, which you need to remove and only keep one of the entries. There is also other data in the second file that does not exist in the first file (i.e. no duplicates), so you do not need to remove them.

In order for a stream to be cleaned up using `Unique Rows`, it must first be sorted. You can use the `Sort` transform for sorting.



Sort rows

Transform name	Sort for Firstname
Sort directory	${java.io.tmpdir}
TMP-file prefix	out
Sort size (rows in memory)	1000000
Free memory threshold (in %)	
Compress TMP files	☐
Only pass unique rows (verifies keys only)	☐

Fields:

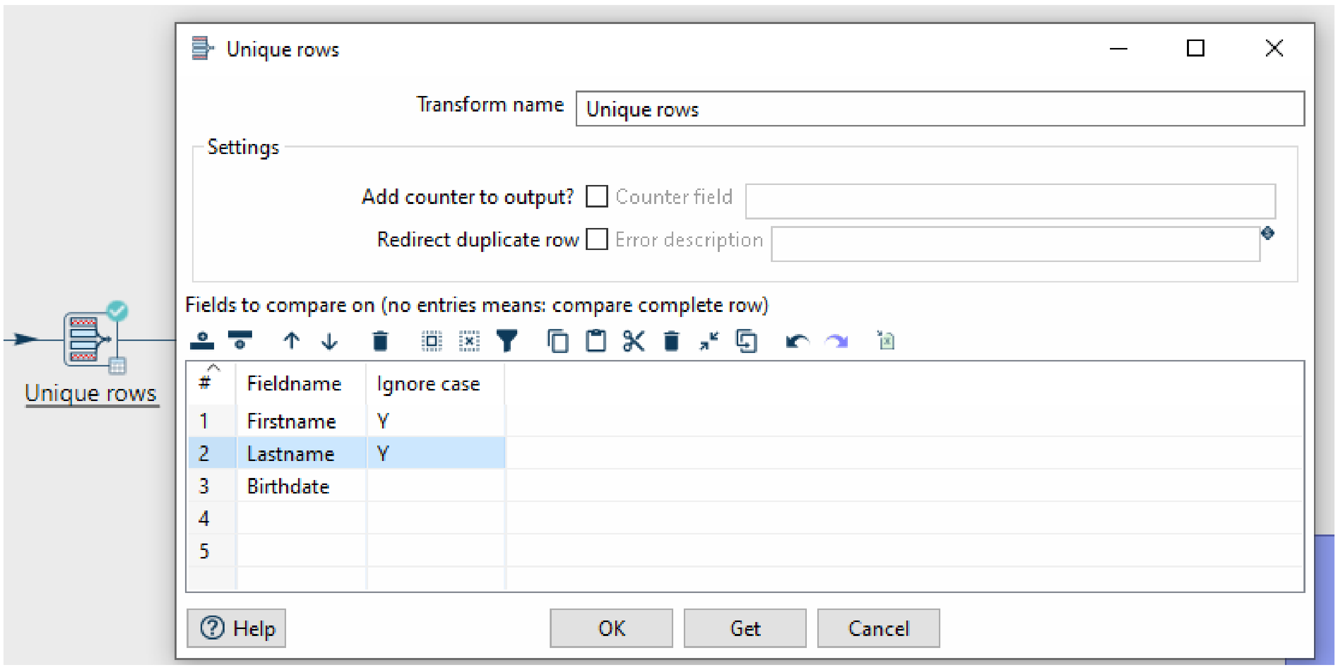
#	Fieldname	Ascending	Case sensitive compare	Sort based on current locale	Collator Strength	Presorted
1	Firstname	N	N	N	0	N

Figure 8. Sorting according to the column 'Firstname'.

It is important that all rows are sorted according to the columns that are used to identify the duplicate rows, are sorted. However, it does not matter whether this is in ascending or descending order.

For a data record to be considered a duplicate, the following condition must be met: If the first name, surname and date of birth are identical, it is the same person, i.e. a duplicate.

For this we can use the `Unique Rows` Transform, which removes duplicates. If no specific columns are specified in the parameters, all columns are checked.



Transform name: Unique rows

Settings

Add counter to output? ☐ Counter field

Redirect duplicate row ☐ Error description

Fields to compare on (no entries means: compare complete row)

#	Fieldname	Ignore case
1	Firstname	Y
2	Lastname	Y
3	Birthdate	
4		
5		

Figure 9. Deduplicate using the `Unique Rows` Transform.

Step 3: Finalisation

The telephone numbers are stored in different formats in the ‘Phone’ column. Apply what you have learnt from task 1. However, make sure that international telephone numbers also have the correct dialling code.

Use other columns for help if necessary.

Once this last task has been completed, the source data is successfully integrated, validated and deduplicated in the target data set.

Step 4: Exporting the data

Add an `Export` tranform to your pipeline to export the files in *MS Excel 2007+ (.xslx)* format and finish this exercise. complete this exercise.

Exercise 3: Using Value Mapper Transform

The aim of this exercise is to add another column based on the existing data. In this case, the `Sex /Gender` column is to be used to create a new column called `Salutation`, which contains the corresponding salutation.

Data

For this task, you can either use the pipeline from task 2 or use `address_list_original.xlsx`.

Step 1: Create project and import data (optional)

If you are working with a new pipeline, you must first create a new pipeline via the **[+]** and read the data from the Excel file using an `Excel Reader`,

Step 2: Create Value Mapper Transform

Use a `Value-Mapper` transform to create a new column called `Salutation`, which inserts `M` → `Male` or `F` → `Female` depending on the gender specified. is specified, which inserts `M` → `Mr` or `F` → `Female`.

Step 3: Export data

To complete this task, the data must be exported to the format *MS Excel 2007+ (.xslx)* using an `Excel Writer` transform. format.

Conclusion

Based on the points discussed, it was shown that Apache Hop is a powerful tool for processing and preparing data. data. The intuitive and visual environment makes it easy to visualise and understand complex processes.

Working with data and transforming it into something meaningful and useful requires a lot of logical, technical and practical knowledge. practical knowledge is required. In the exercises, you will hopefully have learnt how to deal with confusing data and which functions and tools to choose for different data problems. tools you can choose for different data problems.

Things learned

- Create a project in Apache Hop.
- Transforming a dataset with Apache Hop, using different transforms.
- Using Apache Hop to clean/duplicate and integrate one dataset into another.
- Use a `Value Mapper` transform to create a new column.

Appendix: Definitions

Tools

Apache Hop offers several tools for modelling, executing and managing data integration processes. The most important are:

- **Hop Gui**: Main user interface for graphically creating pipelines and workflows. When starting out, the only tool required.

- **Hop Run**: Command line tool for executing workflows and pipelines.
- **Hop Conf**: Configuration tool for managing projects and environments.
- **Hop Server**: Lightweight web server for remote execution of pipelines and workflows.
- as well as **other tools**, such as Hop Encrypt (passwords), Hop Search (metadata objects in projects), Hop Import (for Kettle)

Workflow

A structured sequence of one or more **pipelines** and **actions**. Is used to control the **data flow**, i.e. for flow and error logic, time control and conditional executions. Used for **data flow management**. (Not used in this exercise).

Pipeline

Structured sequence for data processing, consisting of individual transforms, connected by **hops**. Used for the actual **data processing**: Loading, transforming and writing data. Can be parallelised and runs data set-oriented.

Transform

A processing step within a pipeline (e.g. reading, filtering, converting data). Performs operations such as filtering, renaming, joining or converting. Works on **tabular data** and changes schema or format if necessary. Part of **data processing**.

Hop (within pipelines)

Connects **Transforms** in pipelines, **Actions** in workflows and controls the flow of data from one step to the next.

Action

An operation within a workflow - e.g. to start a pipeline or send an email. Performs tasks such as starting a pipeline, sending an email or waiting for a file. Returns **boolean** result values (success/failure). Does **no own data processing**. (Not used in this exercise).

Project

A logical collection of pipelines, workflows, metadata and configurations. Used to structure hop content. Supports **project-specific configurations** such as parameters (declarative variables, can also have default values) or simple variables (key-value pairs that are used at runtime).

Environment

A collection of configuration values that are applied to a **project** at runtime. Allows pipelines and workflows to be executed in different contexts (e.g. Dev, Test, Prod).

Metadata

Reusable objects such as database connections, file formats, table descriptions or user-defined values. Are managed centrally and referenced in pipelines/workflows.

File Definition

A metadata structure that describes how a structured file (e.g. CSV) should be read or written (e.g. separator, header, character set).

Execution Engine

Apache Hop supports multiple execution engines, e.g. the local engine or Apache Beam for scalable and distributed execution.

Hop Web

(future) planned web interface for control and modelling. Currently not included in the stable version, but under development.

Metadata injection

Dynamic insertion of metadata into pipelines to create flexible, parameterisable processes - e.g. based on Excel or database definitions.

Row

Basic unit of data processing in a pipeline. A row can consist of any number of fields (e.g. attributes such as `customer_id`, `amount`, `timestamp`).

Open questions? Feel free to contact OpenStreetMap Schweiz or Stefan Keller!