Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

Wirtschaft

Modern Data Engineering in the Cloud

Dr. Christian Dollfus
Dozent

T direkt +41 41 228 22 54
christian.dollfus@hslu.ch

Luzern

Dr. Pavlin Mavrodiev
Dozent

T direkt +41 76 733 61 66
pavlin.mavrodiev@alumni.ethz.ch

PART 3 : Data Engineering with the ETL/ELT -Tool PDI introduction

Data Engineering with PDI

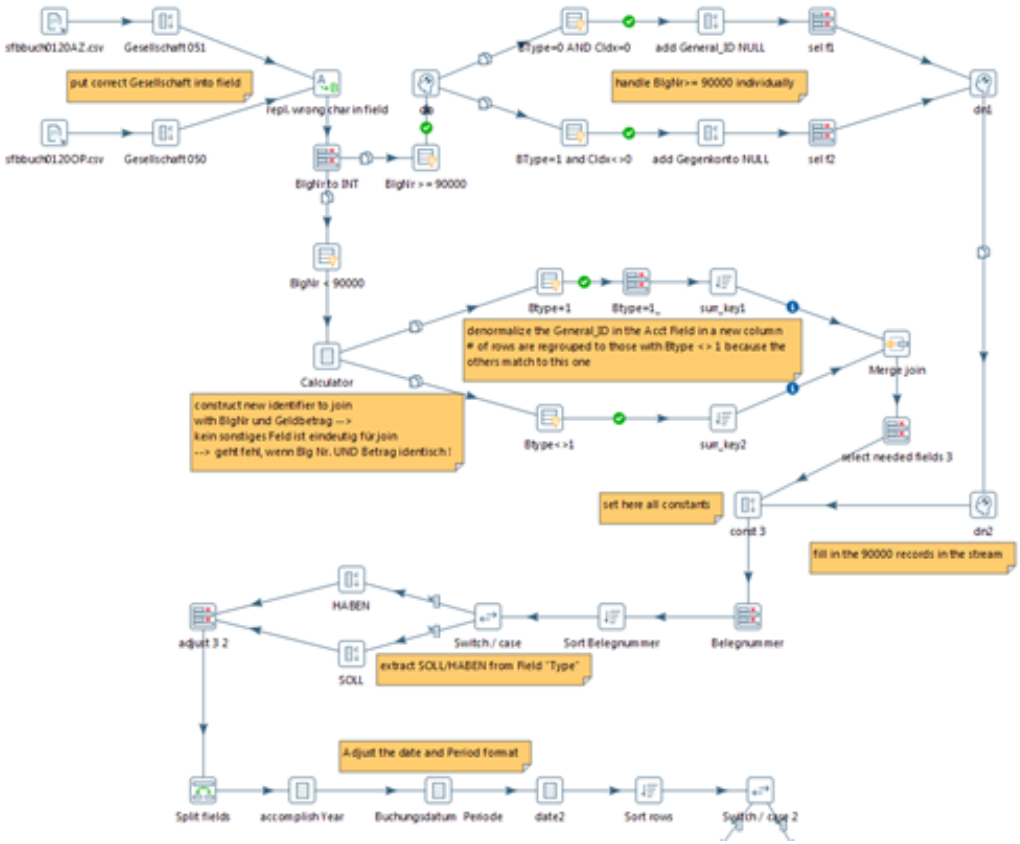
- Introductory words
- What is Data Engineering?
- Motivation and Value Proposition - a historic overview
- How does Data Engineering look like in many companies?
- What are features and advantages of workflow-based ETL?
- Motivation for the use of ETL tools.
- Data Engineering and Business Process Automation - similarities and differences

INTRODUCTION to ETL/ELT

- Processing Data is a consecutive **Workflow** of tasks that needs to be run; either continuously or in batch (every hour, every day, every week, ..)

- Data Processes can be more complex than a “linear” series of tasks: split into 2 tasks, performing a “loop” and reenter a stream :

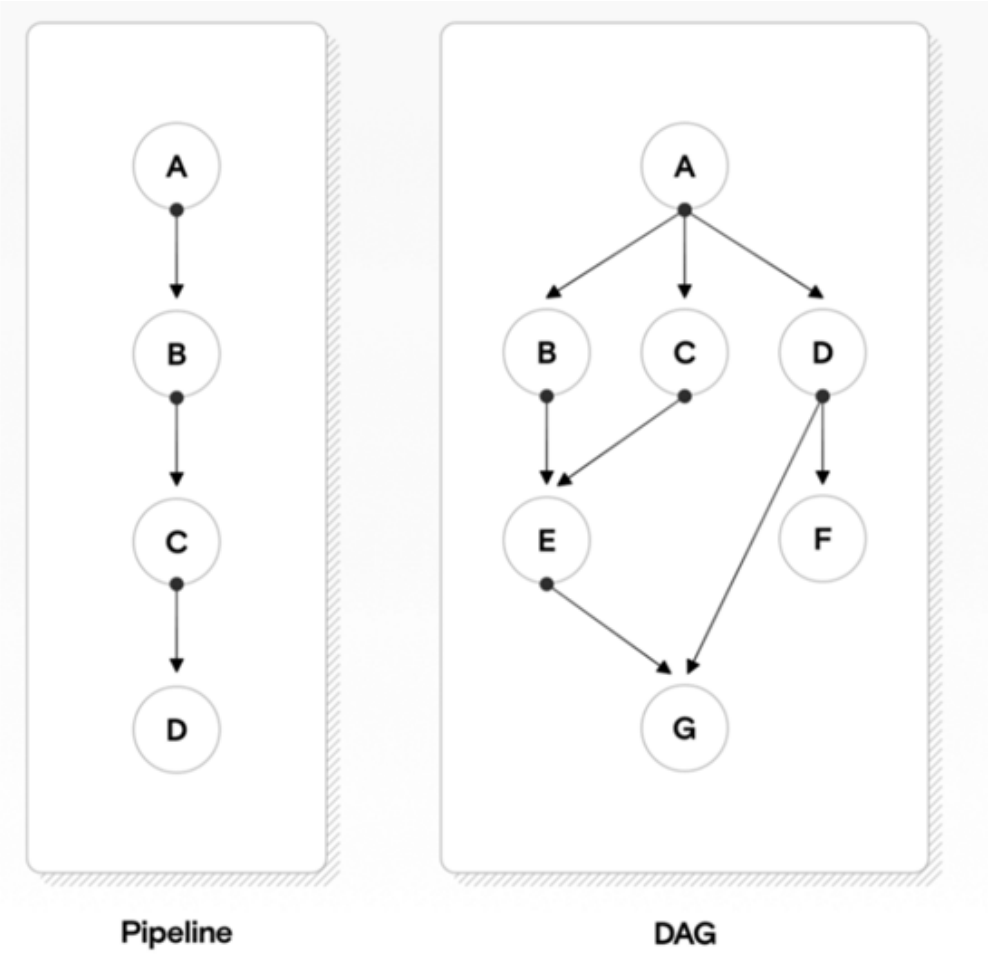
Directed Acyclic graph: DAG



Workflow orchestration tools allow you to define DAGs by specifying all of your tasks and how they depend on each other. The tool then executes these tasks on schedule, in the correct order, retrying any that fail before running the next ones. It also monitors the progress and notifies your team when failures happen.

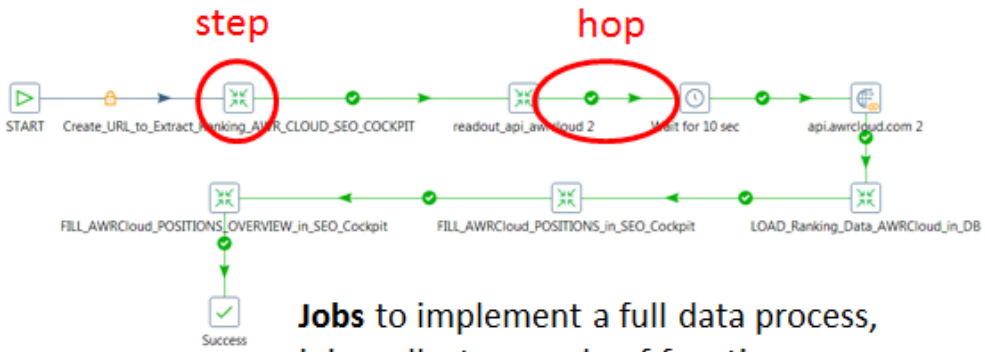
Data Engineering with PDI

Overall, the focus of any orchestration tool is ensuring **centralized, repeatable, reproducible,** and **efficient** workflows: a virtual command center for all of your automated tasks.

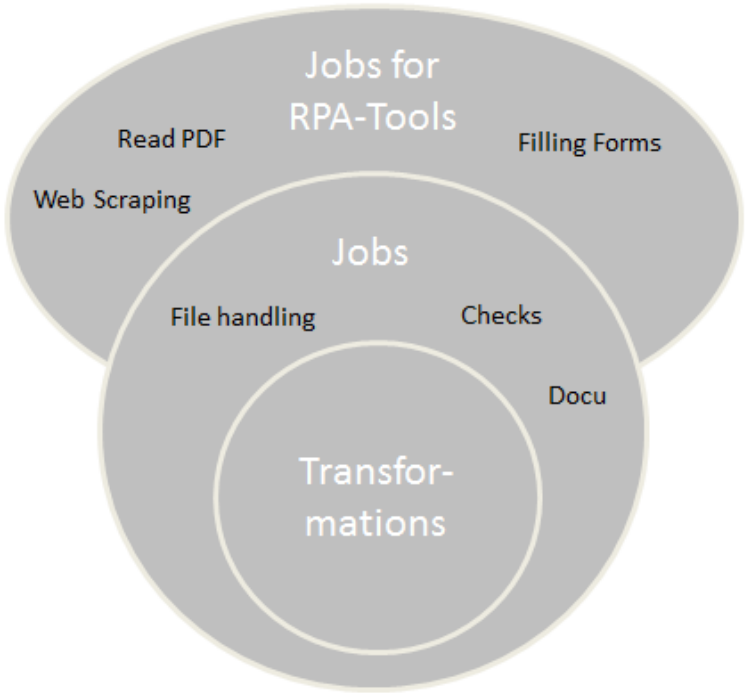


Source: https://towardsdatascience.com/airflow-vs-luigi-vs-argo-vs-mlflow-vs-kubeflow-b3785dd1ed0c Markus Schmitt

Data Engineering with PDI



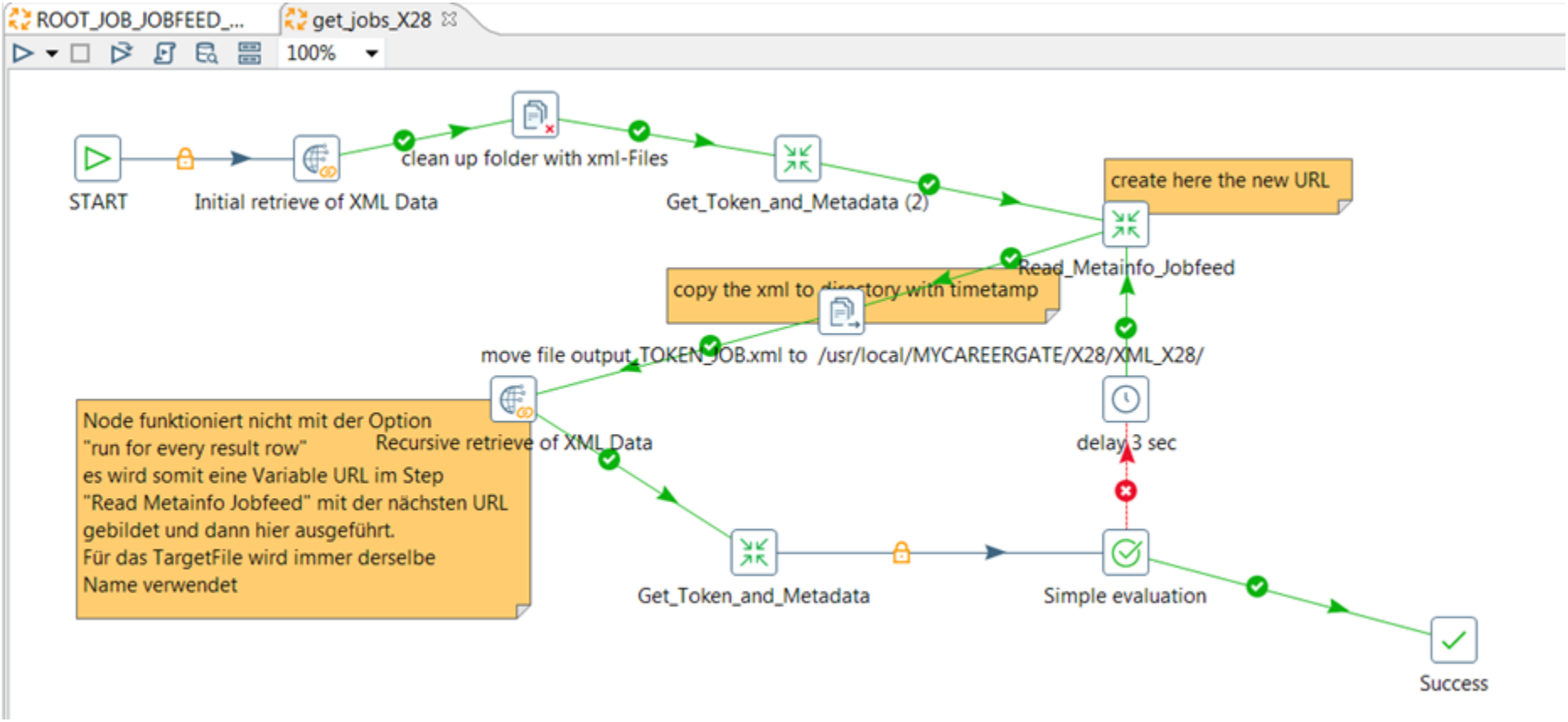
Jobs to implement a full data process, jobs collect a couple of functions (transformations) or other sub-jobs
They are the sorrounding functions of a data pipeline: filehandling, downloading data, Webservice interfaces a.s.o.



Transformation is a directed graph with logicasl tasks and can be built as a network. The elements are the STEPS.
Transformations are *dataflows* in the essence

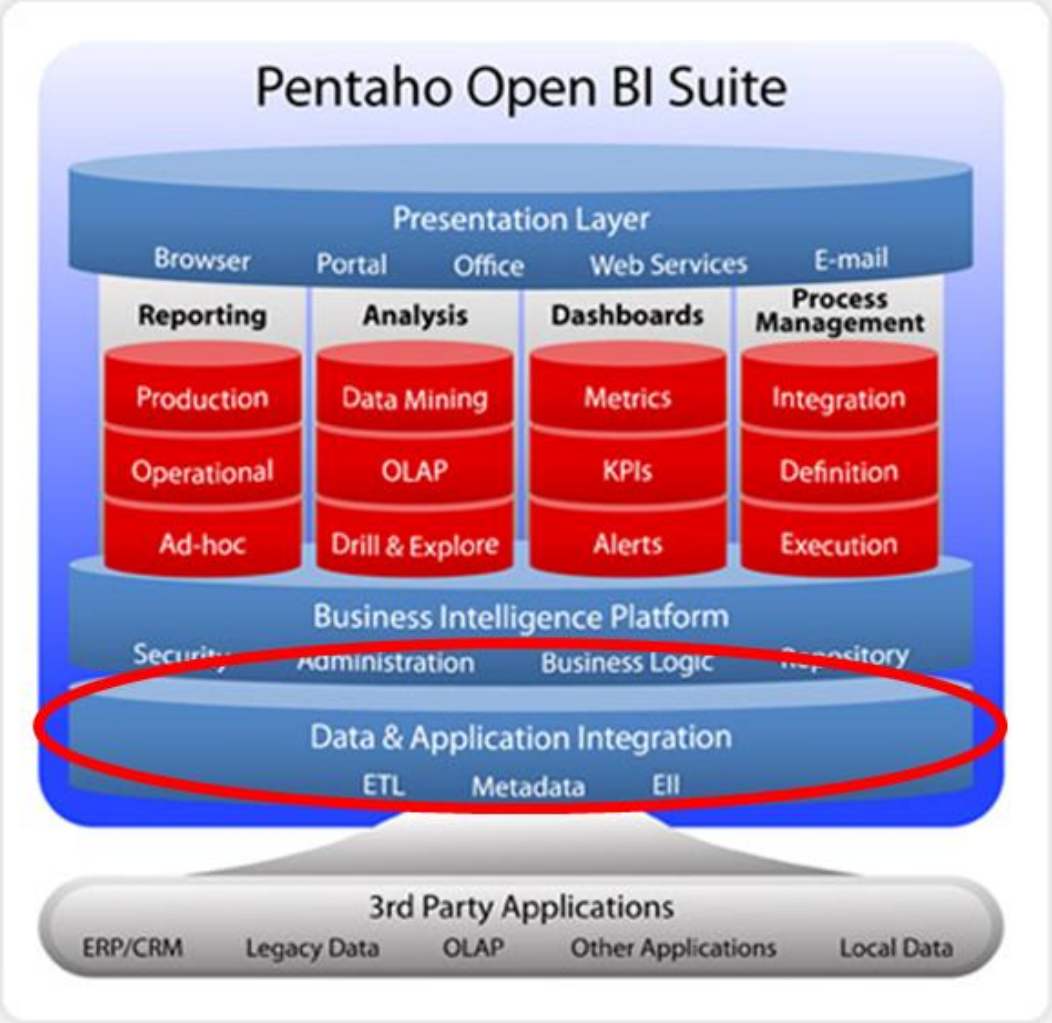
https://help.pentaho.com/Documentation/7.0/0L0/0Y0/030/010

Data Engineering with PDI



Data Engineering with PDI

We use only one part of the Whole Hitachi Pentaho/Lumada suite: The Pentaho Data Integration part (former «KETTLE»)



INTRODUCTION to ETL/ELT

APANCHE HOP: https://hop.apache.org/

1) Jobs and Transformations

There is an Apache Software Foundation incubation process starting soon, using a FORK of PDI:

A lot has changed behind the scenes, but don't worry, if you're familiar with Kettle/PDI, you'll feel right at home immediately

https://hop.apache.org/manual/latest/

https://hop.apache.org/tech-manual/latest/hop-vs-kettle/index.html

IMPORT KETTLE (PDI) PROJECTS IN APACHE HOP (INCUBATING)

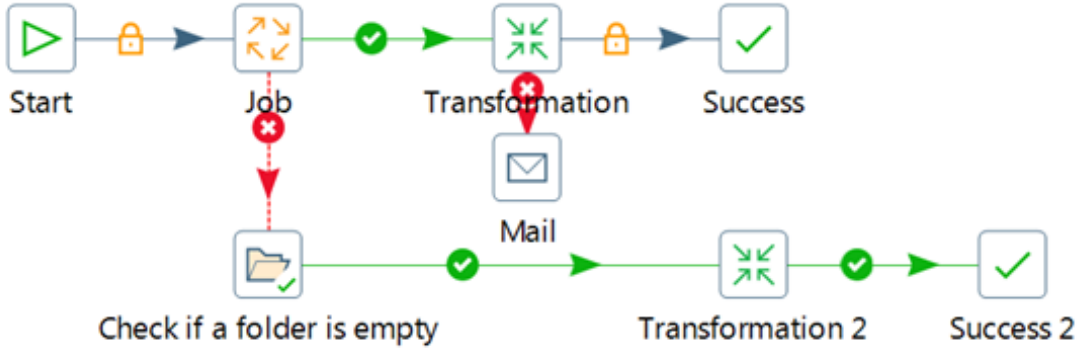
Data Engineering with PDI

Help and Hints in Pentaho

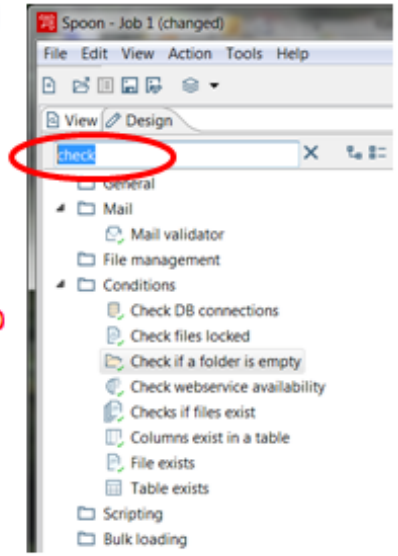
Comprehensive Documentation On the Web from Hitachi Vantara	https://help.pentaho.com/Documentation/9.0
Forum for Pentaho user	https://forums.pentaho.com/forums/135-Pentaho-Data-Integration-Kettle/
Comprehensive collection of examples of functions, Jobs und Transformations	INSTALL_DIRECTORY\NANE_PDI_DIRECTORY\data-integration\samples\transformations D:\Programme\pdi-ce-9.0.0.0.0-4323\data-integration\samples\transformations
REGEXP Tester	https://regex101.com/
Other ressources	https://etl-tools.info/pentaho-tutorial.html

Data Engineering with PDI

Jobs and Transformations can be NESTED hierarchically (but not transformations in themselves): A Job contains several transformaitons and jobs (subjob)



Kann Begriff suchen



On Hobs in Jobs you can incluse a conditional logic of the pipeline flow:

Möglichkeit	Beschreibung
Bedingungslos	Gibt an, dass der nächste Jobeintrag unabhängig vom Ergebnis des ursprünglichen Jobeintrags ausgeführt wird
Folgen Sie, wenn das Ergebnis wahr ist	Gibt an, dass der nächste Jobeintrag nur ausgeführt wird, wenn das Ergebnis des ursprünglichen Jobeintrags wahr ist. Dies bedeutet eine erfolgreiche Ausführung wie "Datei gefunden", "Tabelle gefunden", ohne Fehler usw.
Folgen Sie, wenn das Ergebnis falsch ist	Gibt an, dass der nächste Jobeintrag nur ausgeführt wird, wenn das Ergebnis des ursprünglichen Jobeintrags falsch war. Dies bedeutet, dass die Ausführung nicht erfolgreich war, die Datei nicht gefunden wurde, die Tabelle nicht gefunden wurde, Fehler aufgetreten sind usw.

Data Engineering with PDI

Transformations

Jobs

Jobs

INPUT Steps

Transformations

TRANSF Steps

Transformations

OUTPUT Steps

Transformations

The following slides are giving a structured overview over the different nodes in PDI grouped in this structure above:

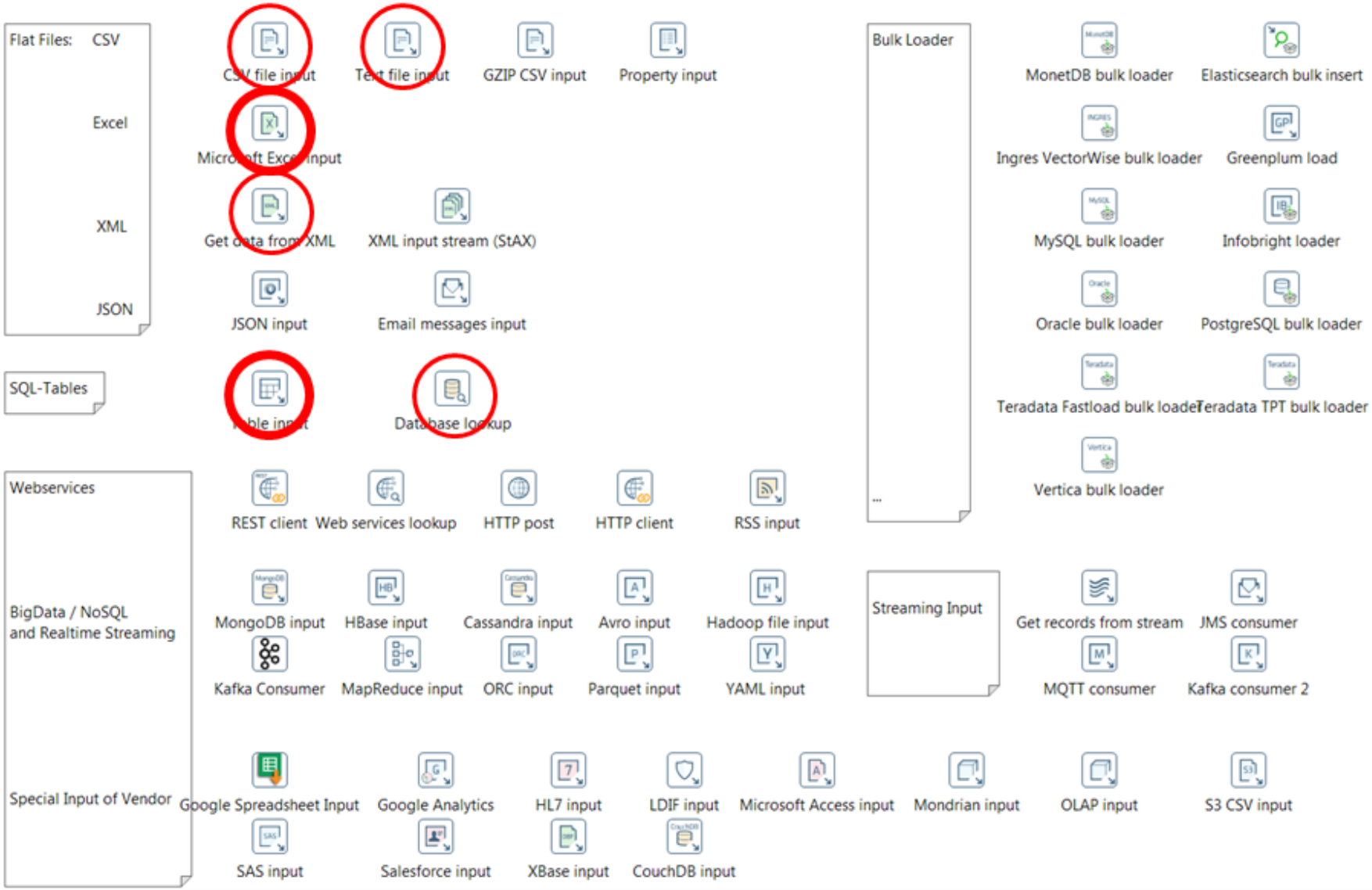
denote important steps

denote VIP's

INTRODUCTION to ETL/ELT

Transformations

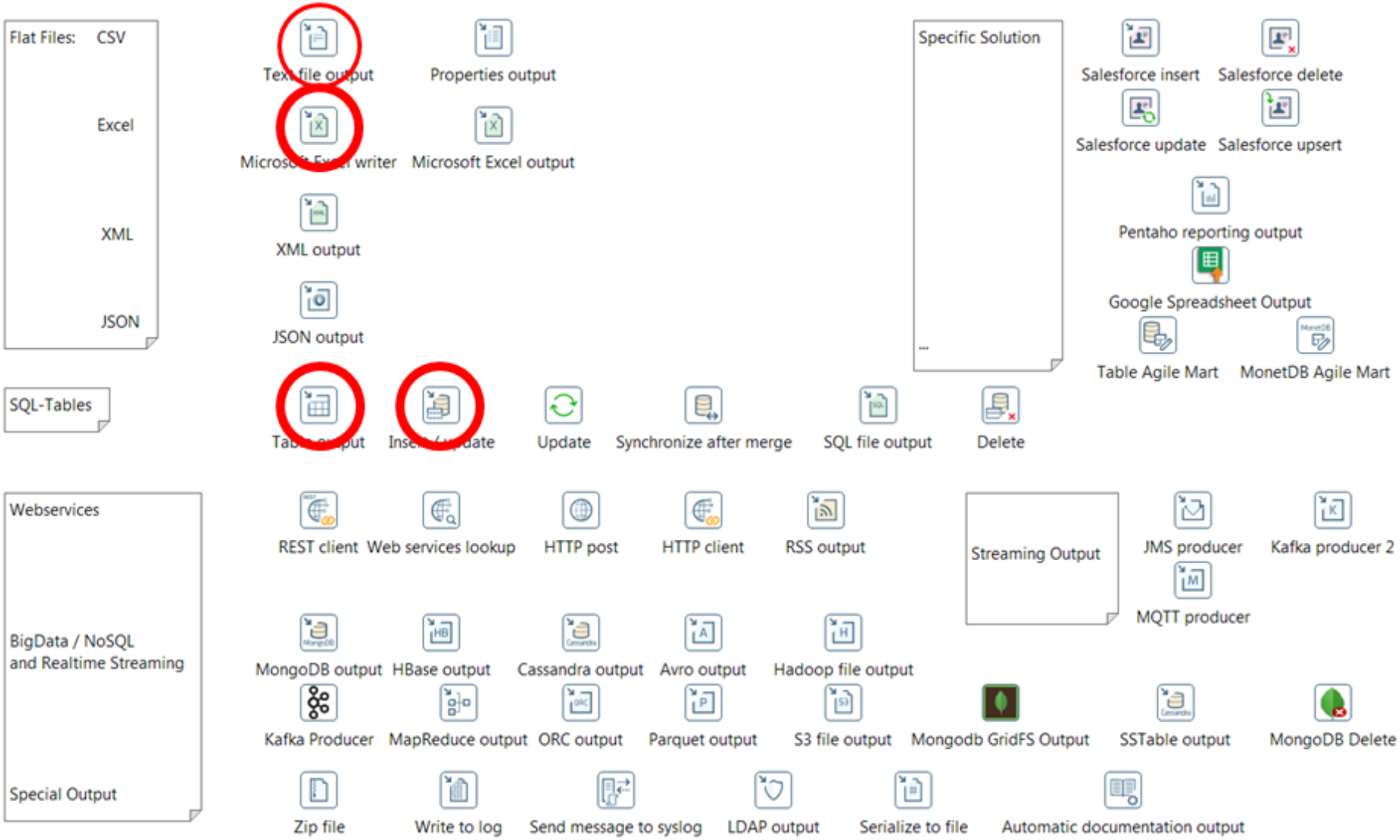
2.1) **Handling INPUT in Data Pipelines:**



INTRODUCTION to ETL/ELT

Transformations

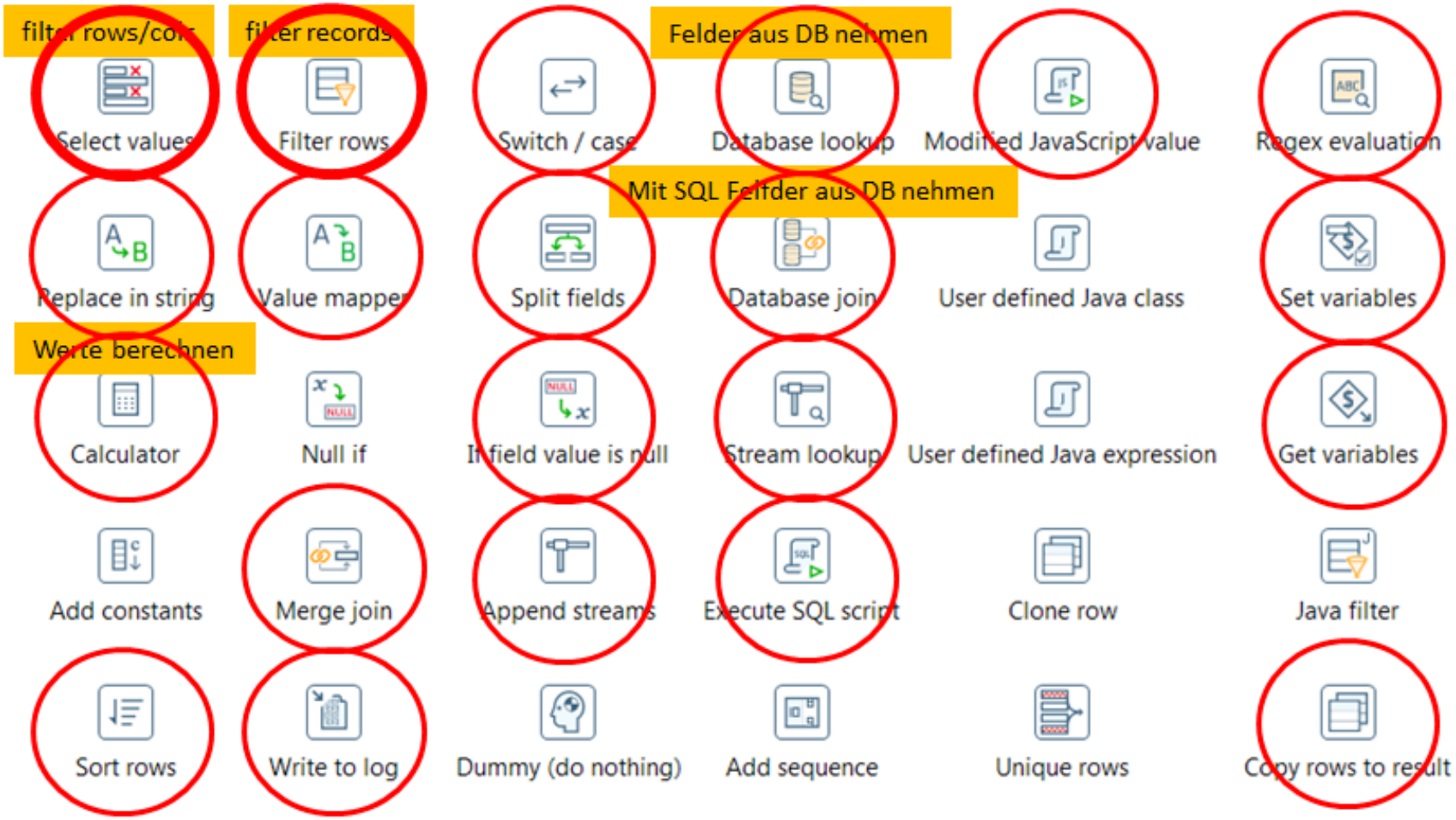
2.2) **Handling OUTPUT in Data Pipelines:**



INTRODUCTION to ETL/ELT

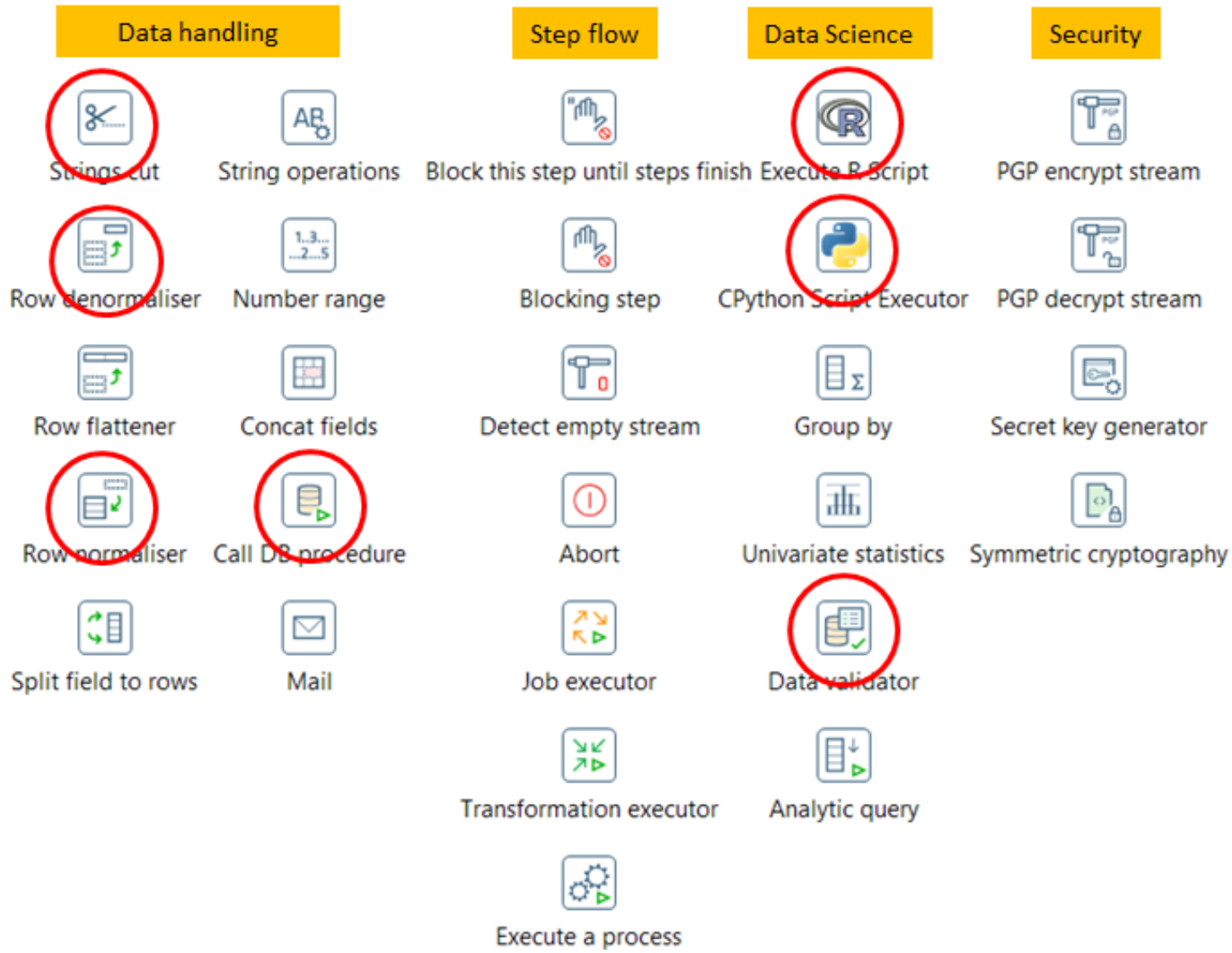
Transformations

3.1) **Basic TRANSFORMATION functions in Data Pipelines:**
Lookups (stream, DB), Joins, Sort, String handling, Filtering (fields/records), types, split fields, denormalization, normalization and much more



Transformations

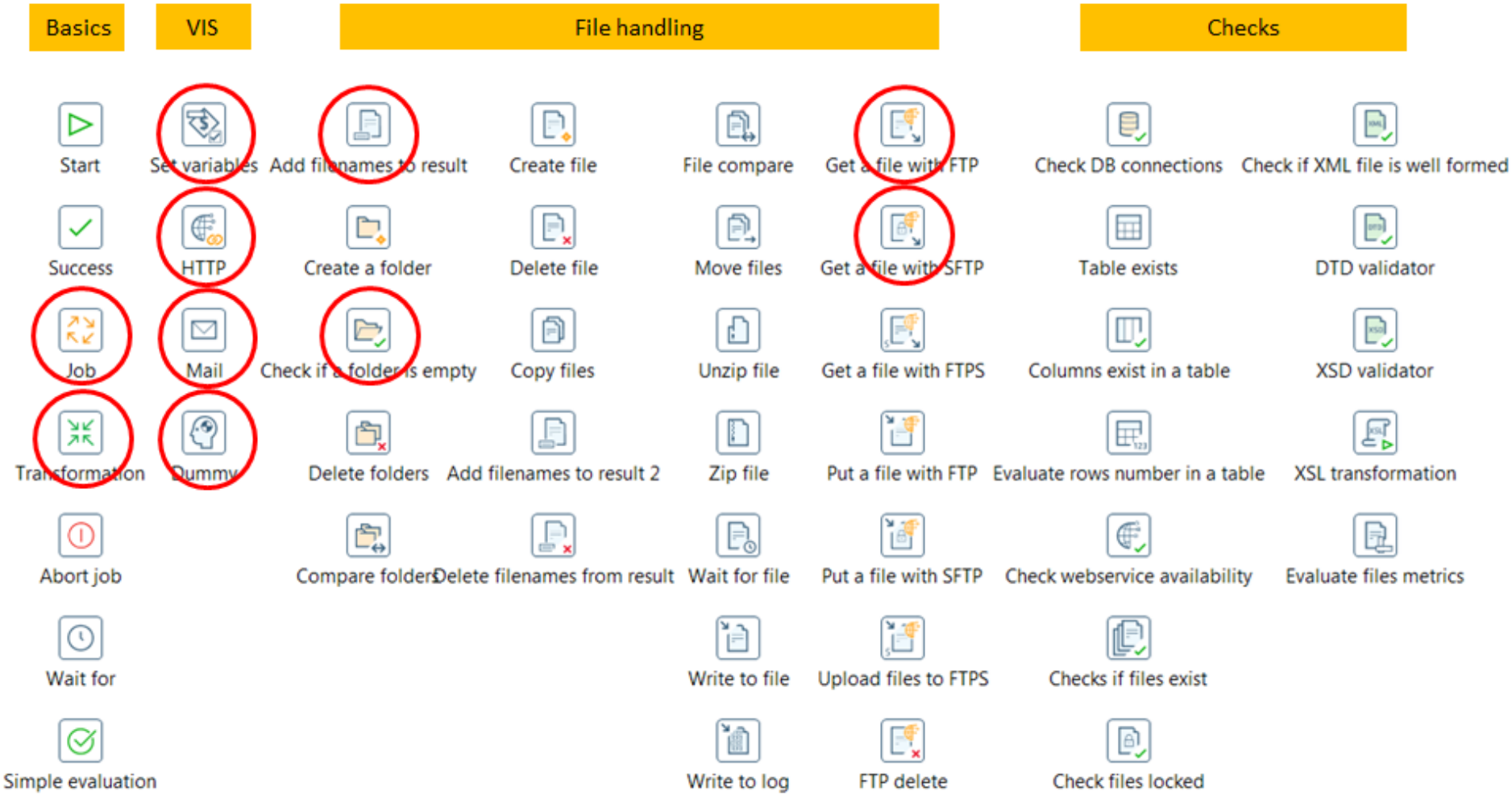
3.2) **More TRANSFORMATION functions in Data Pipelines:**



INTRODUCTION to ETL/ELT

Jobs

3.3) **Basic JOB functions in Data Pipelines:**



Jobs

3.4) **More JOB functions in Data Pipelines:**

Big Data

- Hadoop copy files
- Hadoop job executor
- Spark submit
- Sqoop export
- Sqoop import
- Oozie job executor
- Amazon EMR job executor
- Amazon Hive job executor

Scripts

- JavaScript
- Shell
- SQL

Security

- Decrypt files with PGP
- Encrypt files with PGP
- Verify file signature with PGP

Utilities

- Ping a host
- Telnet a host
- Get mails (POP3/IMAP)
- Mail validator
- Truncate tables
- Wait for SQL

Bulk load

- Bulk load from MySQL into file
- Bulk load into MSSQL
- Bulk load into MySQL

More functions and features with PDI

Modified JavaScript value

Step name: shape the Period as needed format

Java script functions:

- Transform Scripts
- Transform Constants
- Transform Functions
- Input fields
  - TimeStamp
  - BusinessDate
  - SourceSystem
  - CampaignNominalIntere
  - ContractArrearsDate
  - ContractNumber
  - ContractOrigination
  - CustomerEffectiveInteres
  - EffectiveContractEndDat
  - EffectiveBuyerNumber
  - FirstRestructuringDate
  - GPA-Flag
  - NewAcquisitionFlag
  - OutstandingPrincipalAm
  - PlannedContractEndDate
  - ProposalRecordedBy
  - ReplacedContractNumbe
  - ReplacementContractNu
  - Scoring DateTime
  - Standcode
  - Vertragsart
  - FirstInstalmentNetAmour

Java script:

Script 1

```
//Script here

var TimeStamp = new java.lang.String(TimeStamp);
var Period1 = TimeStamp.substr(2,4); // strip away first 2 and last 2 chars
var nullnull = '00';
var Period = Period1.concat(nullnull); //fill the last 2 chars with 00 values
```

Linenr 0

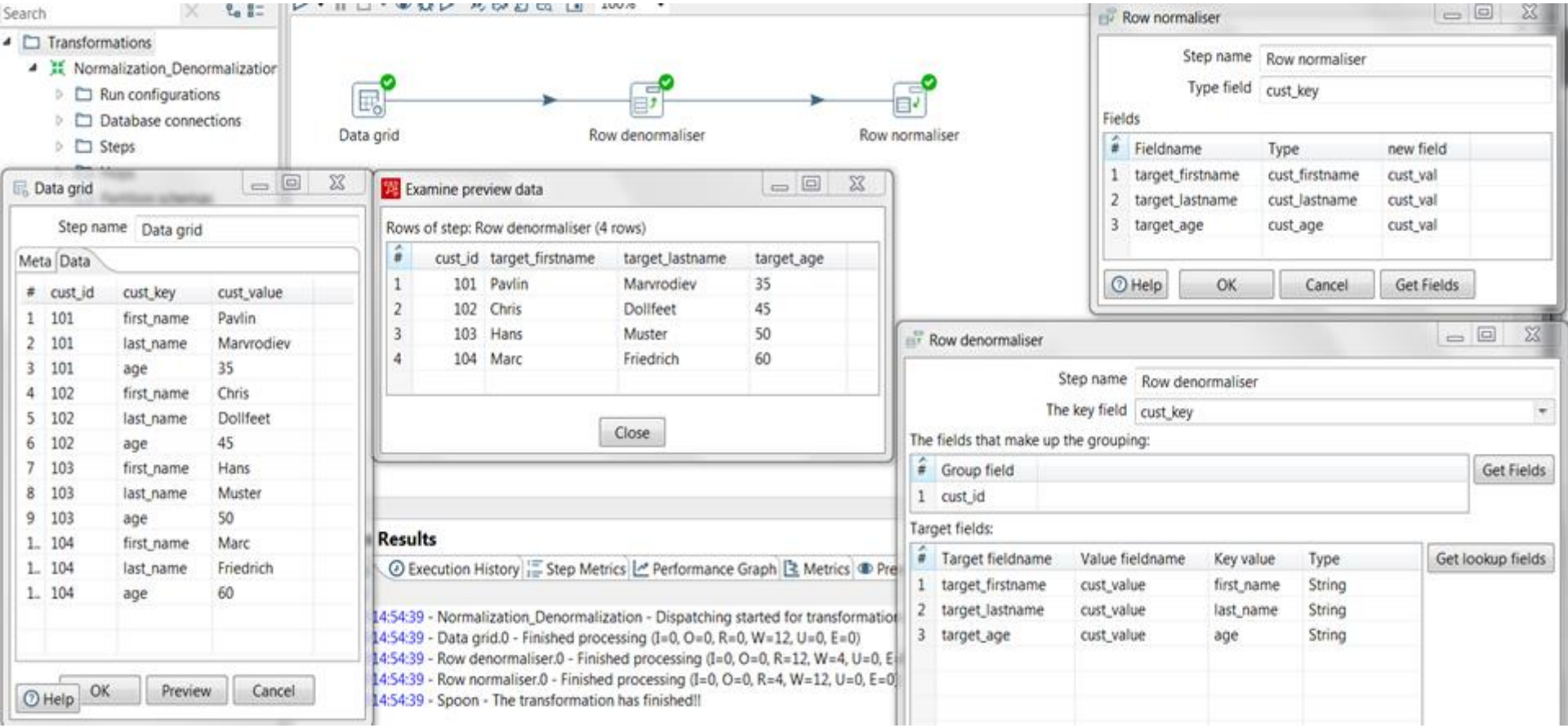
More functions and features with PDI

- Automatic SQL-statements using Table Input and Table Output (CREATE TABLE) Statement

- SQL in PDI:
  - **Table Input**
  - **SQL-Script**
  - **Database Join**

- Variables in PDI → https://help.pentaho.com/Documentation/8.2/Products/Data_Integration/Data_Integration_Perspective/Run_Modifiers/Variables

Normalization and Denormalization



Regular Expression

- Regular Expressions: (Wikipedia: **regex** or **regexp**;[1] also referred to as **rational expression**)[2][3] is a sequence of characters that define a *search pattern*)

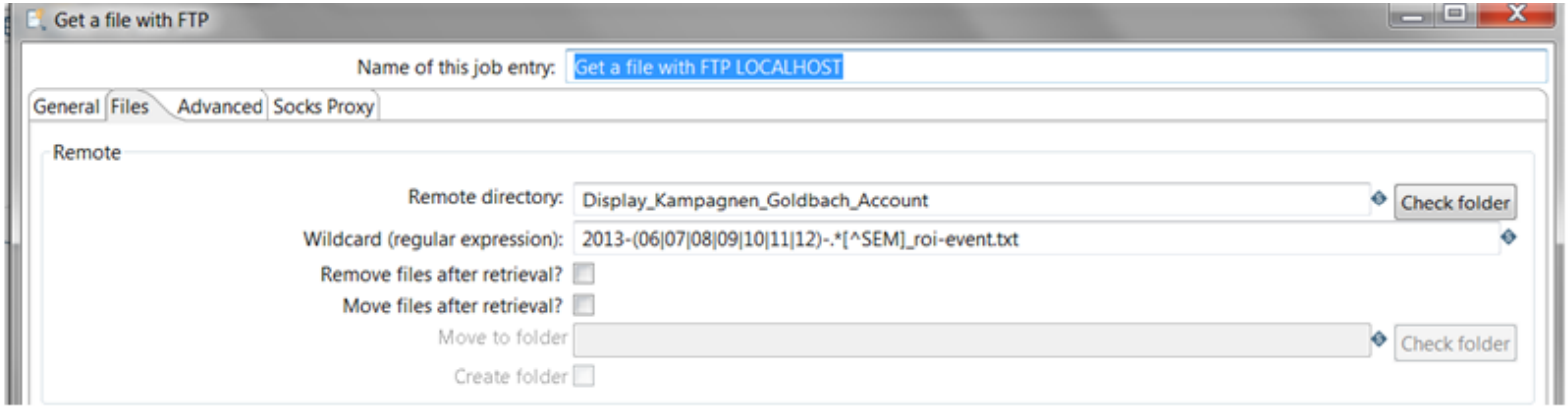
Example: 2013-(06|07|08|09|10|11|12)-.*_roi-event_TEST.txt

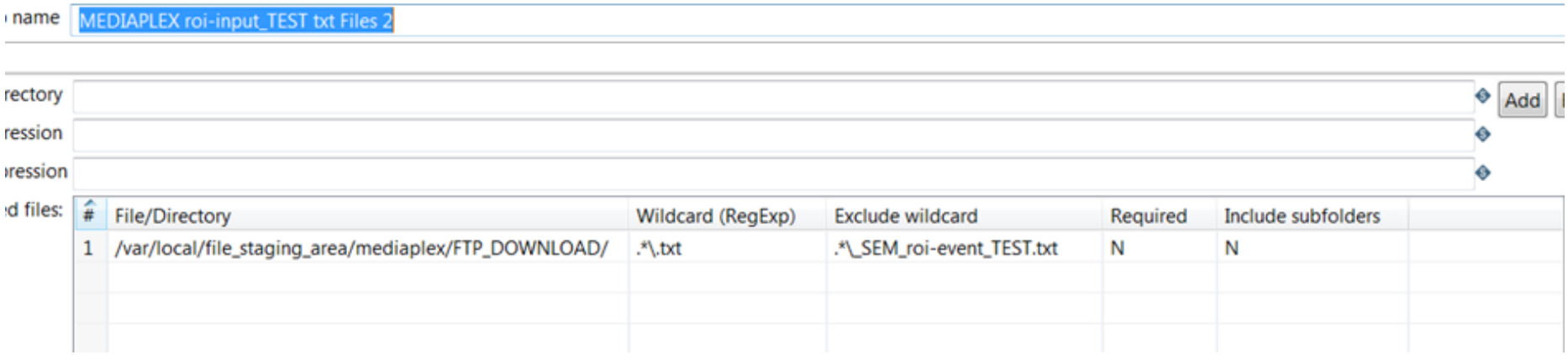
[!"#$%&'()*+,-.\\\/:;<=>?@[\\]`{}^_`{|}~] → select all special chars

In PDI Regexp can be used in different applications :

- As selection of files
- As Filter for records
- String Operations (i.e. «replace in String»)

Get File with FTP





Filling the SQL-statement automatically (CREATE TABLE)

Table output

Step name: t_stage_nodes2
Connection: localhost_regmarket (Edit... New... Wizard...)
Target schema: (Browse...)
Target table: t_stage_nodes22 (Browse...)
Commit size: 1000
Truncate table ☑
Ignore insert errors ☐
Specify database fields ☐

Main options | Database fields

Partition data over tables ☐

Simple SQL editor

SQL statements, separated by semicolon ';'

```
CREATE TABLE t_stage_nodes22
(
  source_id INT
, target_id INT
, source_node_name VARCHAR(85)
, source_node_type VARCHAR(85)
, target_node_name VARCHAR(85)
, target_node_type VARCHAR(85)
)
;
```

Line 1 column 0

Execute | Clear cache | Close

Help | OK | Cancel | SQL