# ETL design training session – data and process quality

This document serves as a training session for using Talend Open Studio (TOS) tool (a.k.a. Talend) for improving quality of data and quality of the ETL process.

**Content:**

a. we will use the ETL design created in the previous training session (provided as **dw_training.zip**) to showcase in Talend the process of improving the ETL data and process quality characteristics, based on different specified quality goals.

***Data sources used in the training:*** All examples are created over the ACME Flying Use Case. For better understanding of internals of the system under the study, the schematic/diagrammatic representations of the domain and the available data sources are available (see slides: **FlyingUseCase.pdf**):

1) Diagrammatic/schematic representation of the **ACME subsystems** (AIMS and AMOS)
2) AIMS and AMOS DB schema (IE notation)
   a. With SQL to create **AIMS** and **AMOS tables (AIMS.sql and AMOS.sql)**.
3) External sources:
   **a. aircraft-manufaturerinfo-lookup.csv**
   **b. maintenance-personnel-airport-lookup.csv**
   **c. aircraft_boeing_master.csv**
   **d. aircraft_airbus_master.csv**

# ETL design: training session – data and process quality

## *Part A: Enhancing data and process quality*

Using the ETL process that we designed in the previous part, we will showcase how we can improve quality dimensions of the process, while maintaining the same functionality. Examples of such quality dimensions are data quality, performance , reliability etc.

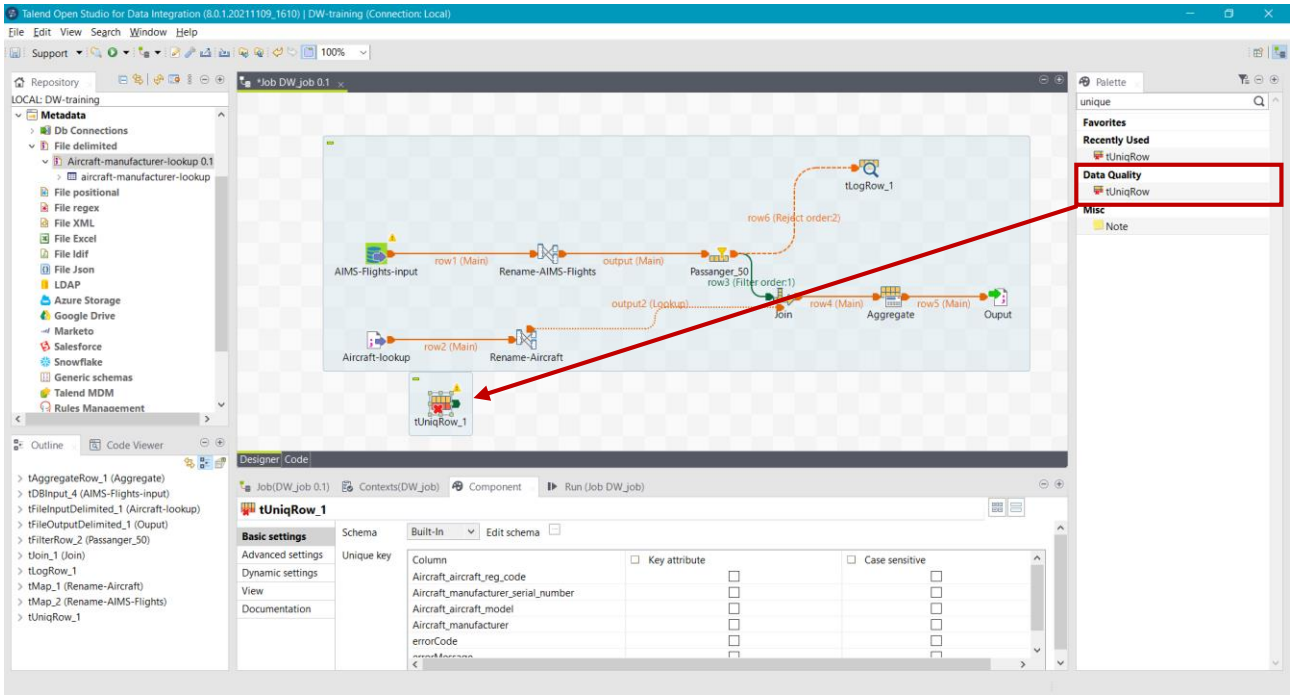
### 1. Improving Data Quality:

Data coming from the sources can be "dirty", as they can contain *duplicates*, *incomplete* or *inconsistent* entries etc. Dealing with those aspects of data quality can improve *data consistency* and *data completeness*. It is important to "cleanse" data *as early as possible in the ETL process* so that "dirty" data and their side-effects are not carried along. To achieve that using Tlaned, we can follow the steps as described bellow.

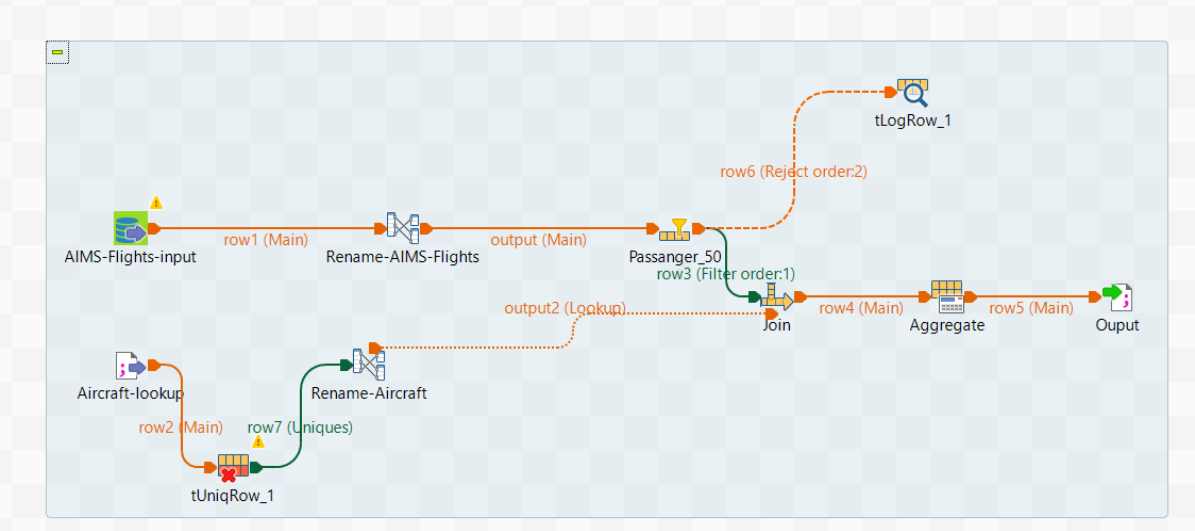
#### a. Removing Duplicates:

Duplicates are duplicate representations (i.e., repeated copies) of the same entity. For example, the same aircraft (*Aircraft_aircraft_reg_code*) might be entered more than once as a **row** entry in the aircraft-manufaturerinfo-lookup.csv file. In Talend we can use the **tUniqRow** component to remove such duplicates.

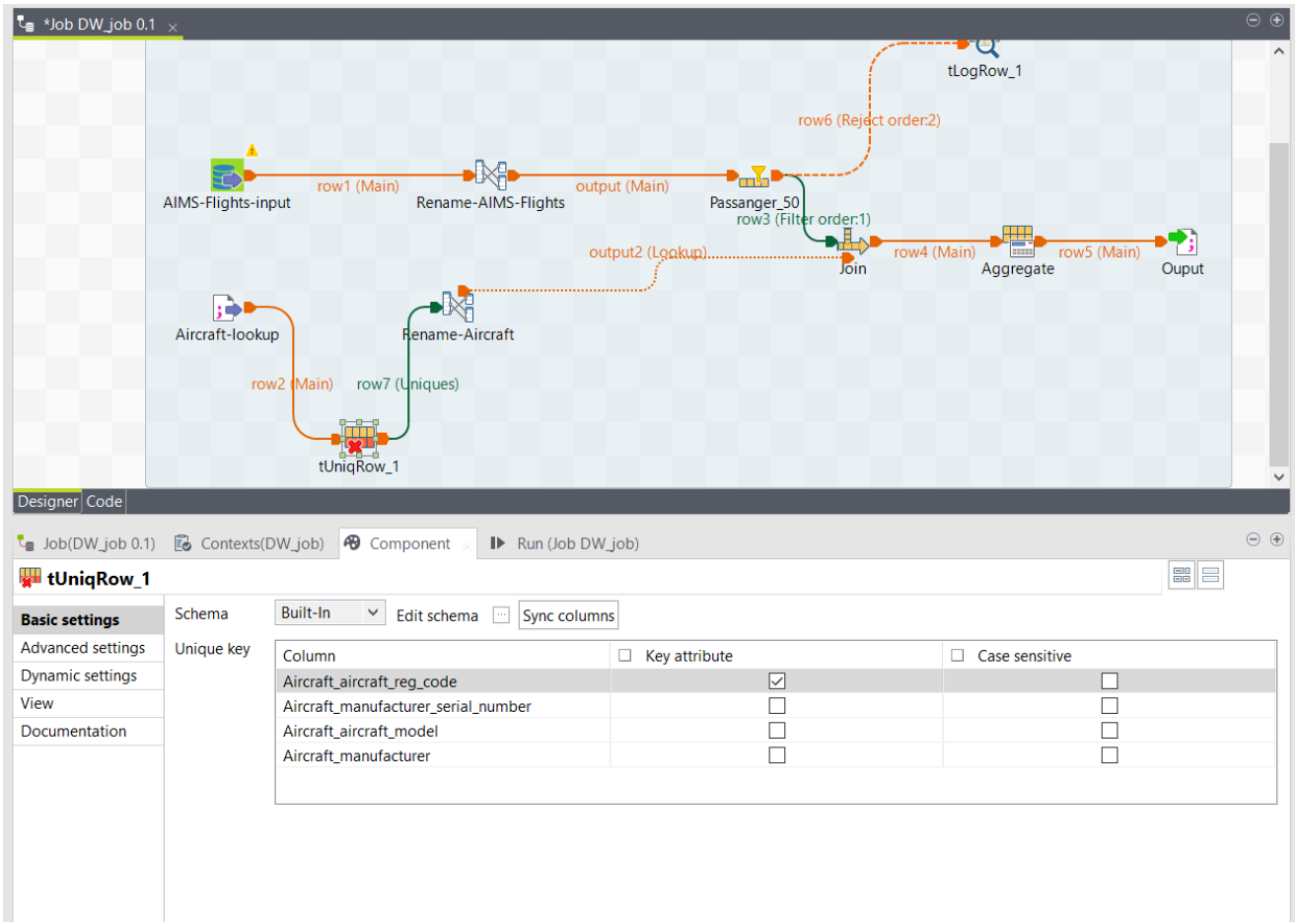
(1) From the **Data Quality** category of the design palette choose the **tUniqRow** component and drag and drop it to the design canvas as shown in the figure.



(2) To connect it to the rest of the ETL flow, remove the hop coming from the Aircraft lookup file input and replace it with a new hop coming from unique rows, as shown in the figure.




(3) We can then parametrize this component, by selecting on which attributes of the input the duplicates should be searched (e.g., over **Aircraft_aircraft_reg_code**), as shown in the figure. In addition, we can also define if the search should consider the case or not (i.e., Case sensitive).

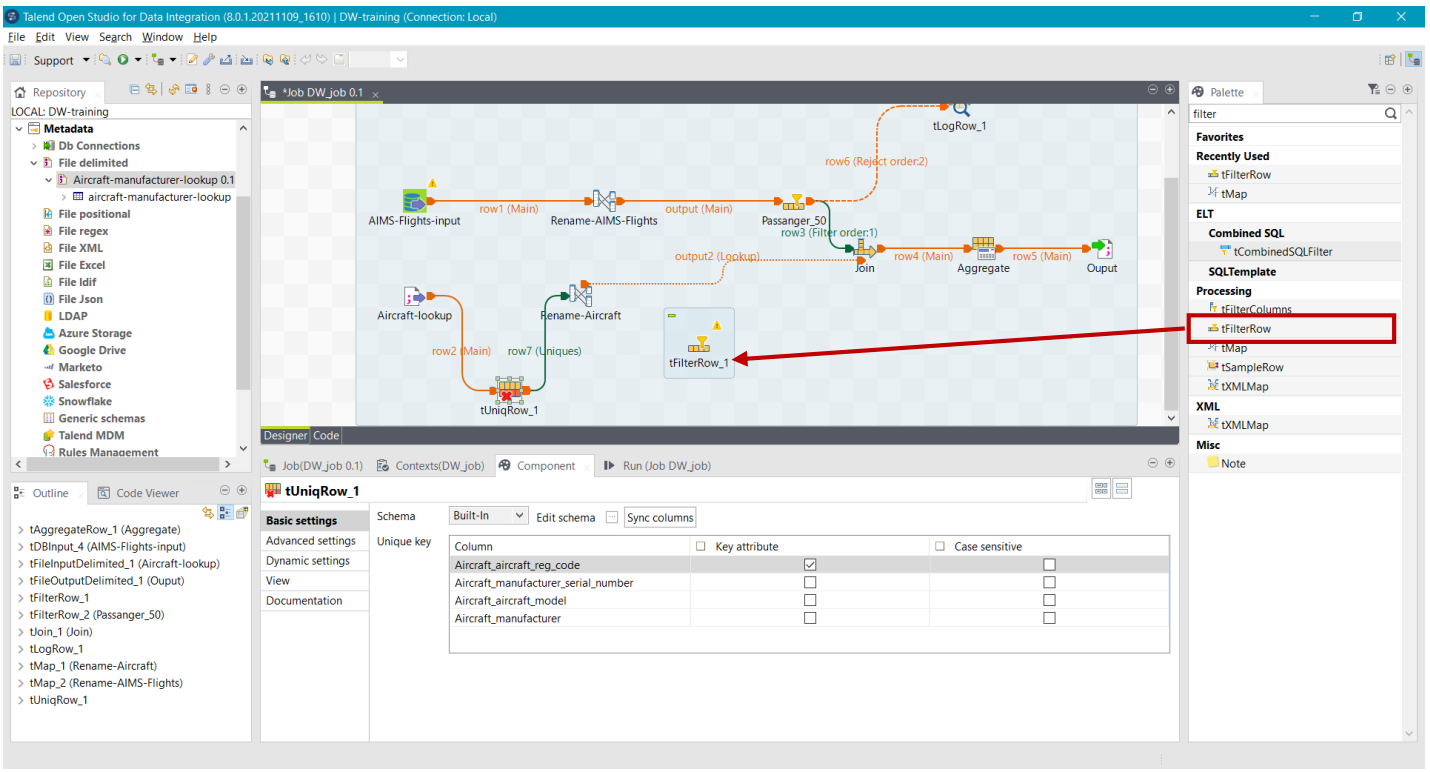



*Note: This step will only eliminate identical entries, i.e., entries with the same values for all fields. In case there are entries for the same entity but with different values, more advanced entity resolution mechanisms have to be applied.*

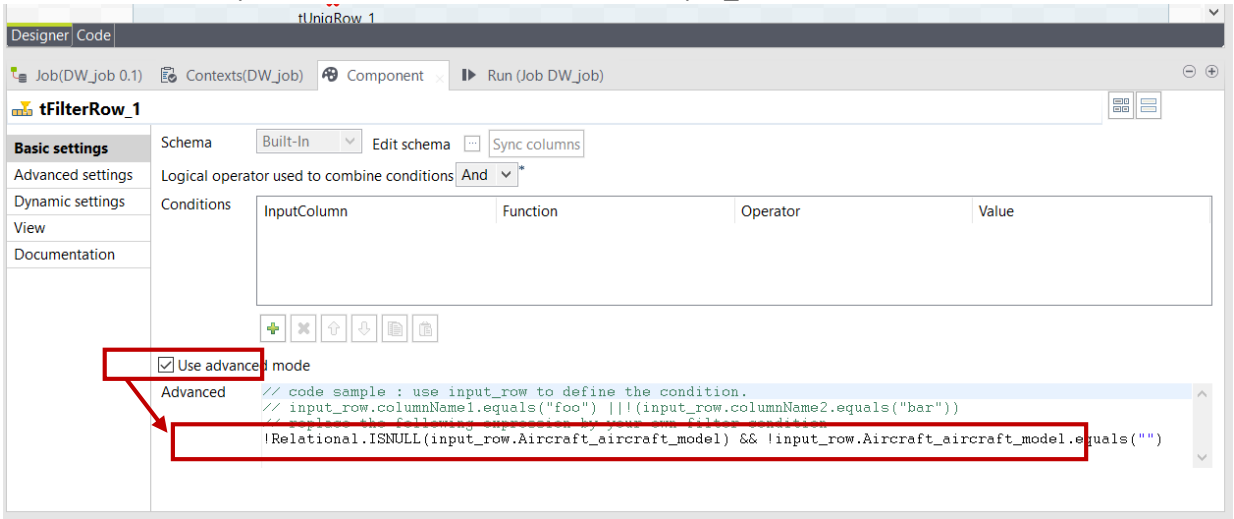
### b. Removing incomplete records:

Incomplete records are entries with incomplete or missing data, due to error at the entry, poor maintenance, processing errors or other causes. For example, in the aircraft-manufaturerinfo-lookup.csv file there could be row entries with missing important information such as the model, making these entries useless for loading our Aircraft dimension. In Talend we can use the **tFilterRow** component to remove such incomplete entries.

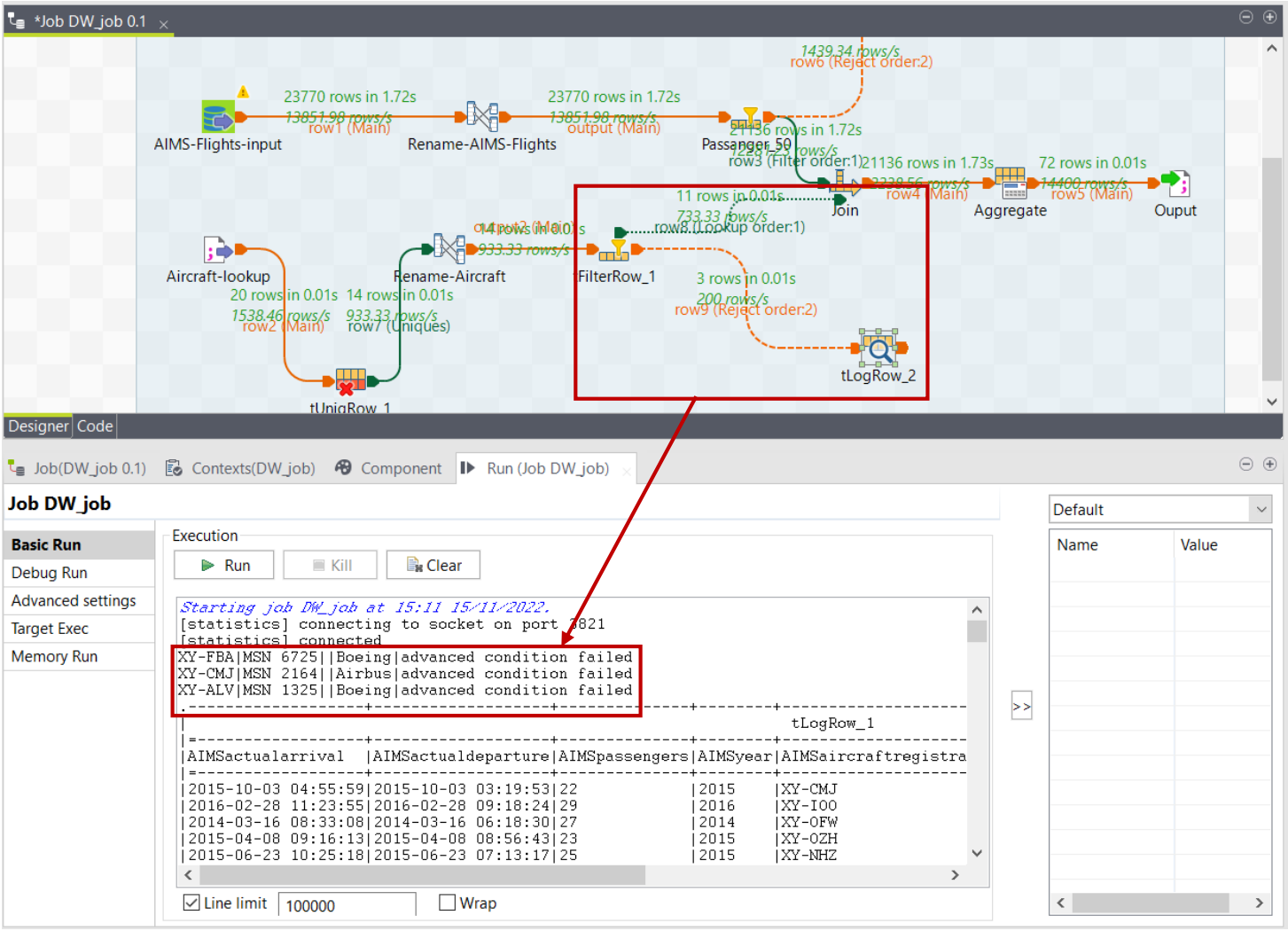
(1) From the Processing category of the design palette choose the **tFilterRow** component and drag and drop it to the design canvas as shown in the figure.




(2) We connect the filtering component to the flow and parameterize it so that only the tuples with model that is not null and not empty (“”) are considered. See figure below. Notice that for more complex filtering conditions we need to use the advanced mode (by selecting “Use advanced mode” option) and then express our filtering condition with Java code. When writing the code, we can always refer to the current row with the *input_row* variable.



**Important note:** *filtering steps like tFilterRow allows controlling the rejected rows of the step (e.g., in data cleaning operations to investigate further on the erroneous rows). In that case, along with the main output of the step toward the following step (Join), we will provide another output (Reject) of the filtering step toward a* tLogRow component. *That way as we can see in the figure below the rows that fulfill the filter condition (model is not null and not empty), 11 of them are sent to the next Join operation, while 3 of them that do not full fill the condition are sent to the tLogRow and printed in the console.*



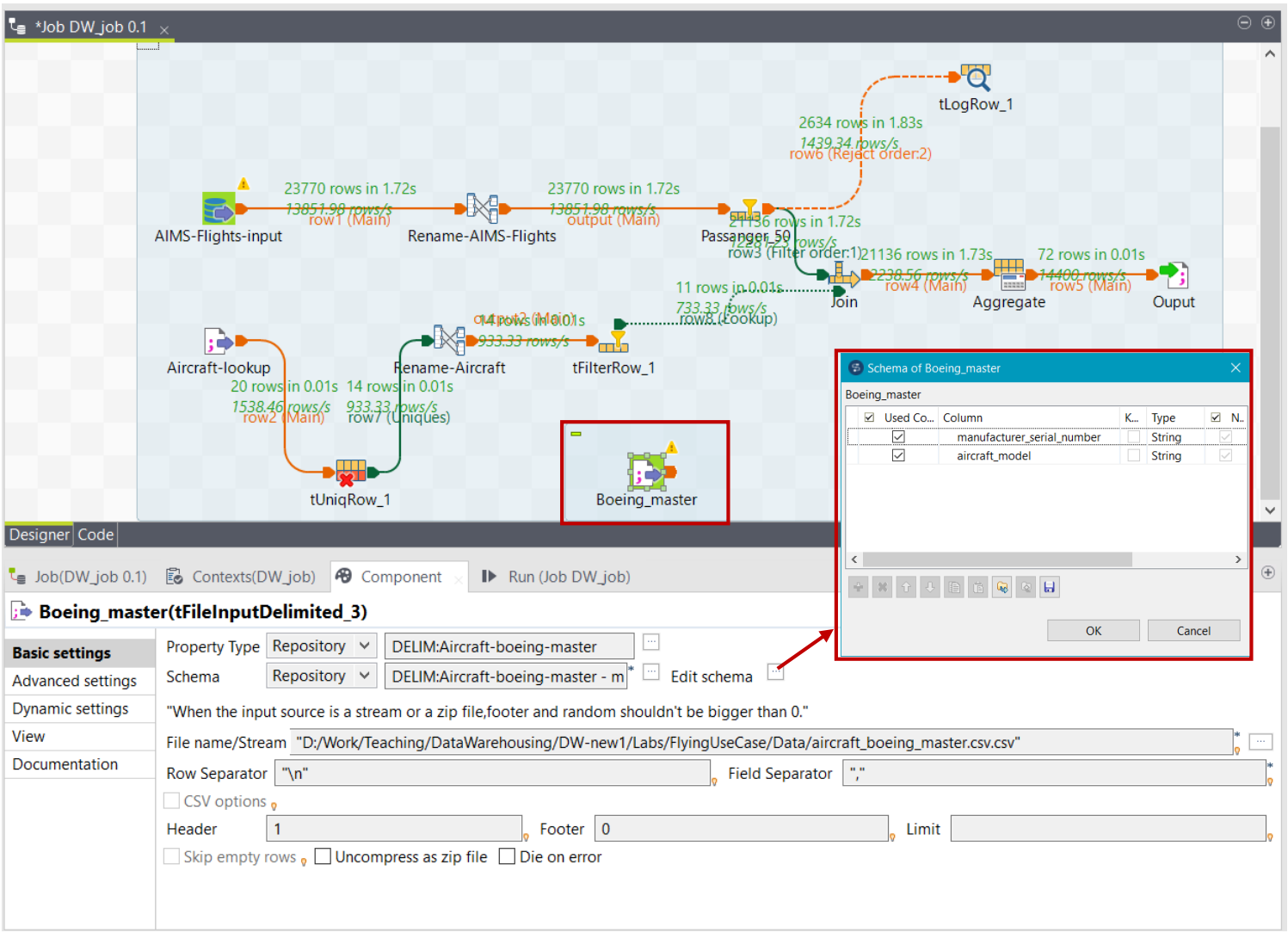

### c. Crossing multiple sources:

One technique that is frequently used in order to improve data consistency and data completeness is crossing data from one source, with data from other source(s) available. In this approach, after conducting entity resolution and deciding which entries from all available sources correspond to the same objects in the real world, the entries can be compared and completed. For example, missing values from the entries' fields can be completed with found values from another source. Other choices could be to keep only the entries corresponding to the same object that contain the most information and disregard the rest, or apply some more complex functions and aggregations to the field values coming from multiple sources (min/max/avg value, most recent value etc.).

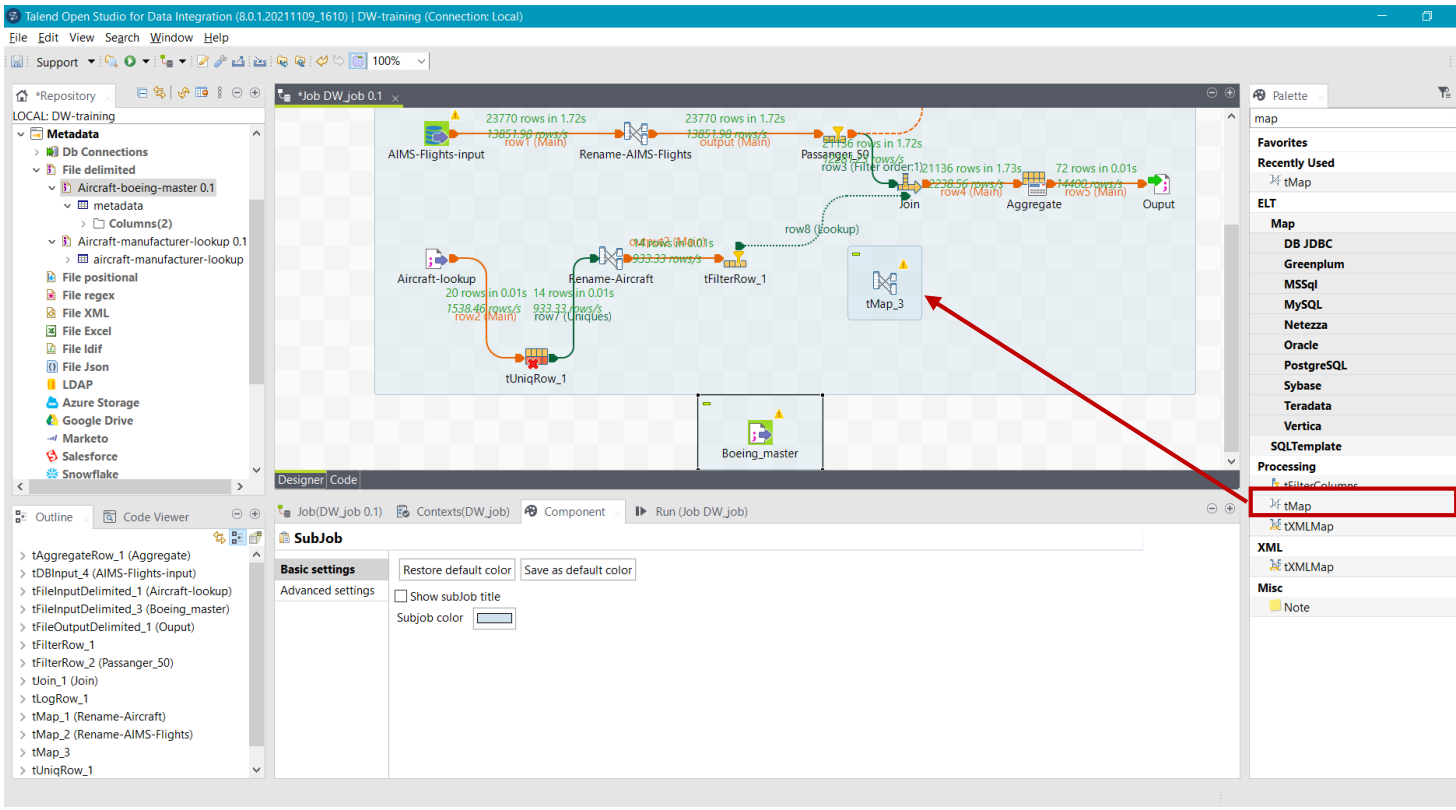
For the following example we showcase the simple first case, when there are missing values in one field that can possibly be found in the fields of another source. For this purpose, we will use another available source files that come from each manufacturer (Airbus and Boeing) and contain aircrafts and their models in CSV file (aircraft_boeing_master.csv and aircraft_airbus_master.csv). These master files include the

manufacturer_serial_number and the corresponding aircraft model and are considered a trustworthy information coming from the manufacturer directly. The manufacturer_serial_number for one aircraft is the same as the manufacturer_serial_number that is found in the records of aircraft-manufaturerinfo-lookup.csv, allowing for the records for the same user to be matched. We will assume that we are interested in having at the output of the ETL process, the correct models of the aircraft for Boeing manufacturer, so we need to cross it with Boeing master data.

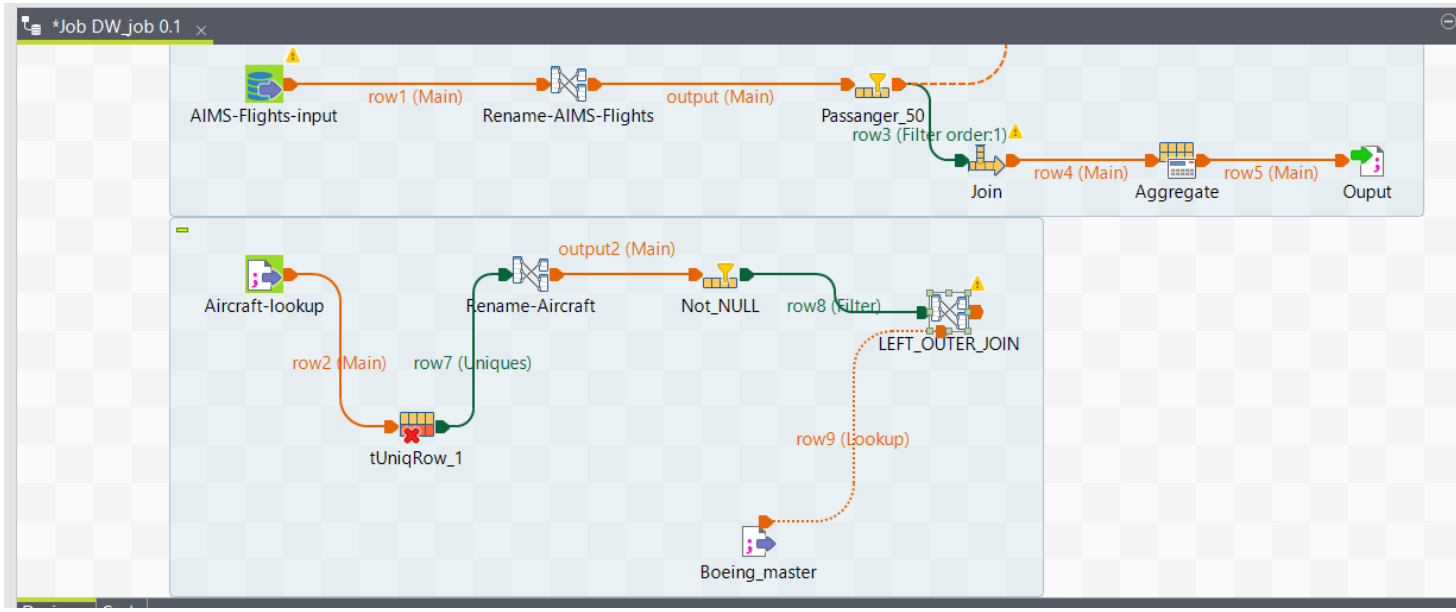
(1) In a similar manner as we defined data source inputs for the Aircraft-lookup, we need to input the csv source files using the **tFileInputDelimited** component and previously define the master files as input metadata. We drag the template to the canvas and parameterize as shown in the following figure. We can also consult the schema of the component to ensure it contains the necessary information.



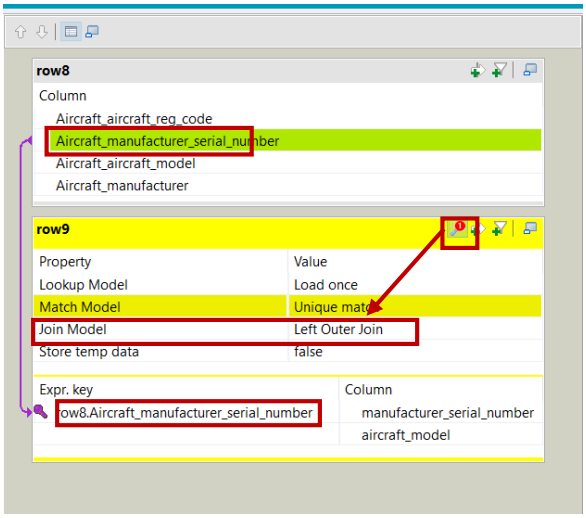

(2) Now we need to join data from the two different sources. In this case, since we want to integrate data from two different sources and keep the master data whenever it is available, we need to do the LEFT OUTER JOIN. To implement this in Talned, we need to use the **tMap** component. From the **Processing** category of the design palette choose the tMap compoenent template and drag and drop it to the design canvas as shown in the figure.




(3) Then we reconnect the flow by connecting the Boeing_master input and Not_NULL filter steps to the added tMap step.



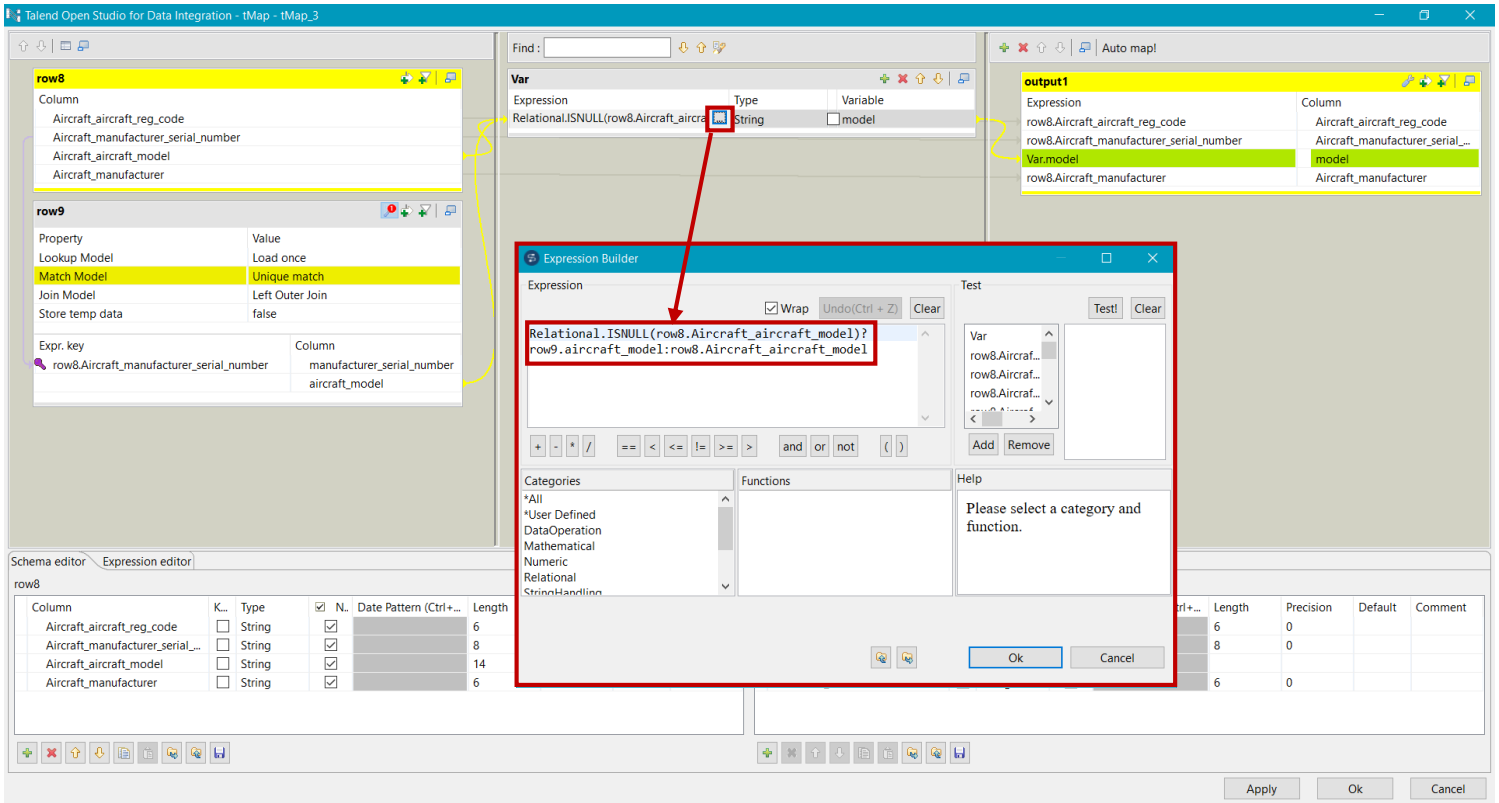

(4) We should then parameterize the tMap step to properly define the join attributes. In this case, we need to drag and drop the **Aircraft_manufacturer_serial_number** column from the main input to the **manufacturer_serial_number** from the master file and create the connecting at the input as shown in the figure. We also need to select the “Left Outer Join” option as a Join model, which defines a left outer join. See figure below.




(5) In addition, within the same tMap compoenent, we need to provide a function that will check for null values at the model attribute that has derived from the Aircraft-lookup input source (**Aircraft_aircraft_model**) and if there are any null or emptry values, replace them with the model values (**aircraft_model**) coming from the Boeing master input source (**Boeing_master**). For this reason, from the in the middle part of the tMap component settings we need to define an expression for a new variable called simply **model** that will in the case that **Aircraft_aircraft_model** is null or empty, take the value of the **aircraft_model** or otherwise keep the existing value.

```
( Relational.ISNULL(row8.Aircraft_aircraft_model) ||
  row8.Aircraft_aircraft_model.equals("") ) ?
  row9.aircraft_model : row8.Aircraft_aircraft_model
```

(6) We finally need to map both the input attributes and the generated new model variable. See figure below.

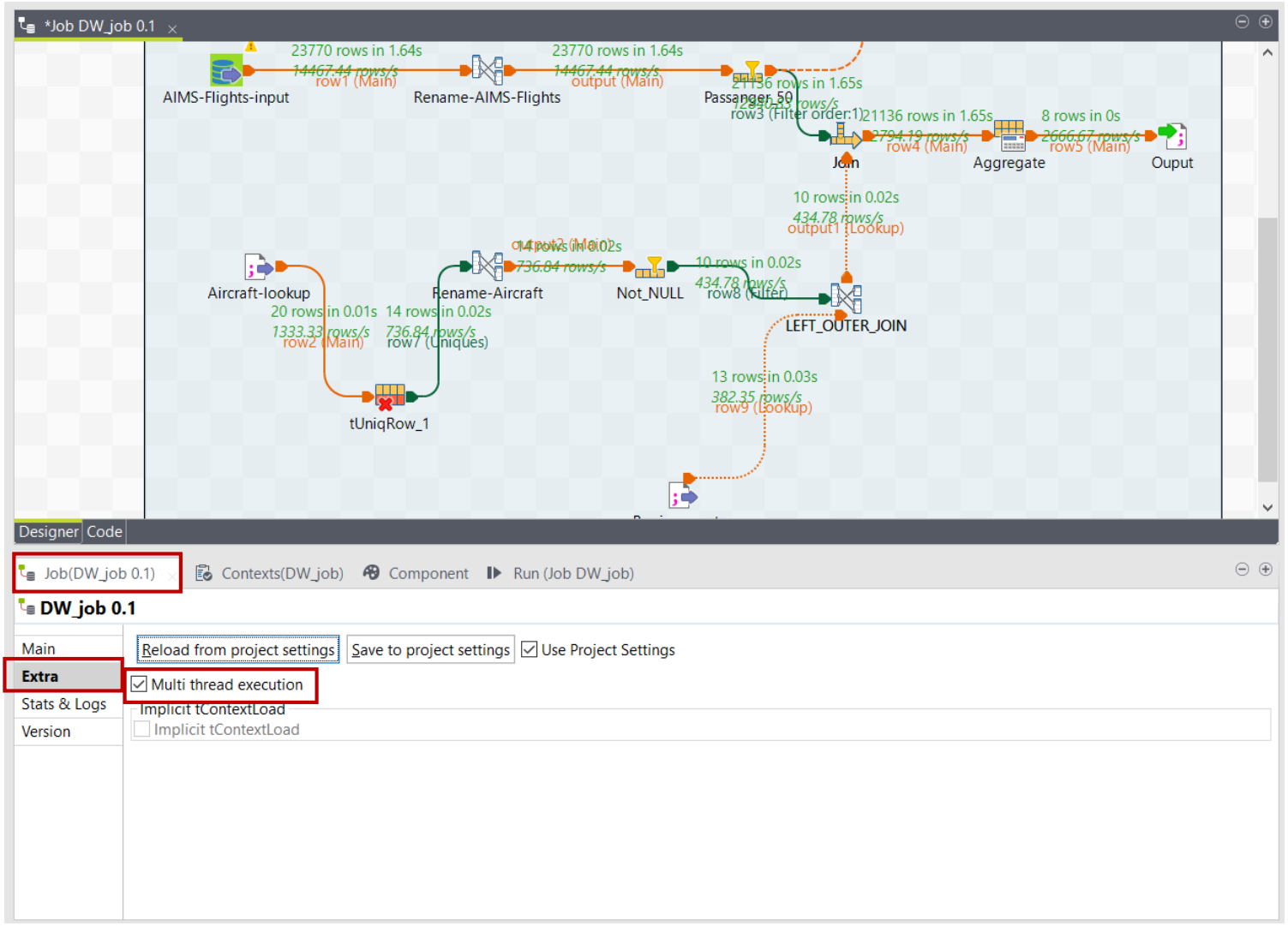


## 2. Improving Performance:

The performance of the ETL process refers to the execution efficiency of the process, in terms of time efficiency, throughput, resource utilization etc. In this example we will consider the time efficiency of the process, which is the degree of low response times, low processing times and high throughput rates. To achieve that using Pentaho Data Integration tool, we have various options, some of which are described below.

### a. Parallelizing the flow:

Parallelization refers to the concurrent execution of activities belonging to the same process, thus resulting in faster execution time and higher throughput of the process. In Talend, this can be achieved by executing some jobs or some subJobs in parallel, i.e., using the multi thread execution. To do so, within the job itself, click on the **Job** tab in the control panel and then **Extra.** There, you should select the "**Multi thread execution**" option, as shown in the figure.

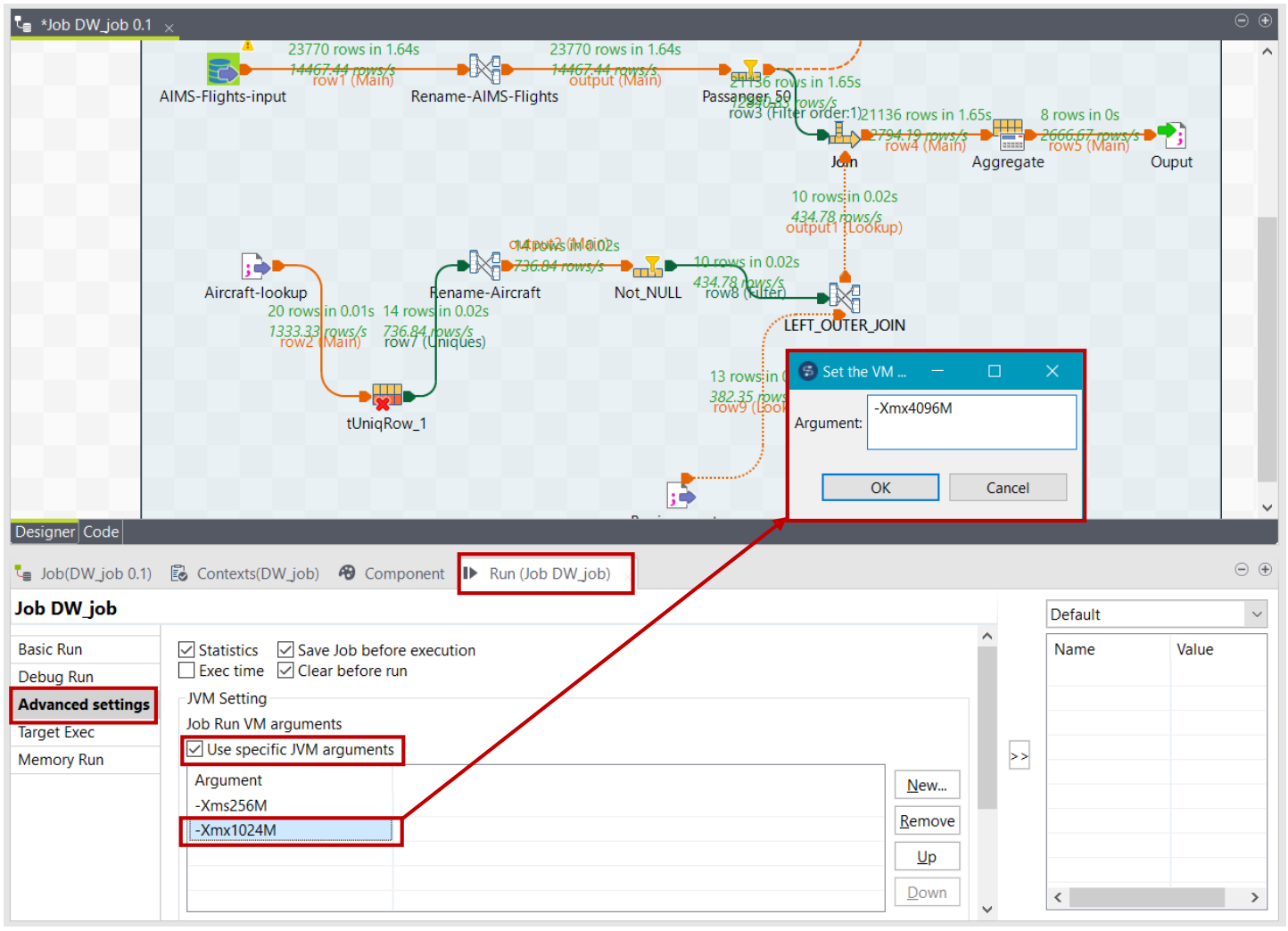



**Note:** This choice should always depend on the complexity and the real cost of the operations being executed within each subJob.

### b. Assigning more memory:

A process with more available memory can faster execute operations such as sorting and joining, since more entries to be processed can be at the same time “loaded” on memory, where algorithms are applied. One way to achieve more memory allocation in Talend, is by increasing the java heap size of the Job run.

Within the same Job, click on the **Run** tab and then **Advance settings,** and select the option “Use specific JVM arguments”. Once selected, we can change the existing or introduce new JVM arguments. In this case, we will change the maximum java memory to 4GB by modifying the argument -Xmx to -**Xmx4096,** as shown in the figure, so that we can allow processing many entries (~100.000) from Flights.



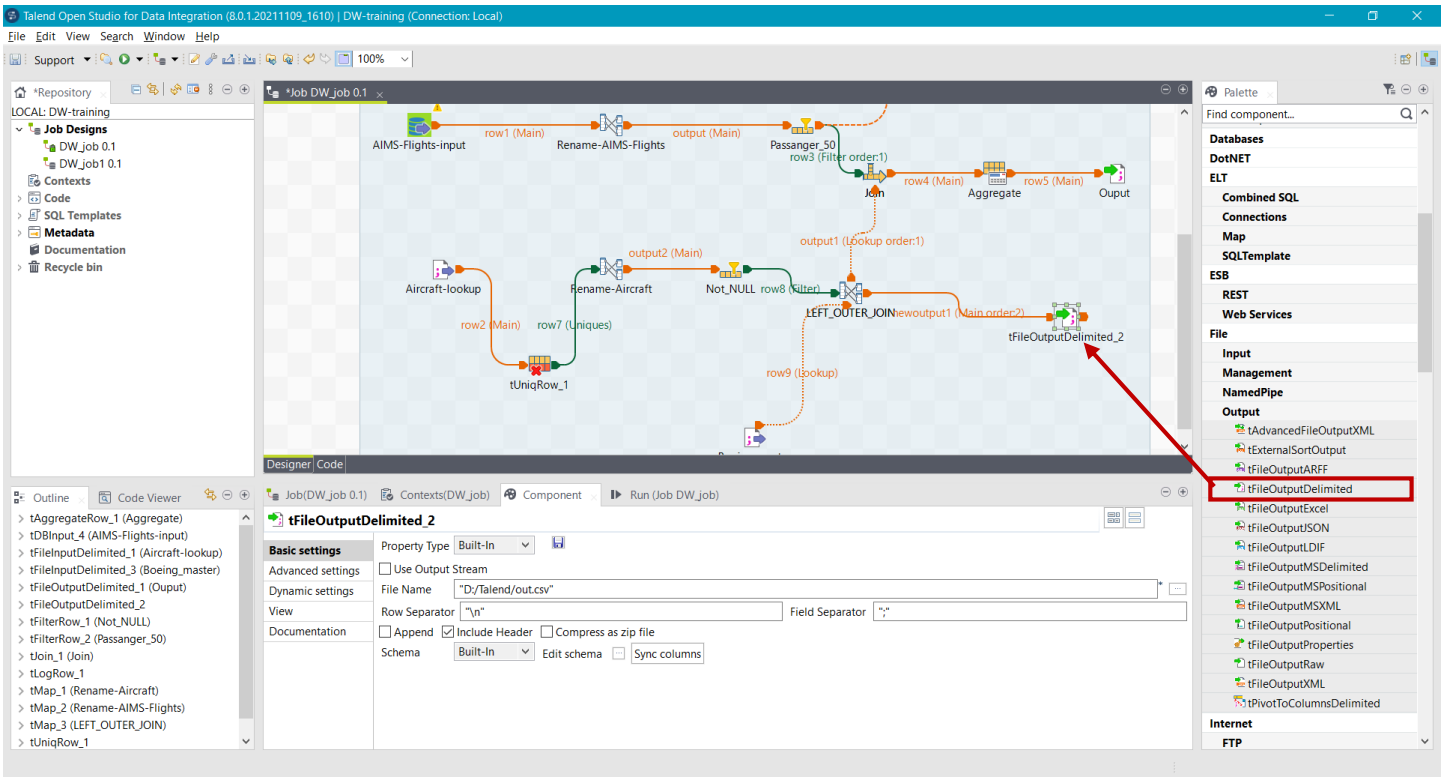
### 3. Improving Reliability:

Reliability of an ETL process is the degree to which the ETL process can maintain a specified level of performance for a specified period of time. It includes *robustness*, the degree to which the process operates as intended despite unpredictable or malicious input and *recoverability*: the degree to which the process can recover the data directly affected in case of interruption or failure. To improve this quality dimension using the Talned tool, we can add recovery points and handle errors, as described below.

#### a. Adding recovery points:

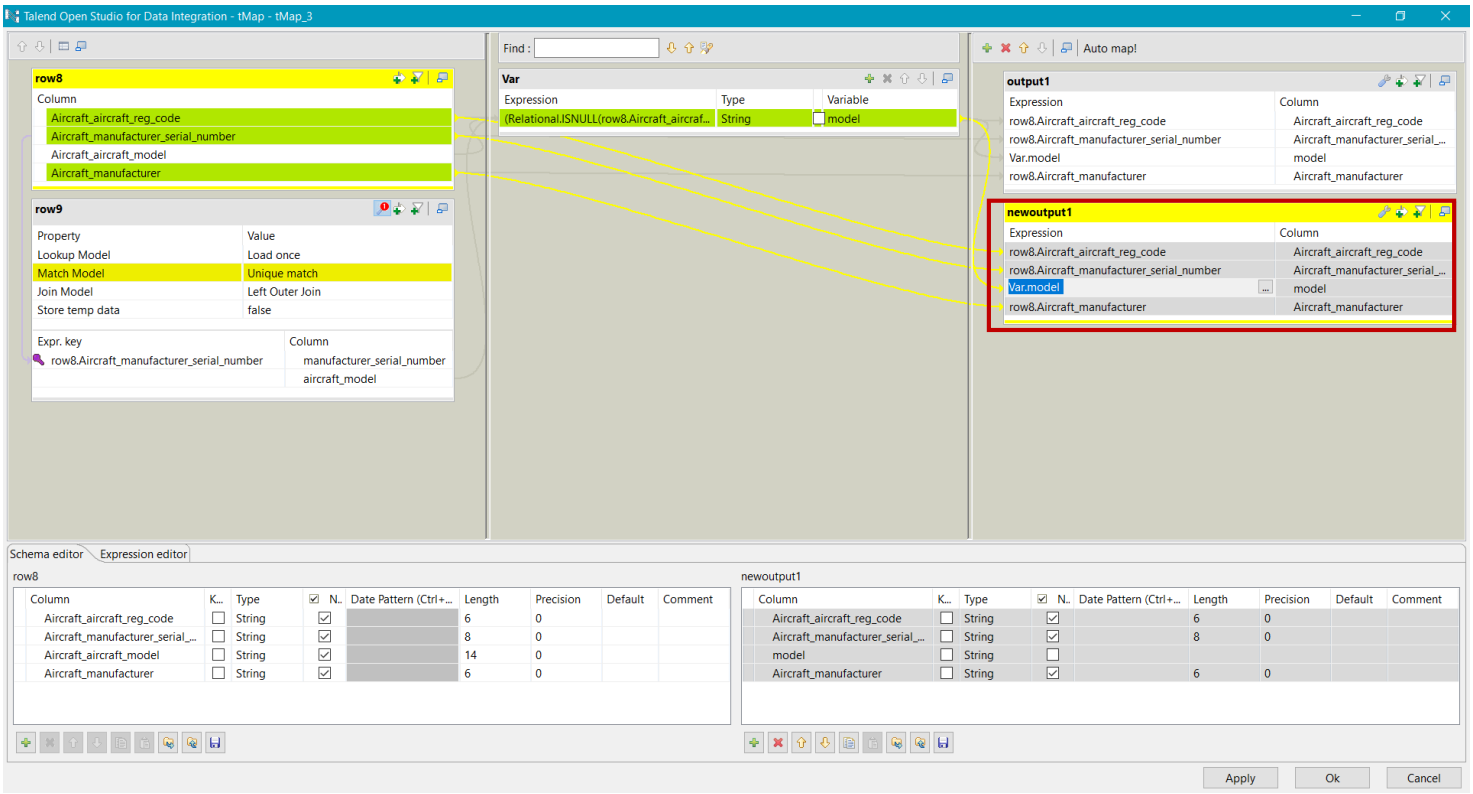
**Recovery points** are execution "checkpoints" that store intermediary execution information so that a process can resume in case of interruption or failure. In Talend this can be performed by adding output files after specific tasks. It would be a good idea to add checkpoints after tasks that are time-consuming\computationally intensive, such as group-by operations, calculations, aggregations etc., in the ETL process.

(1) From the "**File -> Output**" category of the design palette choose the **tFileOutputDelimited** component and drag and drop it to the design canvas and connect it to the previous LEFT_OUTER_JOIN operation (tMap component[1]), as shown in the figure.




[1] **Important note**: in Talend only tMap component allows multiple copies of the same output. Other operations like Join and Filter have different semantics for different output flows (e.g., rows that fulfill the Filter condition vs. those that do not, or rows that are matched in a Join vs. those that are not).

(2) After connecting it to the tMap component, we should give a new unique name of the tMap output (when prompted) and then inside the previous tMap component we must drag and drop the same input or derived attributes to the new output as well, as shown in the following pictures.




(3) Lastly, we can configure the CHECKPOINT operation (**tFileOutputDelimited**), by providing the file path and name of the CSV file that will be stored, row and field separator, if append are allowed to the file and if we want to include a header to the created file, see Figure below. We can in addition modify the schema if necessary, although the schema is synchronized automatically to the output provided from the previous tMap component.