Step Implementation

This section deals with the interesting bits involved in fully implementing a step. A step that implements the ETLStep interface will be recognised as a Data Transformation step. However, Yellowfin has partially implemented the interface and provides a few abstract classes to make implementation easier. We recommend extending these abstract classes:

Class	Purpose
AbstractETLRowStep	Extend this class if the step is required to process data row by row and output data as and when it is processed. Filter and Calculated Field are examples of row steps.
AbstractETLCachedStep	This class should be extended if the step is required to accumulate data before it can begin processing. For example, Merge and Union are cached steps.
AbstractETLInlineRowStep	Inline transforms are expected to be row steps. Extend AbstractETLInlineRowStep to implement an inline transform.

Types of Step Implementation

Depending on which abstract class is used different methods need to be implemented. There are some common aspects though. For ease of understanding, the implementation details are split into three sections:

- **Step user interface implementation:** Methods used to implement the step's UI.
- **Processing implementation:** Methods used to define the step's processing logic.
- **Helper methods:** Additional methods involved in implementing a step. These include:
  - **Input/Output methods:** Handles a step's input and output flow.
  - **Field methods:** Deals with a step's default or output metadata fields.
  - **Configuration methods:** Methods that configure a step's options.
  - **Data processing methods:** Methods used to manipulate data during a step's execution.
  - **Error handling methods:** These are used to handle errors.
  - **Other methods:** Some additional methods that might be of interest in step implementation.