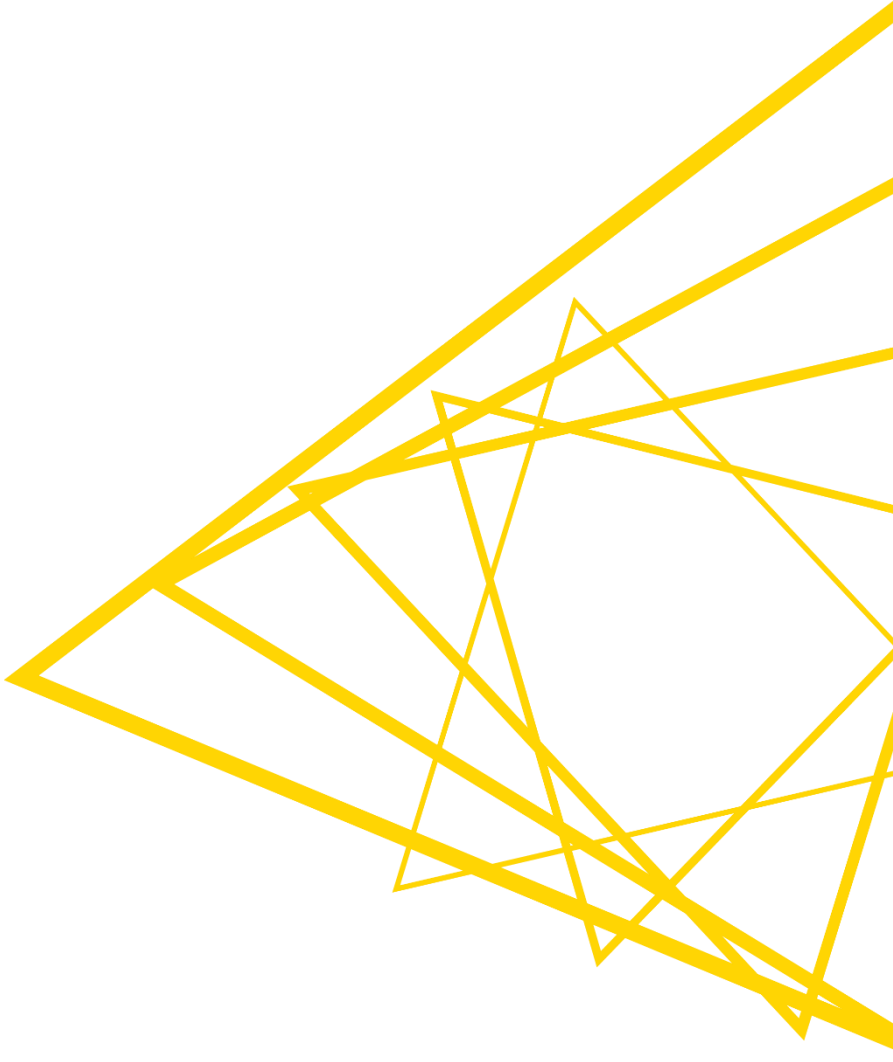
Open for Innovation
KNIME

[L4-DE] Best Practices for Data Engineering

education@knime.com

Introduction to the Course & Technical Setup



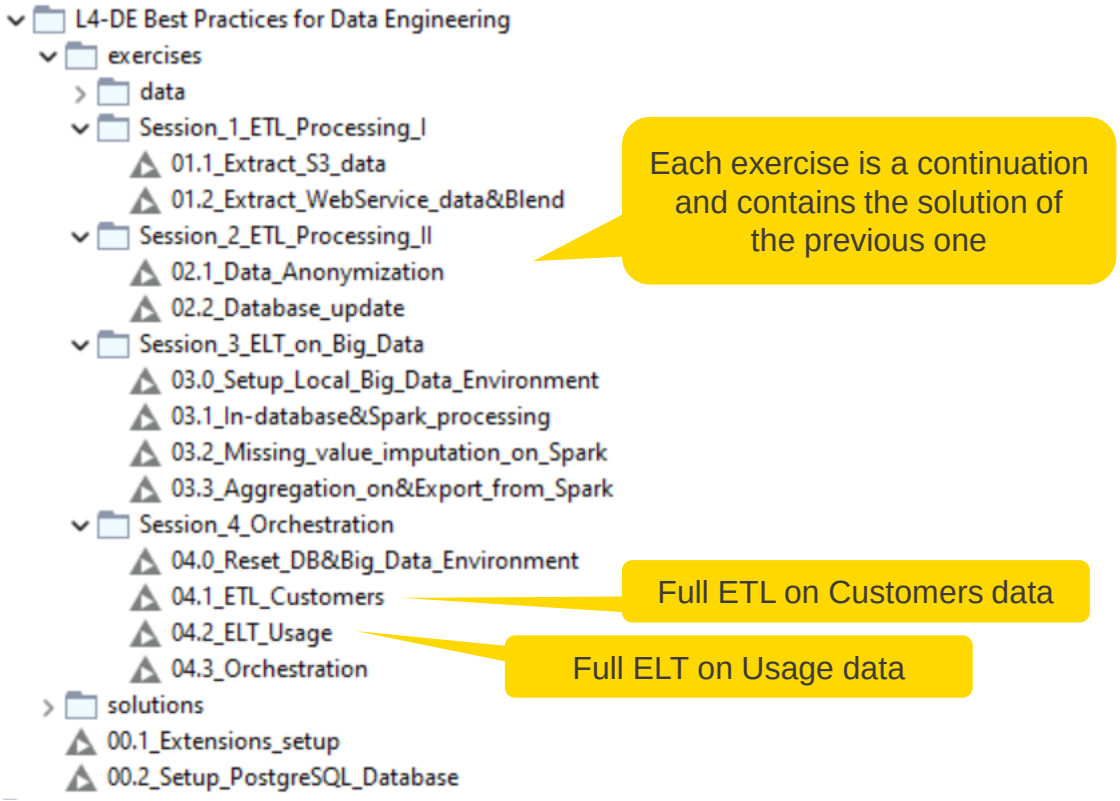
Goal of this Course

- In the end of this course, you'll be able to…
  - Access various data types and connect to various data sources
    - structured and semi-structured data, databases, data lakes, data from the web, cloud, etc.
  - Build ETL and ELT data pipelines
  - Work with big data in KNIME Analytics Platform
    - integrate Hadoop Ecosystem (both on premise and on the cloud) into KNIME Analytics Platform
    - process data and train and apply machine learning models on Spark
  - Orchestrate multiple workflows and build workflow dependencies
- Learn and apply **best practices** for data engineers

Agenda

- Session 1: Introduction & technical setup, ETL, Connectors & Data access
- Session 2: ETL, Data anonymization, Databases
- Session 3: ELT, Big Data, Hadoop, Spark
- Session 4: Cloud and Big Data connectivity, Orchestration

Exercises Overview

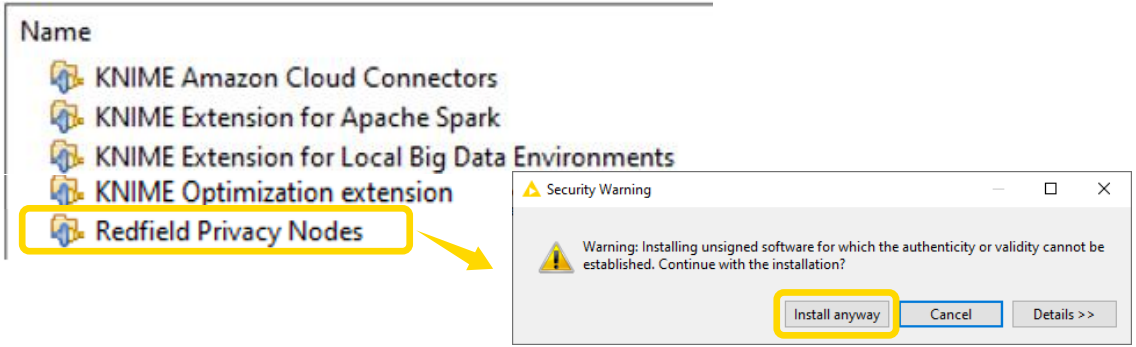


You can download the training workflows from the [KNIME Community Hub]

Technical Setup

What you need:

- PostgreSQL database installed locally
- KNIME Analytics Platform Extensions
  - Install via opening the 00_Extensions_setup workflow
  - Or via File > Install KNIME Extensions > …
  - Or via Drag and Drop from KNIME Community Hub



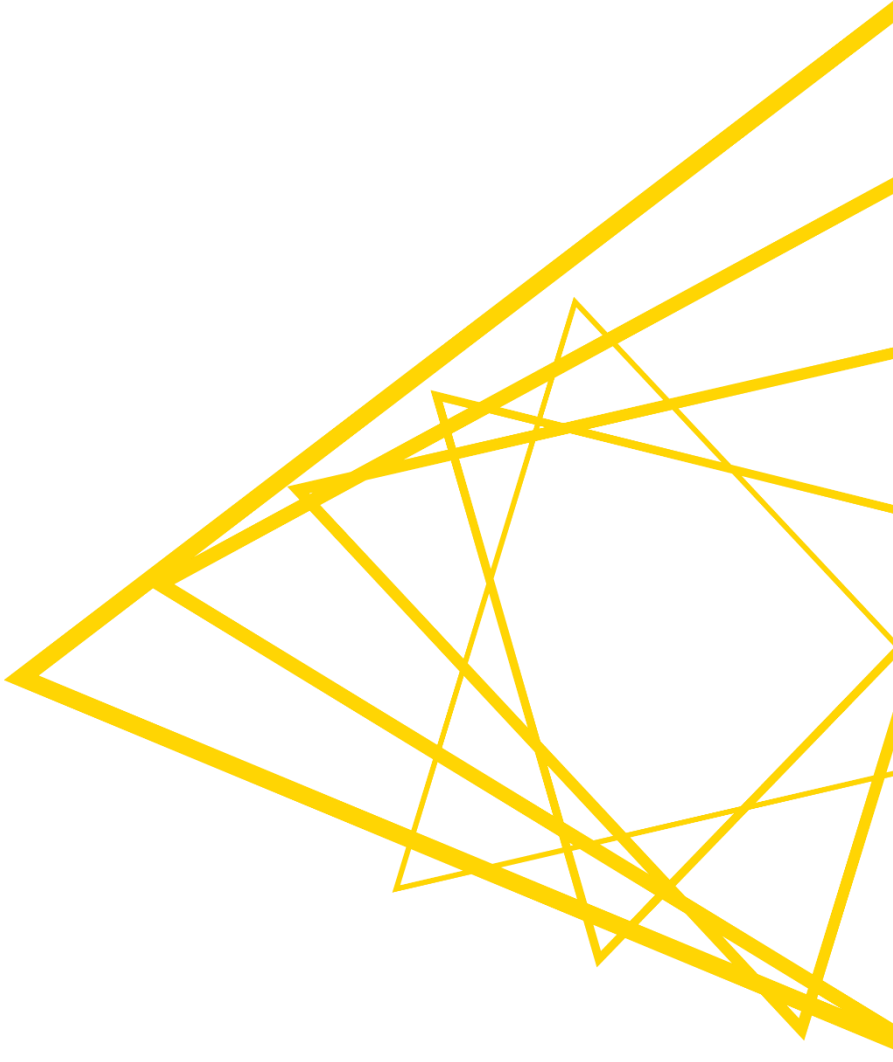
Session 1: ETL, Connectors & Data access

Session 1: Learning Outcomes

At the end of this session, you will be able to:

- Recognize various KNIME connector nodes
- Access, validate, and parse data from a web service and a data lake
- Apply best practices regarding security, efficiency, reusability, and data validation
- Build the first part of the application that accesses, validates, processes, and blends the new customer data from two different sources

Background



Data Engineering

“Data engineering is the development, implementation, and maintenance of systems and processes that take in raw data and produce high-quality, consistent information that supports downstream use cases, such as analysis and machine learning.**”**

Reis, J., Housley, M. (2022). Fundamentals of Data Engineering: Plan and Build Robust Data Systems. O'Reilly Media

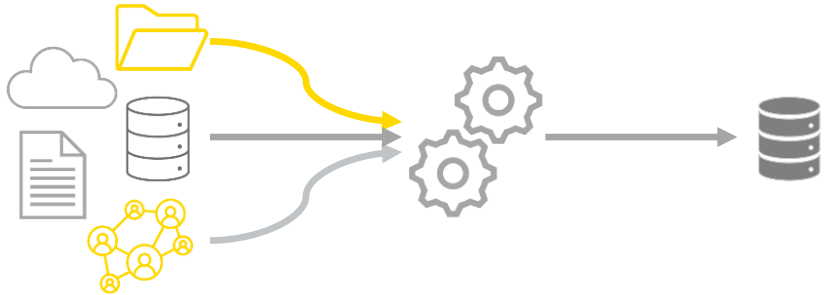
Who is this Course for: Data Engineers

Data Analyst	Data Scientist	Data Engineer
Data cleaning, data analysis, descriptive statistics, reporting, data visualizations, dashboards	Data pre-processing, training machine learning and statistics algorithms, modeling, predicting	Integrating various data sources, building and managing data pipelines (ETL, ELT), databases, data lakes, data warehouses, file systems, and/or data mart maintenance, monitoring and testing

ETL and ELT

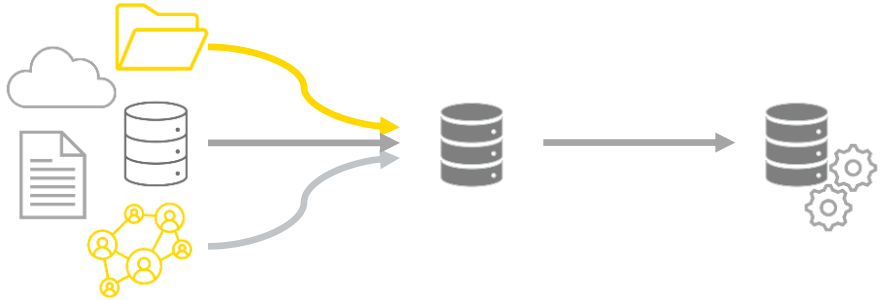
- ETL

Extract data from data sources → **Transform** data in an intermediate storage → **Load** into a data warehouse



- ELT

Extract data from data sources → **Load** into a data lake → **Transform** within the data lake



Why to Apply Best Practices?

Your data ETL/ELT pipeline should work

- repeatedly, automatically, error-free, and smoothly.

Your workflow works well for this data in this context

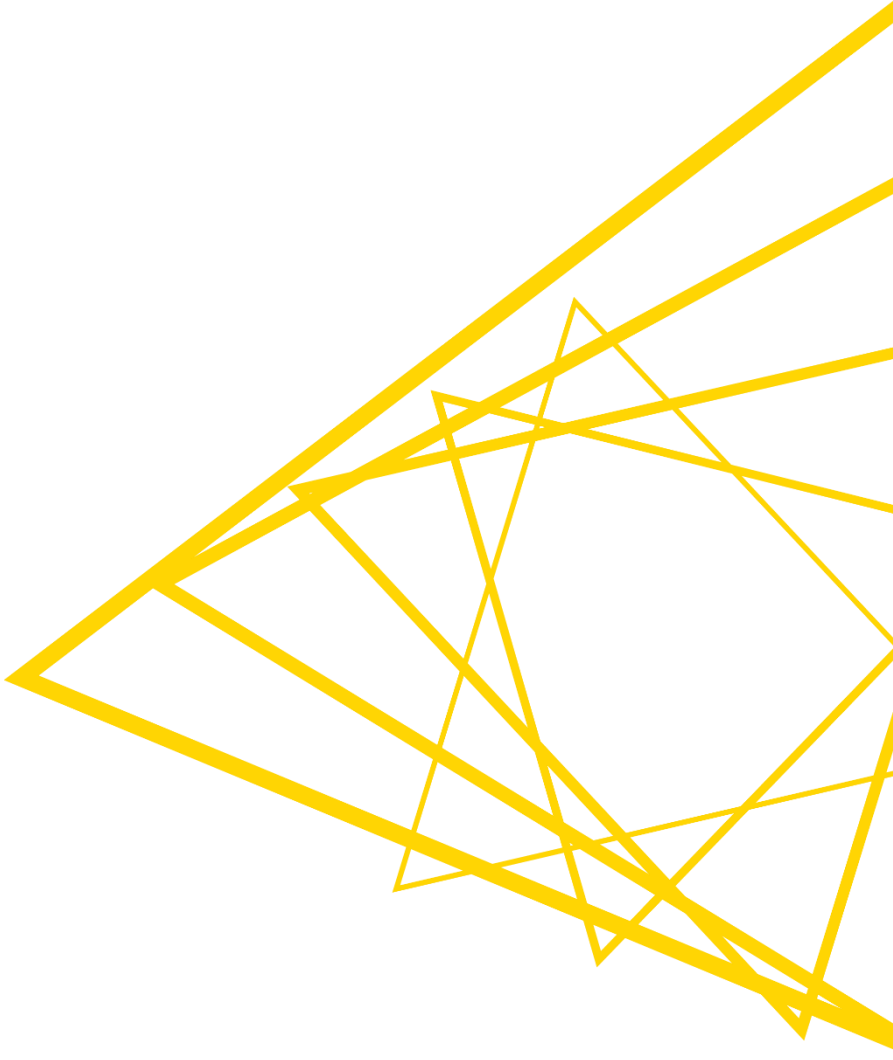
→ But is it ready to be put into use?

- Is its design efficient?
- Will it work with new data?
- Will it scale?
- What will happen in the case of failure?
- Will you be able to trace an error easily?
- Finally, is it secure?

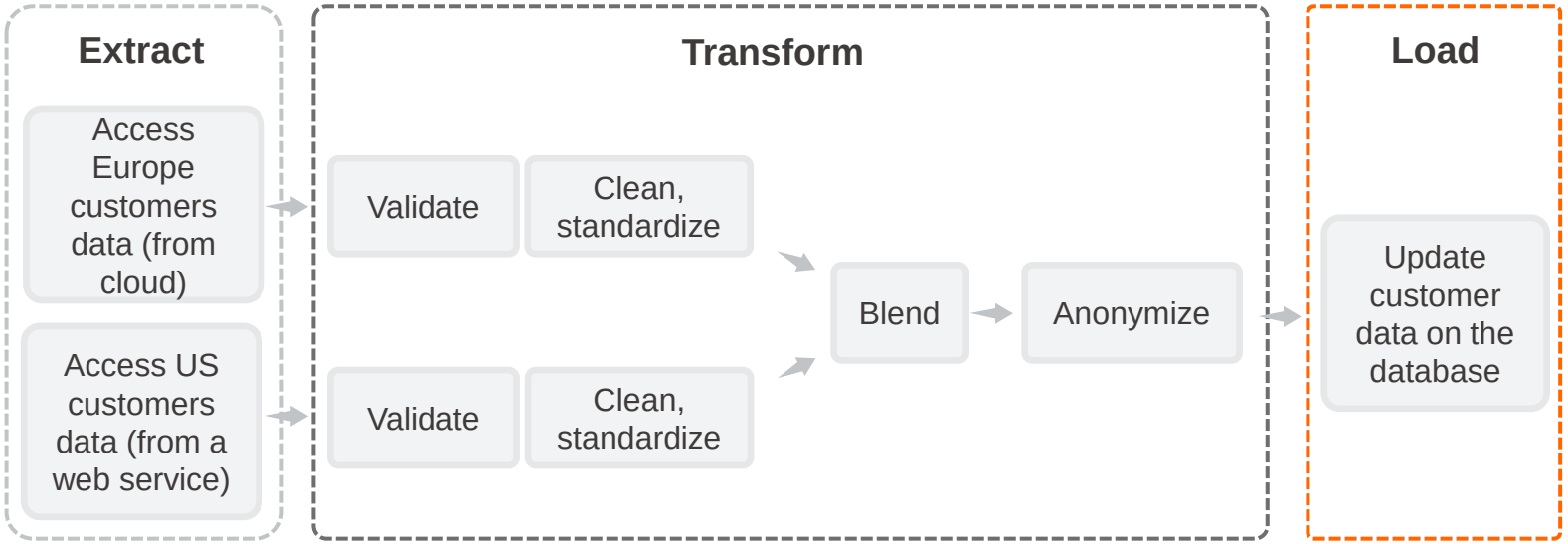
Best Practices for Data Engineers

- Efficiency 
- Scalability 
- Reusability 
- Error handling
- Security 
- Repeatability 

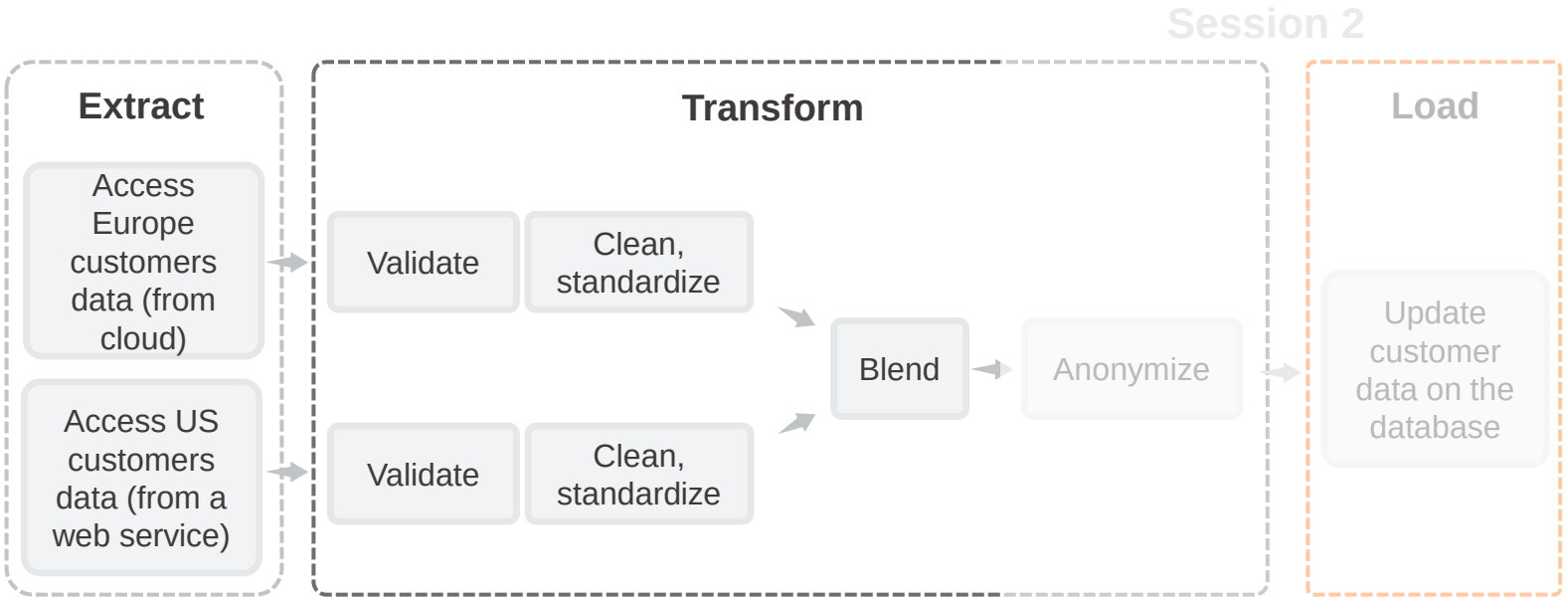
Today‘s example: ETL on Customers data – Part I



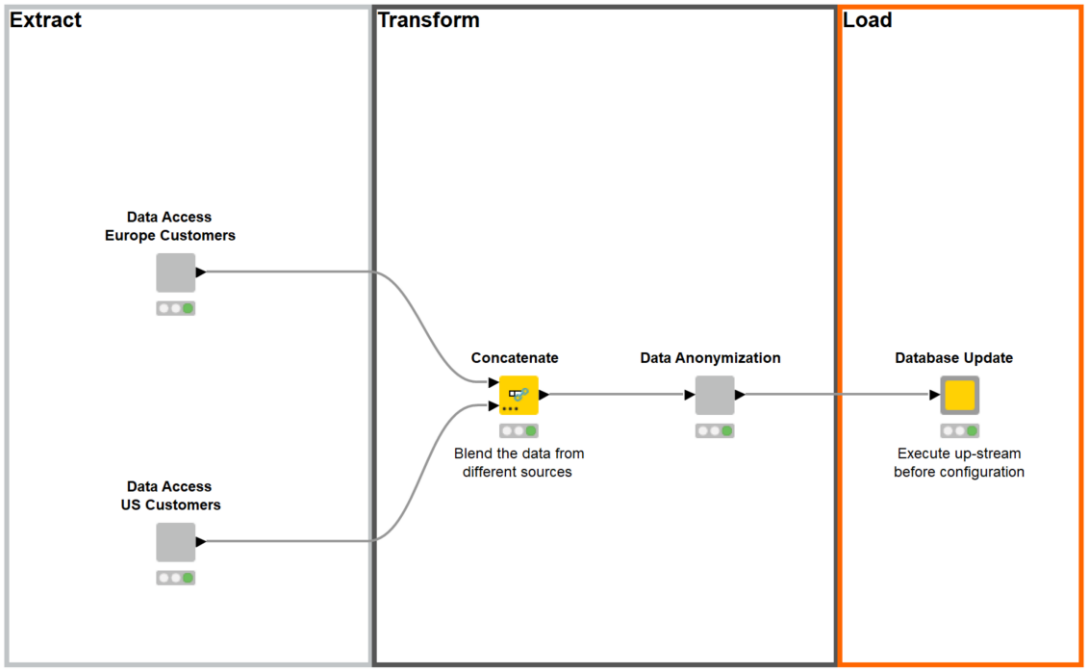
Today’s Example: ETL on Customers Data



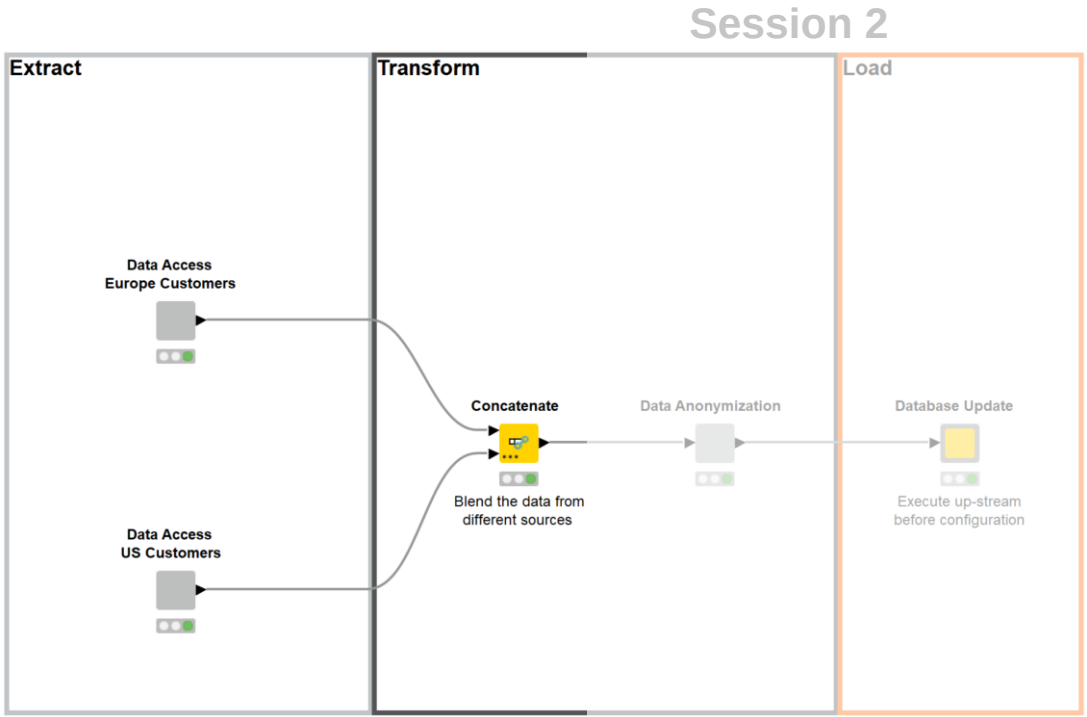
Today’s Example: ETL on Customers Data



Today’s Example: ETL on Customers Data



Today’s Example: ETL on Customers Data



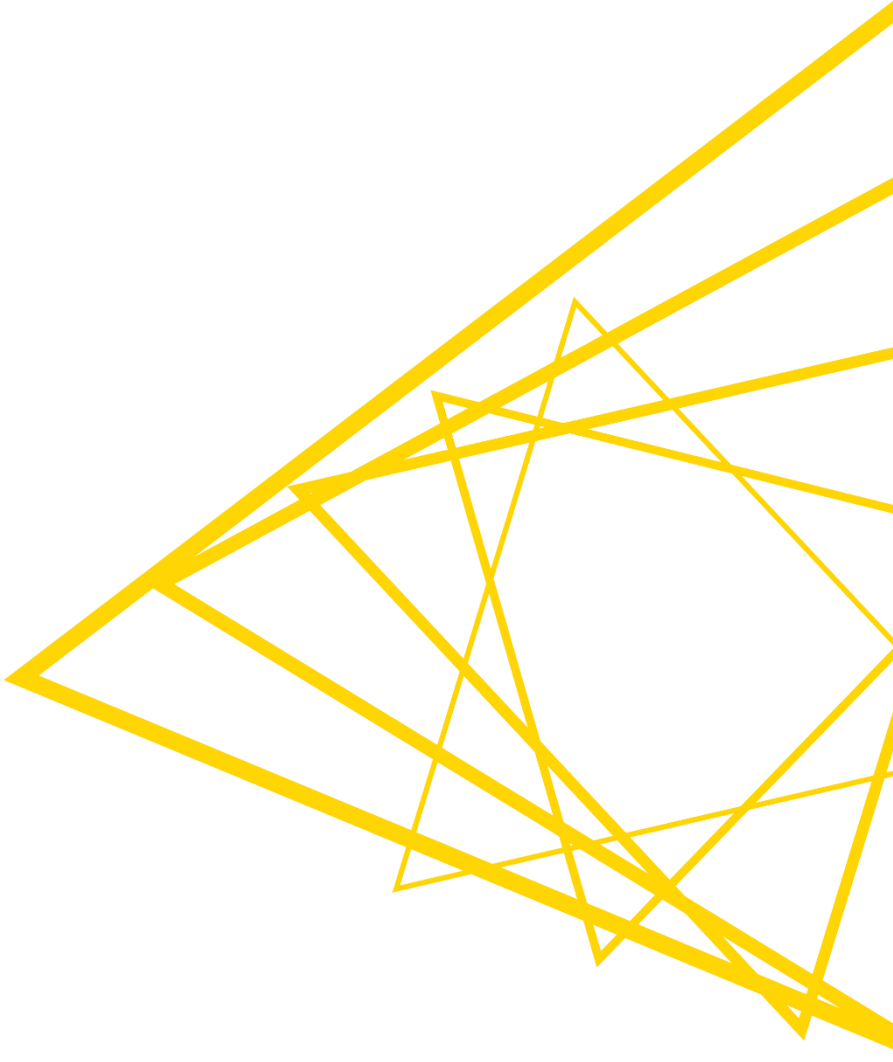
Data Set

Customers

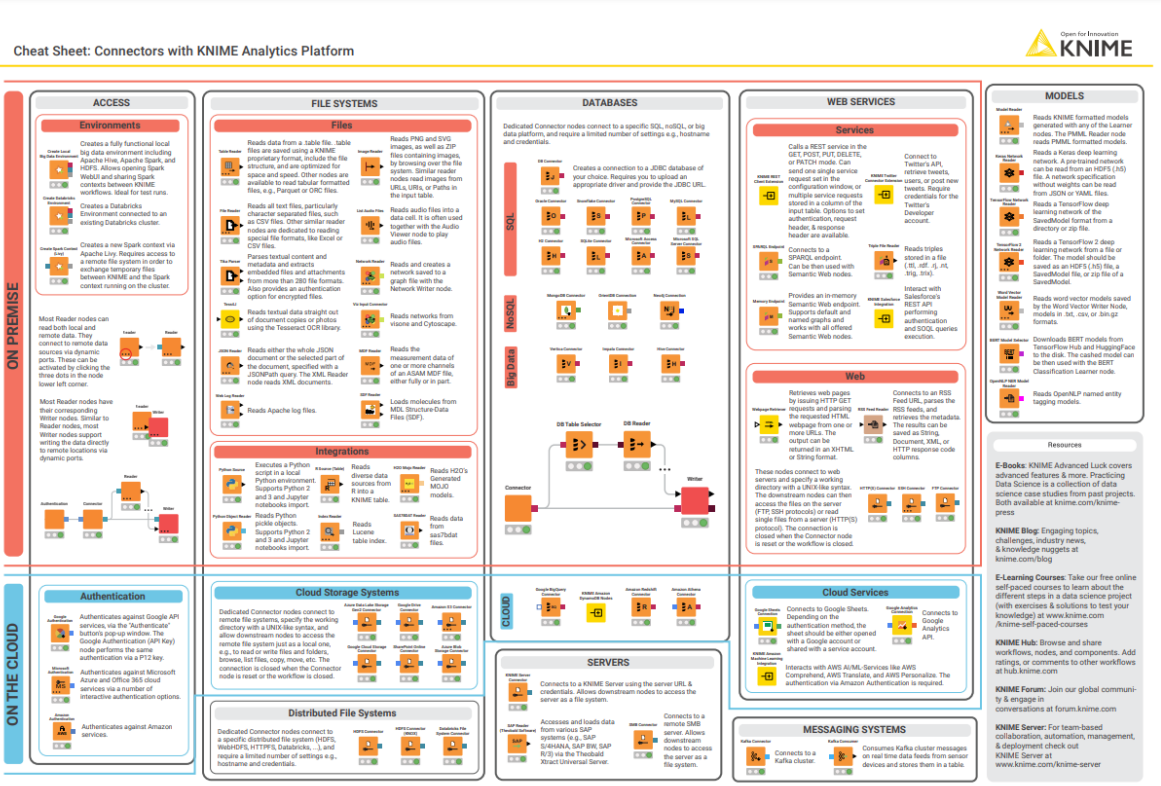
Custom...	Name	Age	Birthday	Countr...	City	Country	Email	Newsle...	Gender	Marital...	Estimat...
11026	Giuseppina Nitsch	64	1975-12-09	United States	Denver	United States	giuseppina.nitsch@provider.com	1	M	S	30,000
11033	Natalija WeiÃŸe	63	2001-12-09	United States	Los Angeles	United States	natalija.weiãÿe@provider.com	0	M	M	20,000
11044	Victor Koll	61	1975-02-12	United States	Fort Worth	United States	victor.koll@provider.com	0	M	M	20,000
11057	Stavros Selle	57	1997-11-14	United States	Chicago	United States	stavros.selle@provider.com	1	M	M	70,000
11066	Salih Limbach	36	1979-11-21	United States	Boston	United States	salih.limbach@provider.com	0	F	M	70,000

The dataset is generated randomly. Any reference to living persons or real events is purely coincidental

Connectors

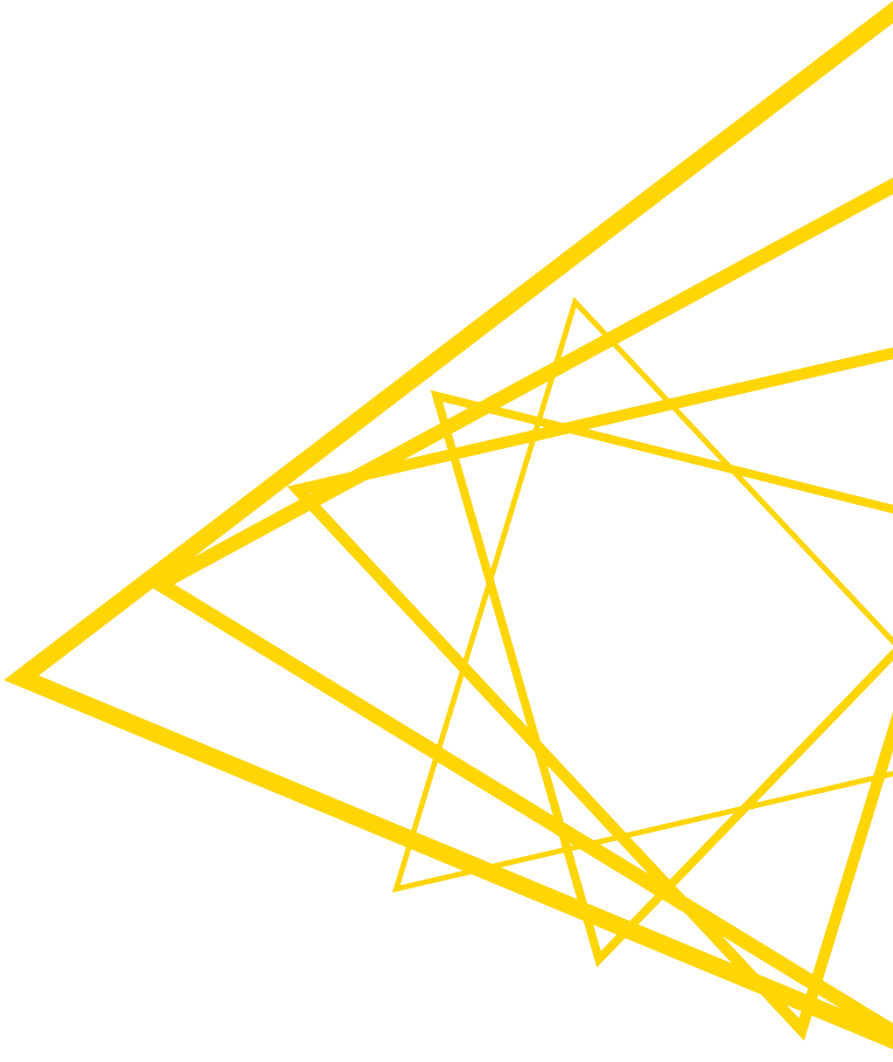


Connectors Cheat Sheet

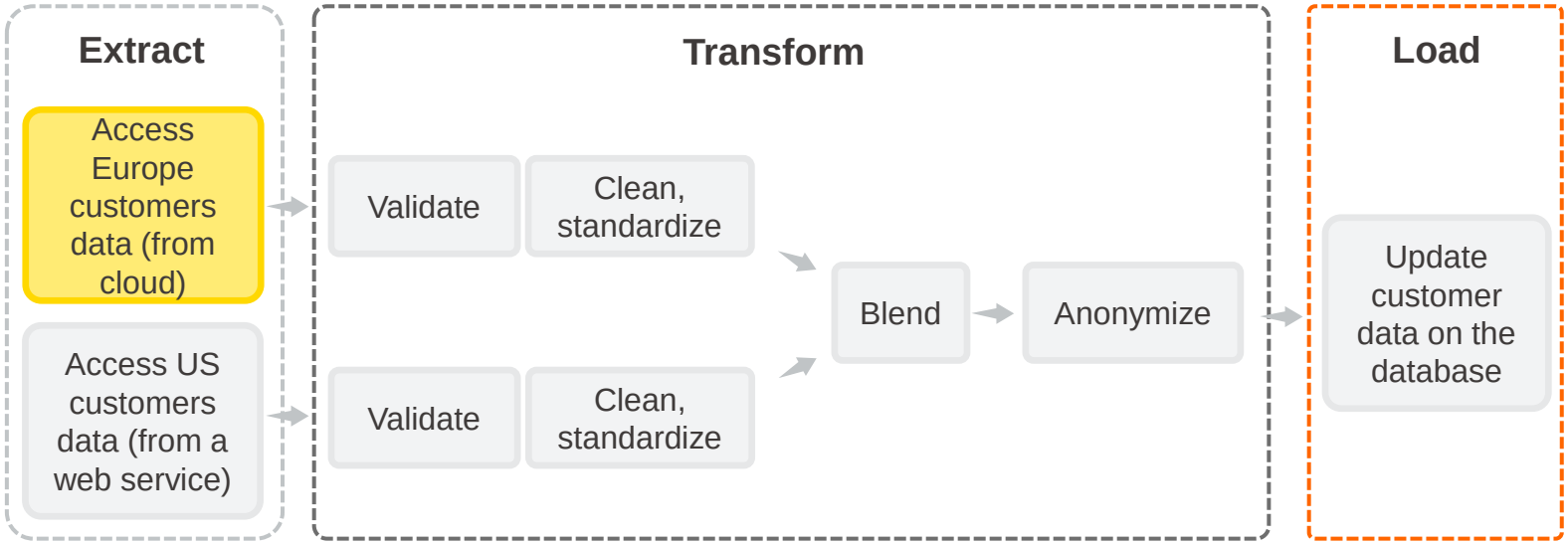


https://www.knime.com/sites/default/files/2021-07/cheat-sheet-connectors.pdf

File Handling

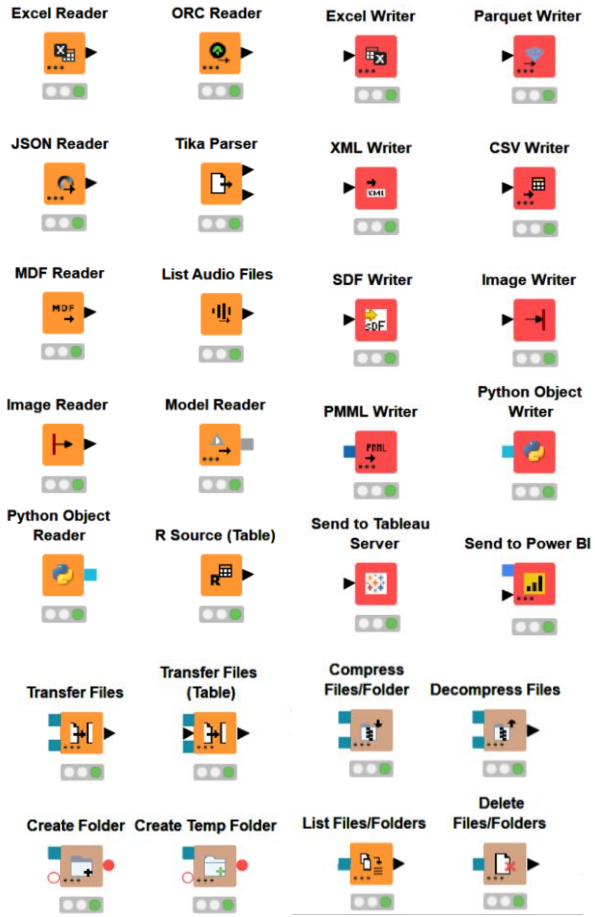


File Handling



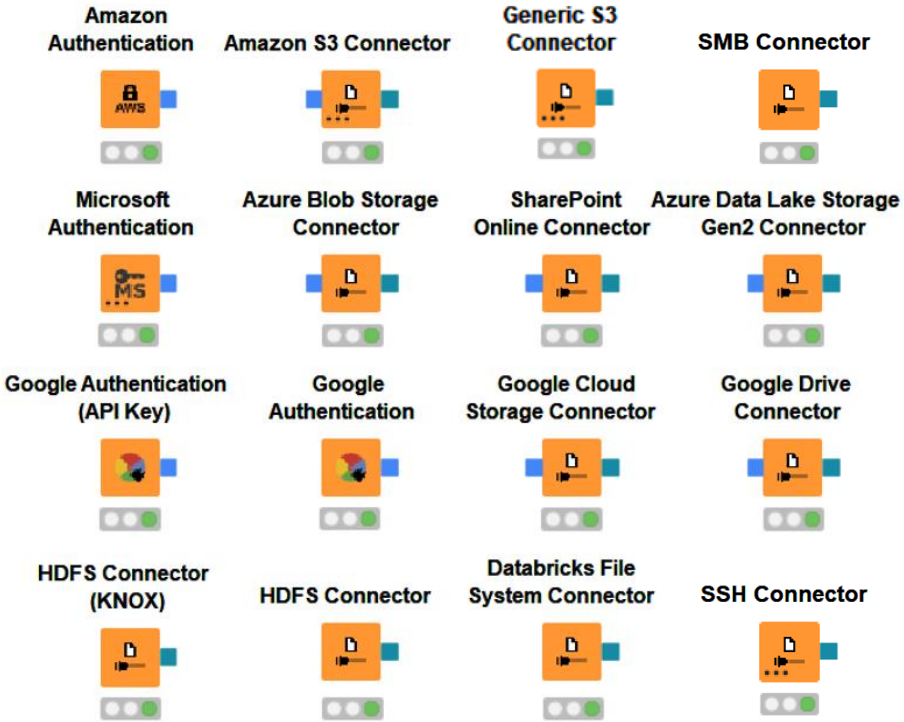
Data Access, Export, & Utility Nodes

- Reading / writing tabular, structured, textual, chemical data, audio, image, and model files
- Support of integrations, e.g., Python, R, H2O, etc.
- Reading one or multiple files
- Transformation before reading into KNIME
- Reading from / writing to local and remote file systems
- Sending data directly to PowerBI, Tableau
- Manage files and folders within one or several local or remote file systems
  - Transfer (copy or move) files and folders between file systems
  - List files and folders
  - Create folders
  - Compress and decompress
  - Delete files and folders

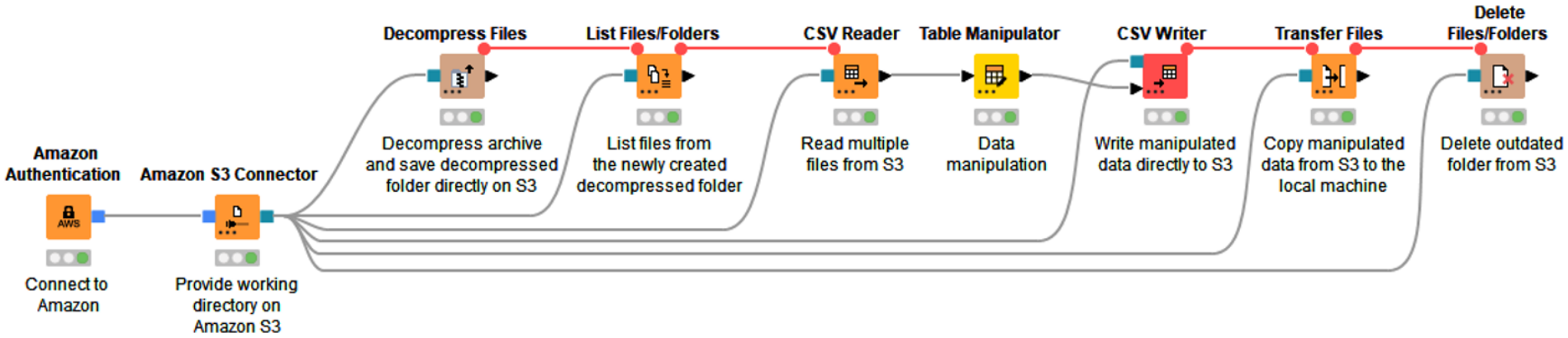


Supported File Systems

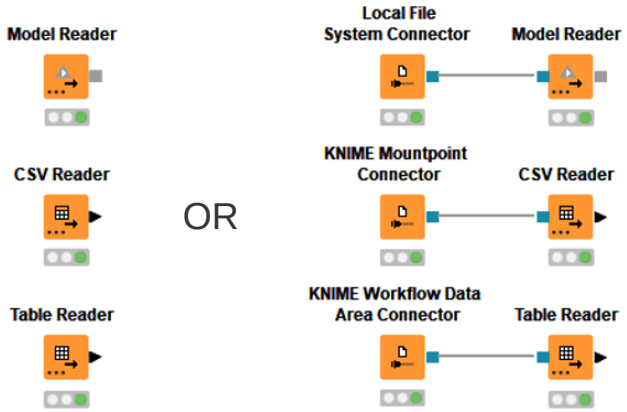
- Standard file systems
  - Local file system
  - Mountpoint
  - Relative to
    - current workflow, mountpoint, workflow data area
  - Custom/KNIME URL
- Connected file systems
  - File systems with external authentication
    - Amazon
    - Microsoft
    - Google
  - File systems without external authentication
    - Databricks
    - BigData file systems (HDFS, httpFS, …)
    - SMB, SSH, HTTP(S), FTP



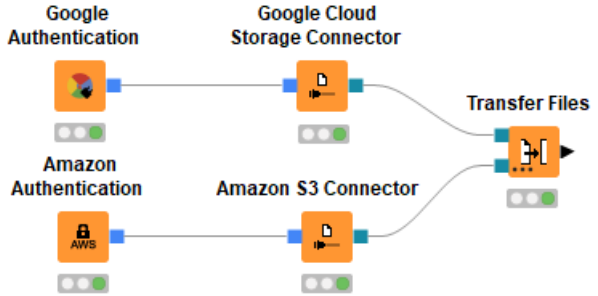
Example: File Handling on Amazon S3



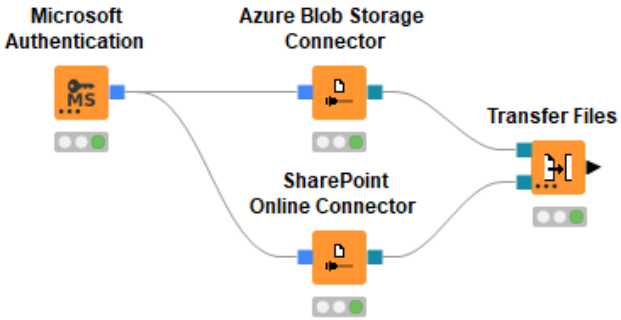
File Handling Framework – Flexibility



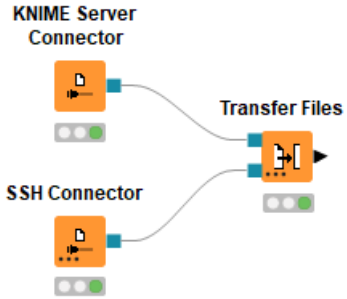
Local file system and KNIME mountpoints



Cross cloud environments

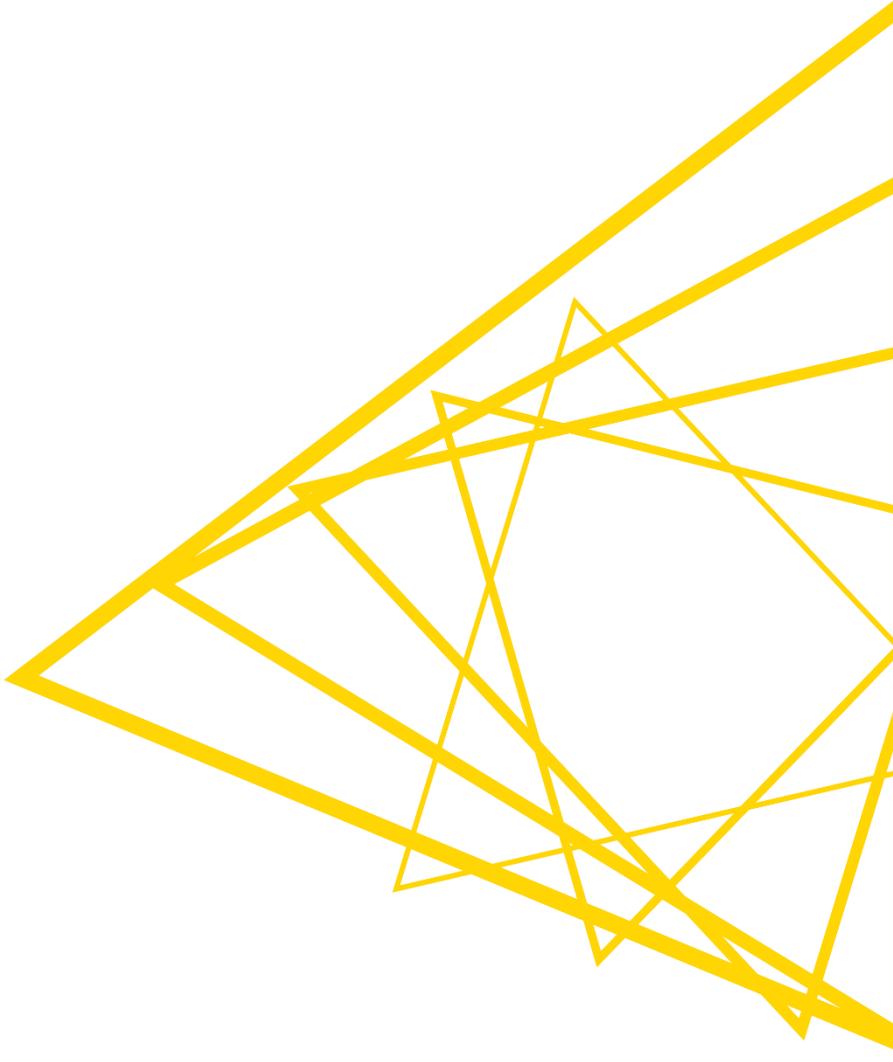


Same cloud environment



On-premise

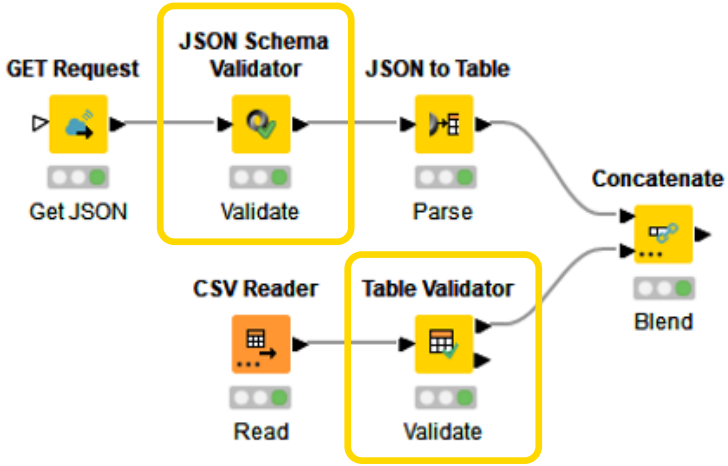
Demo



Best Practices: Data Validation

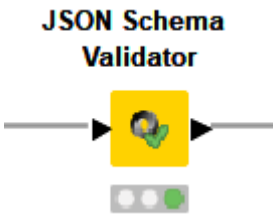
Input Data Validation

- Make sure the workflows are using correct data
  - Validate input data tables, JSON schema, input files, images, etc.



JSON Schema Validator Node

- Validates JSON values based on the specified schema
- JSON Schema
  - “A vocabulary that allows to annotate and validate the format and structure of a JSON Object”
  - Describes how data should be organized, e.g., expected fields and data types



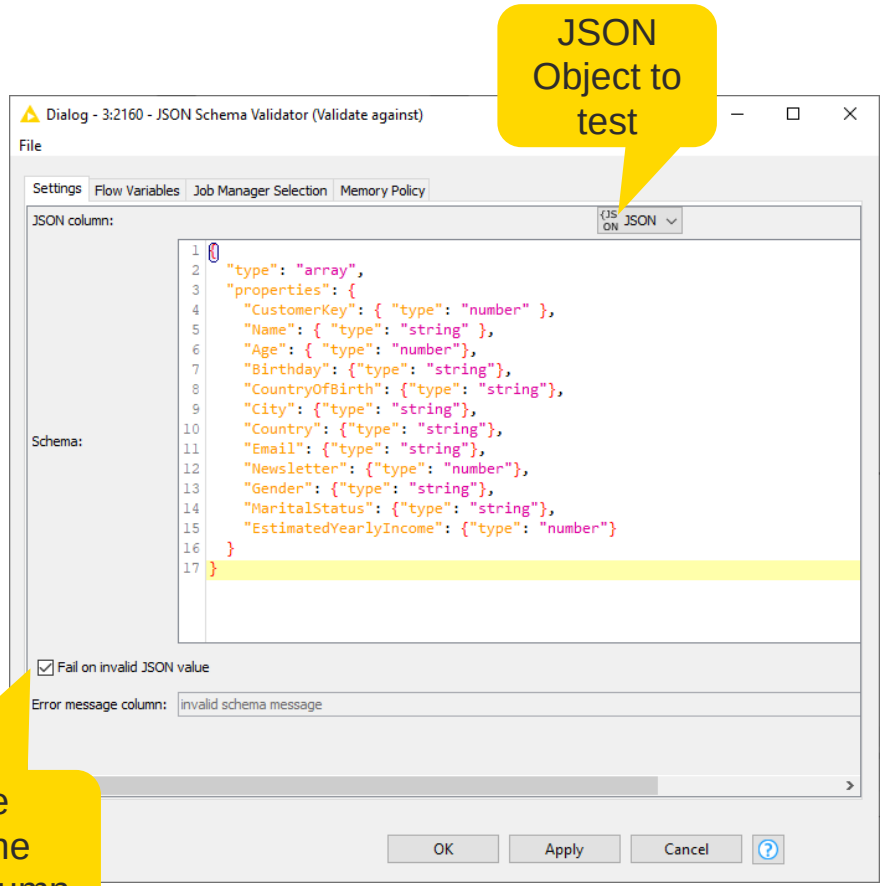
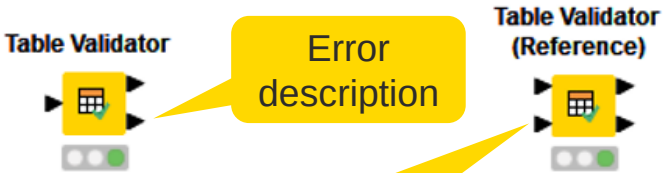
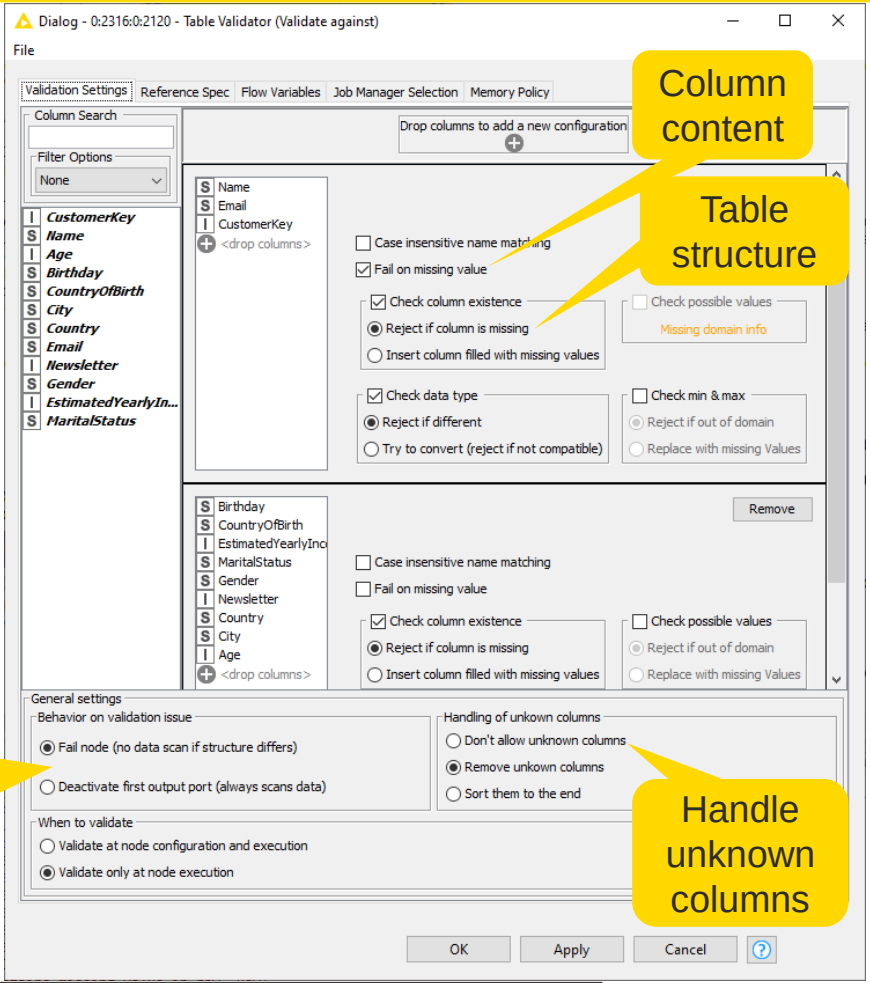


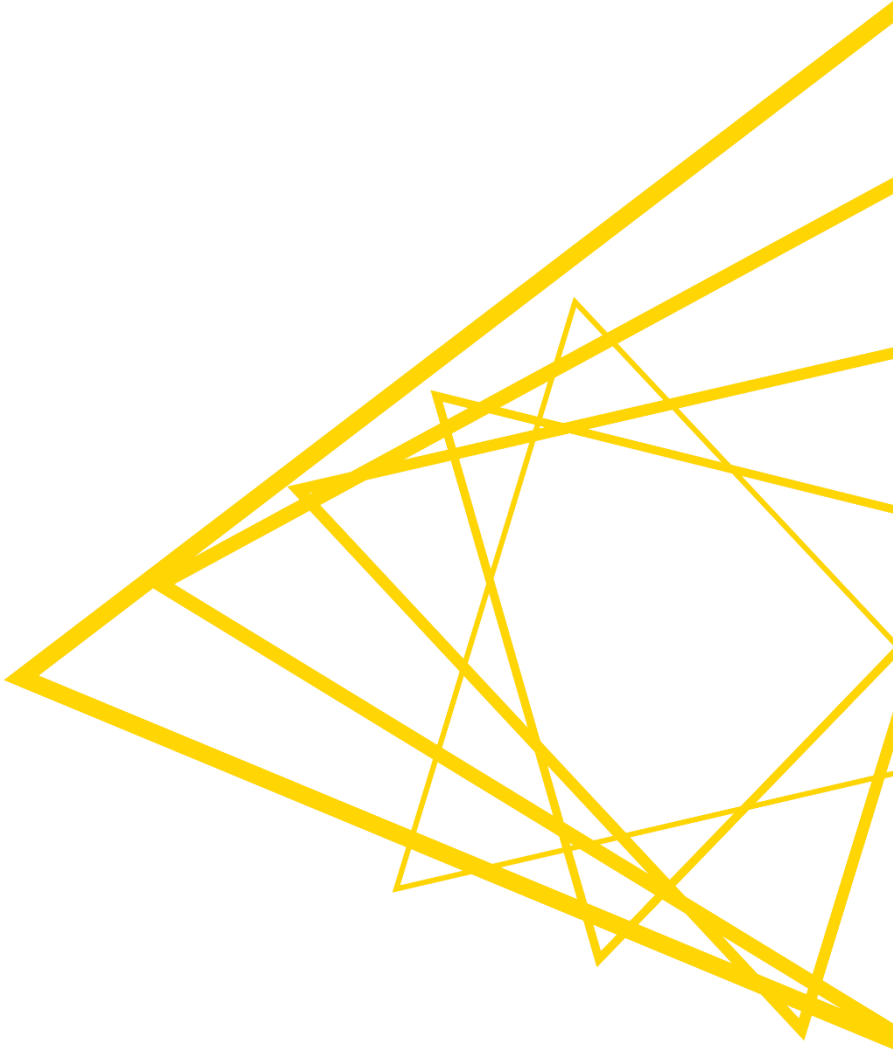
Table Validator Node

- Ensure a certain table structure and content using a reference table specification during configuration



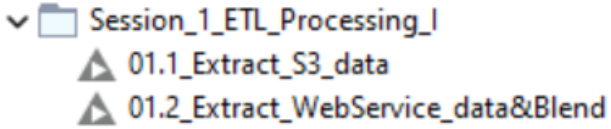


Demo



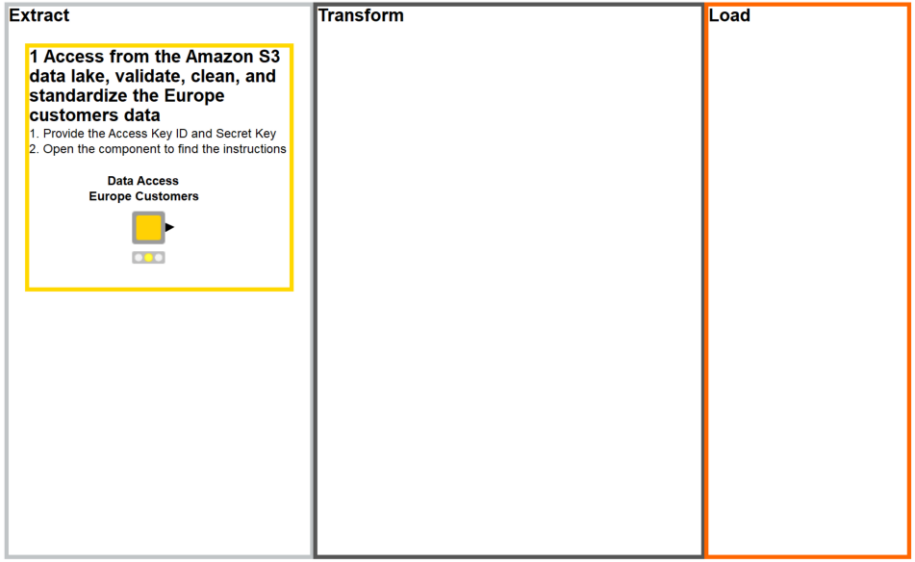
Exercises – Session 1

- Before starting the exercise
  - Install local instance of PostgreSQL
  - Download the training workflows from the KNIME Community Hub
  - Install necessary extensions (open 00.1_Extensions_setup)
  - Execute workflow 00.2_Setup_PostgreSQL_Database
    - Use the credentials for your local instance of PostgreSQL

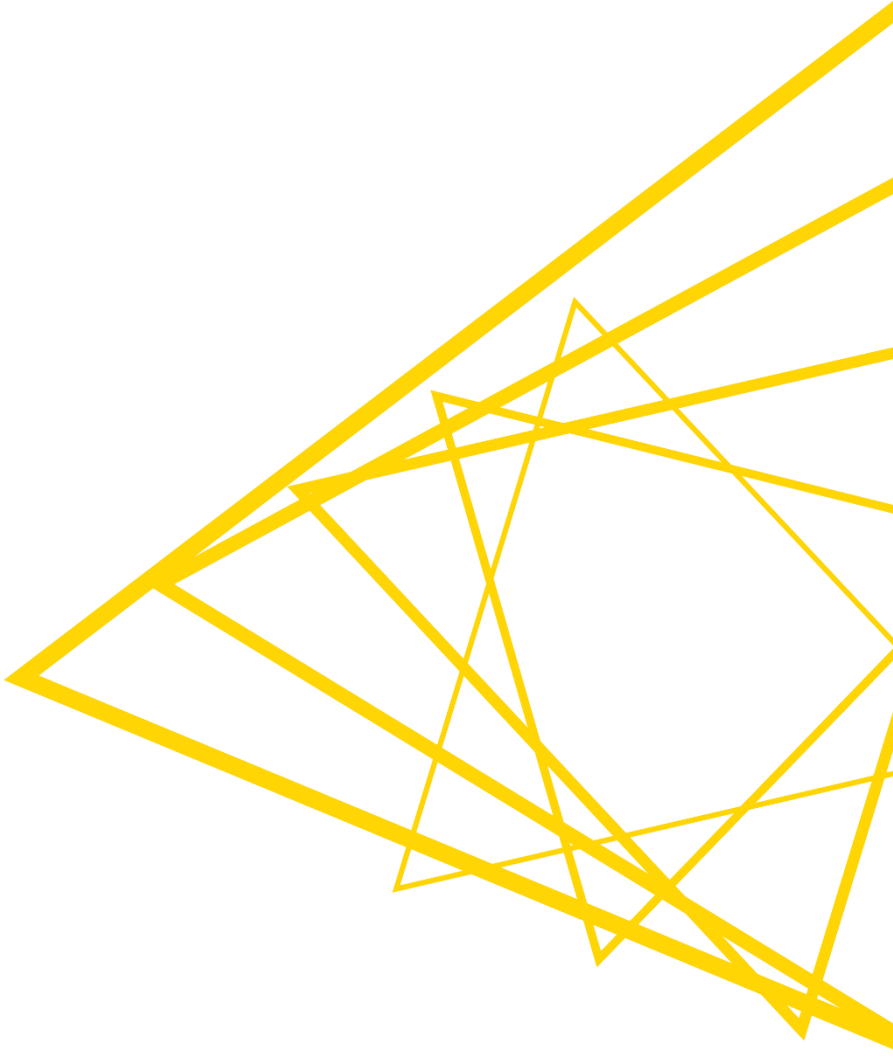


Exercise – 01.1_Extract_S3_data

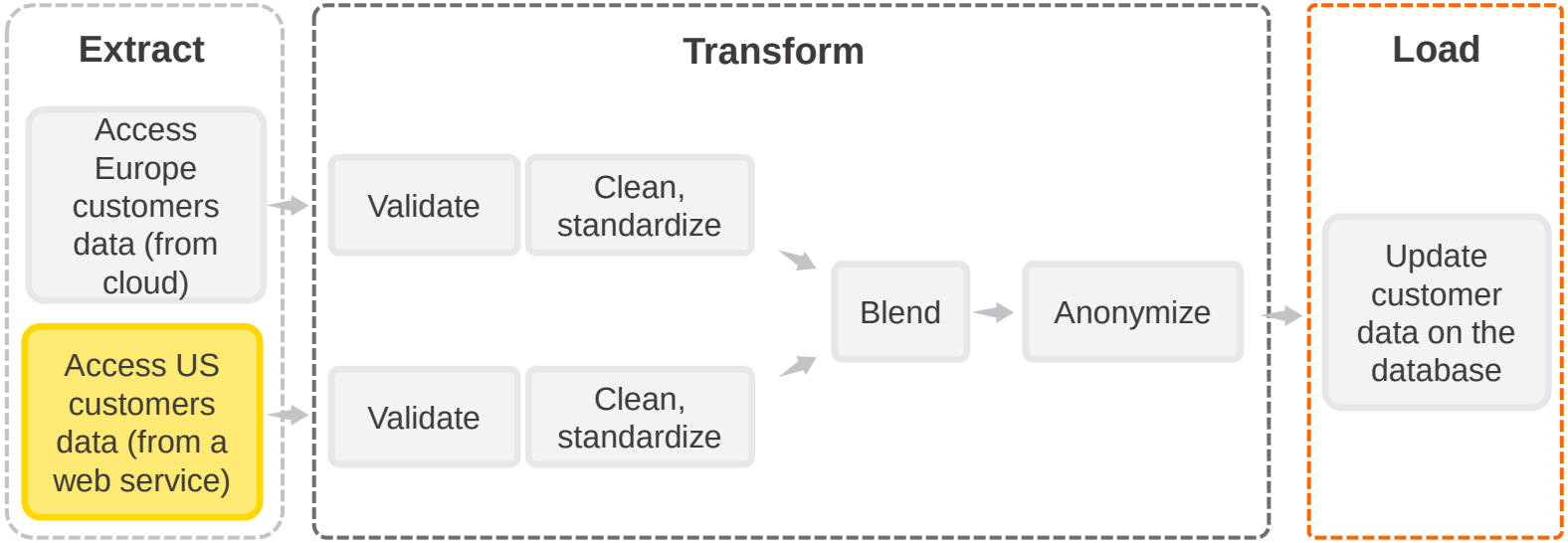
- This exercise is the first step to build application “ETL on Customers Data”
  - 1 Access from the Amazon S3 data lake, validate, clean, and standardize the Europe customers data
    - Credentials for the Amazon Authentication node can be found in the reminder email
  - Find detailed instructions in the workflow



Web Services

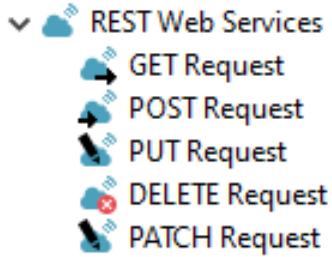


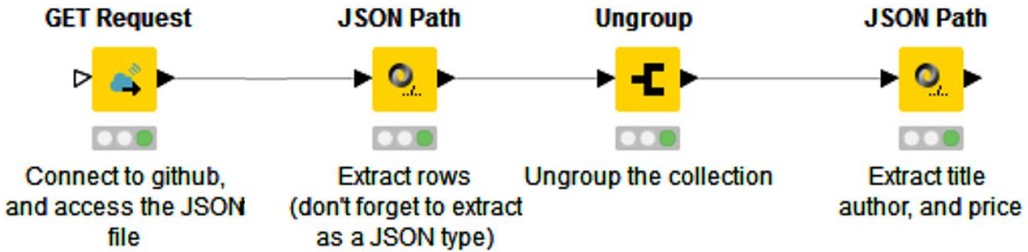
Web Services



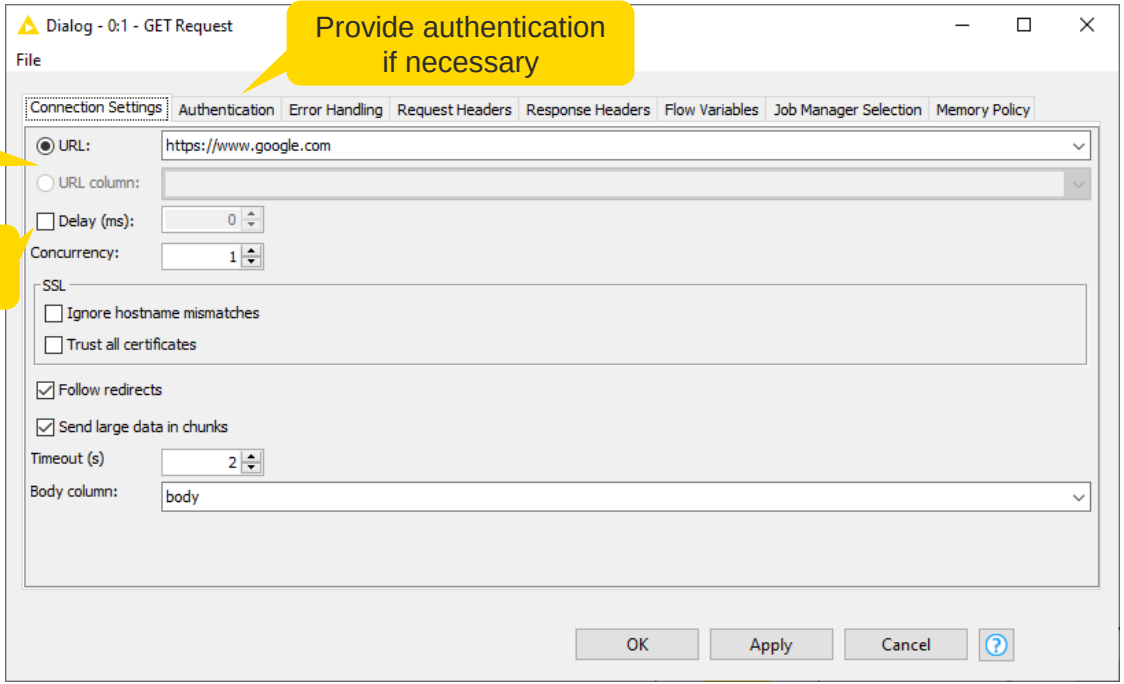
RESTful Web Services / API

- **Re**presentational **S**tate **T**ransfer
  - RESTful APIs are Web Service APIs that adhere to the REST constraints
  - One the most predominant architectures for obtaining and managing data across applications
  - Uses HTTP transfer protocol, resources as endpoints of requests, standard HTTP verbs
  - Simple to use
- Existing KNIME nodes





RESTful Web Services / API

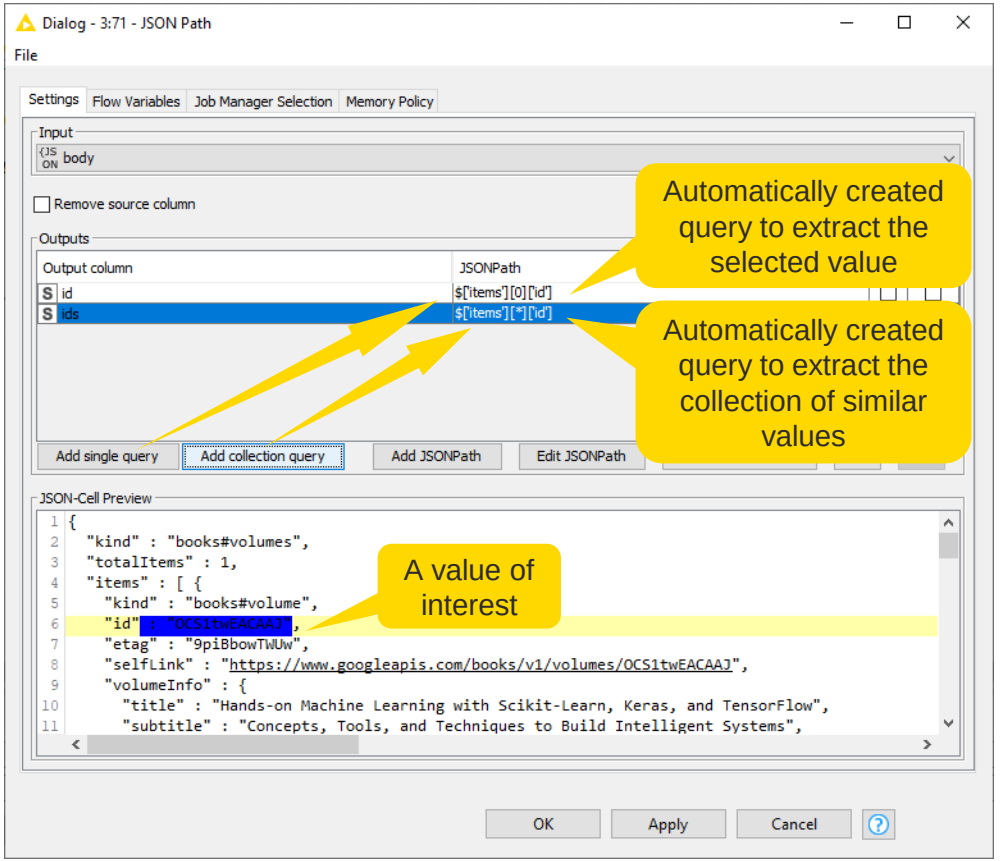


https://www.knime.com/blog/a-restful-way-to-find-and-retrieve-data

https://www.knime.com/blog/OSM-meets-CSV-file-and-Google-API

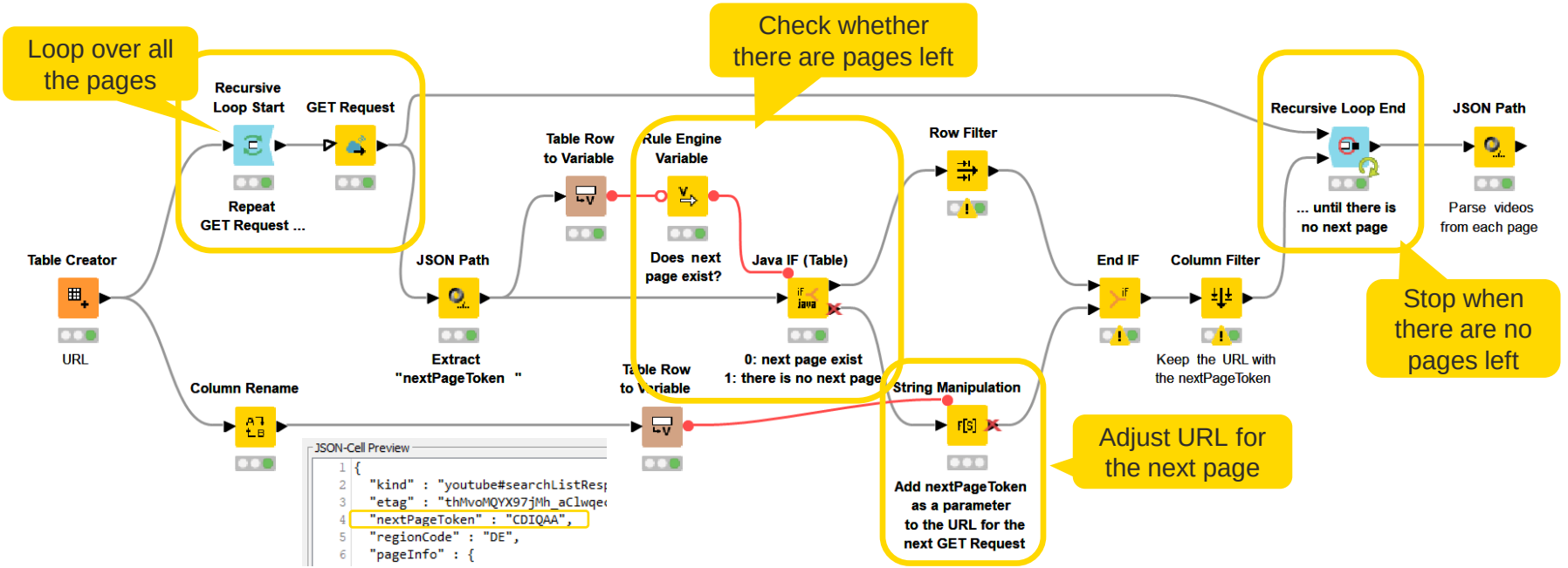
JSON and XML Parsing Tips

- Use the JSON Path node to query the JSON file and extract parameters
- Editor window simplifies construction of JSON queries by auto-generating them
  - Select the value of interest in the JSON-Cell Preview and use the buttons to automatically add a query to extract this single value or a collection of similar values
  - OR write a JSONPath query manually
- Analogously with Xpath



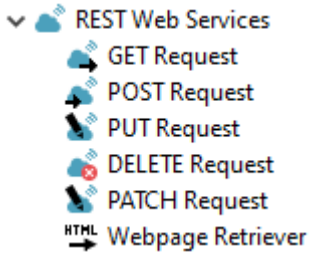
Pagination

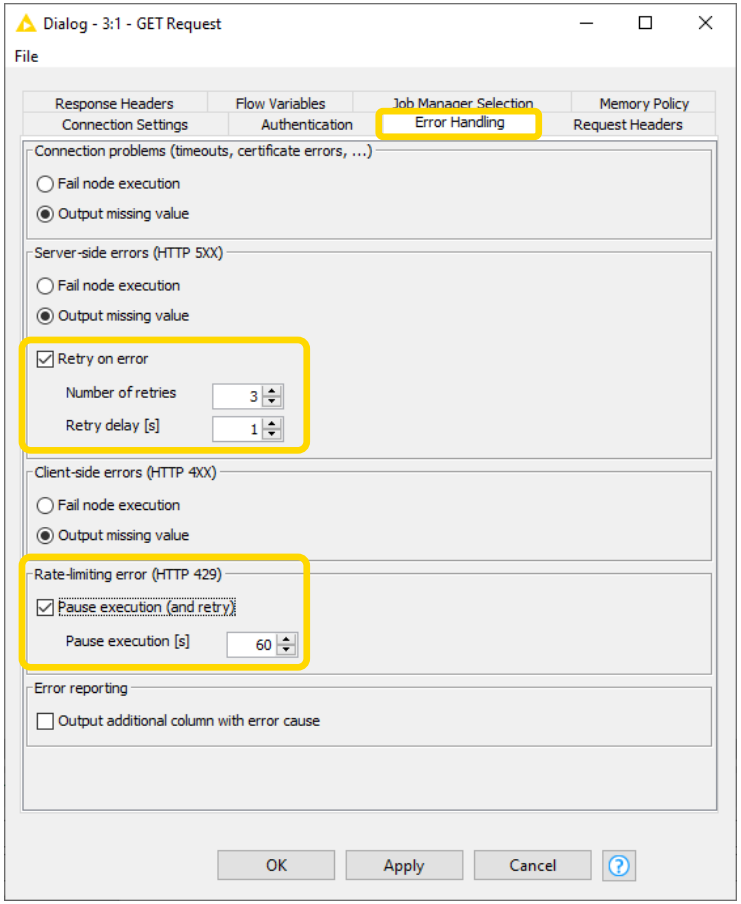
- Pagination is a process of dividing a content into small consumable pages
  - API requests to large collections / dense databases can return a massive response with many pages
  - That can result in a heavy workload for an API
  - Retrieving partial paginated results to handle responses easier and keep network traffic manageable



Error Handling

- Many errors might occur
  - client-side errors, server-side errors or rate-limiting conditions
  - can be individually configured, including the number of retry attempts or invocation pausing
- Specify node behavior in case of error directly in the node
- Works with all the nodes in KNIME REST Client Extension



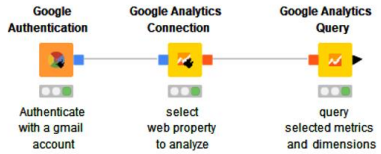


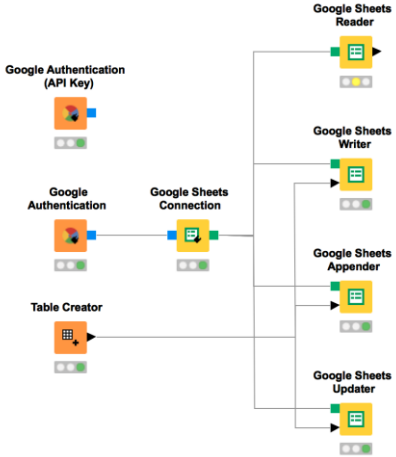
Other Web Resources

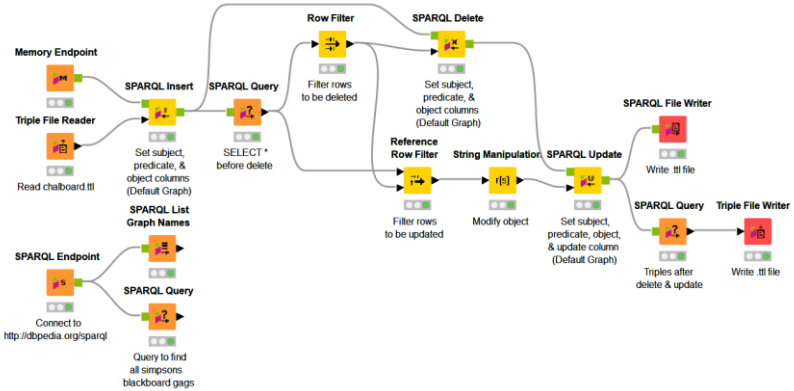
- Webpage Retriever
  - Retrieve the whole webpage by issuing HTTP GET requests
  - Use when no API is available or API endpoints are not enough
  - Provide Authentication if needed
  - Output as XML or String
  - Parse result with the XPath node (when possible)
- Web Servers
  - HTTP(s) Connector
  - SSH Connector
  - FTP Connector
  - Functionality similar to file system connectors
- RSS Feed

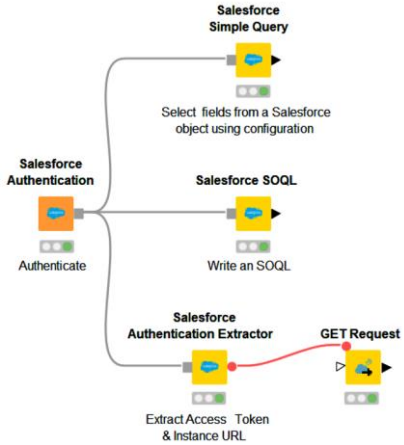
Dedicated Web and Cloud Services Connectors

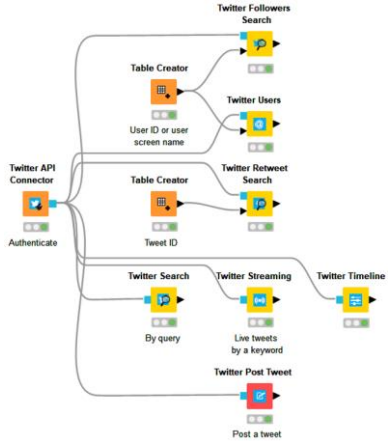
- Based on provider's API
- No need to write API requests
- Many extensions are available
  - Salesforce
  - Twitter
  - Semantic Web
  - Google Sheets
  - Google Analytics



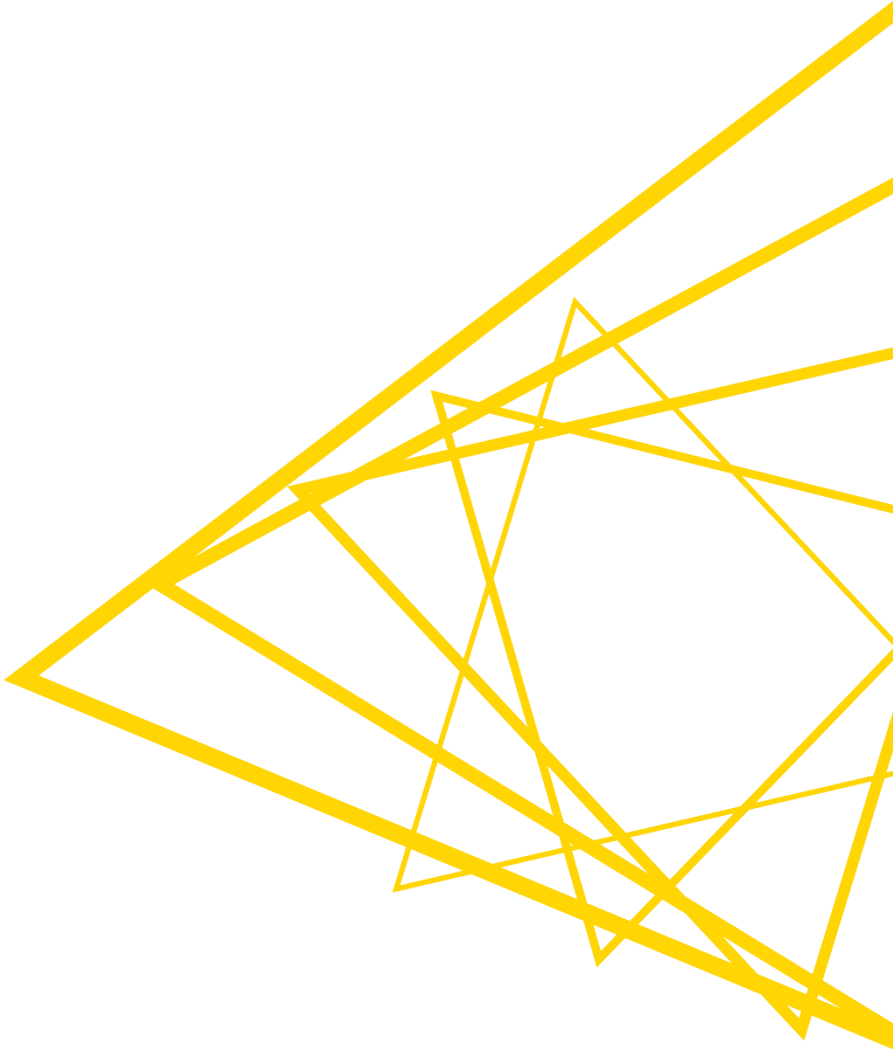






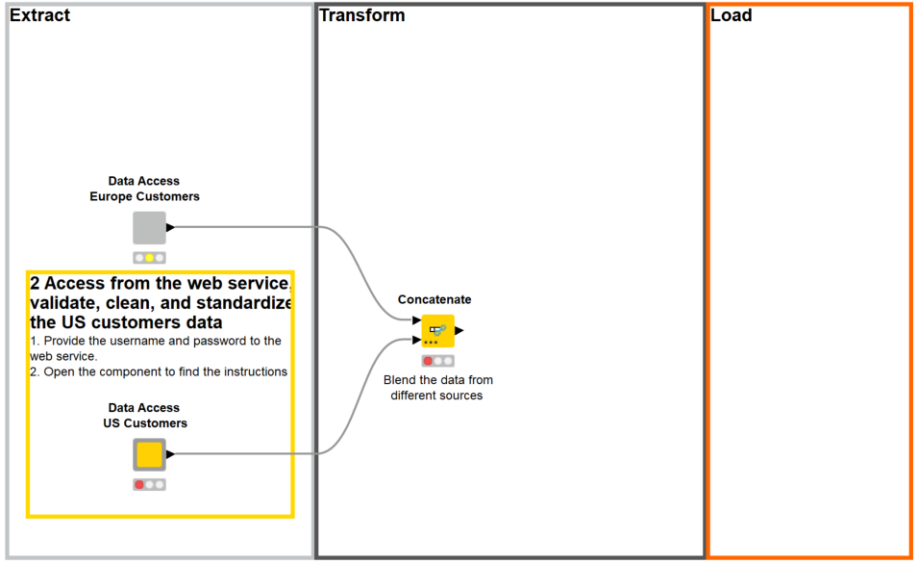


Demo



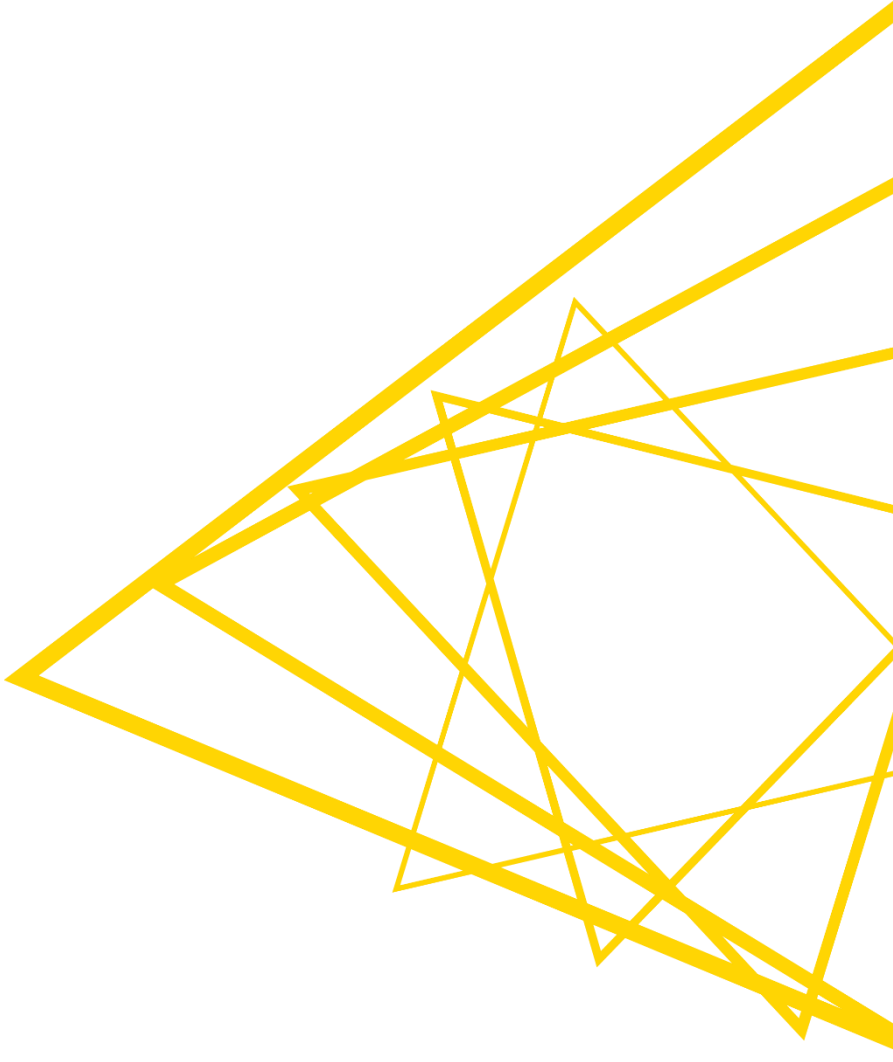
Exercise – 01.2_Extract_WebService_data&Blend

- This exercise is the second step to build application “ETL on Customers Data”
  - The solution to the previous exercises is already in the workflow
  - 2 Access from the web service, validate, clean, and standardize the US customers data
    - Credentials for the GET Request node can be found in the reminder email
  - Find detailed instructions in the workflow



Best Practices: ETL, Connectors & Data Access

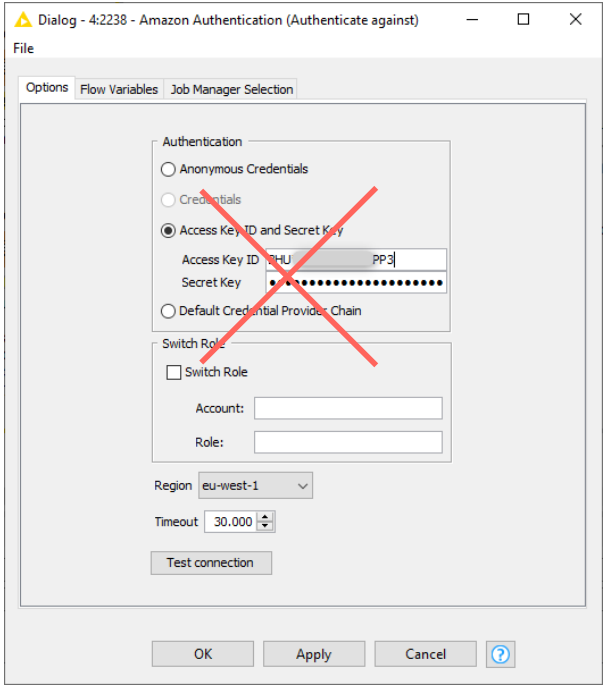
Security

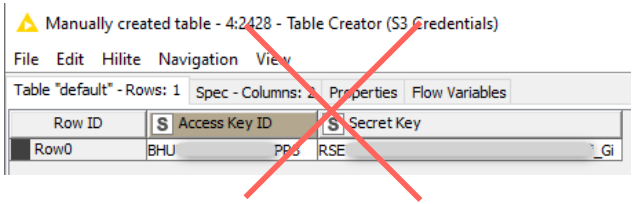


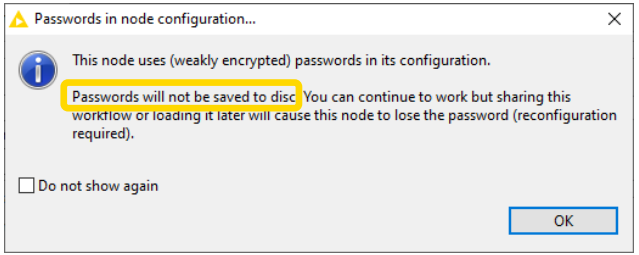
Credentials Security Considerations

- No passwords in plaintext anywhere
- Do not hardcode your credentials in nodes
- To always avoid storing passwords as part of the workflow on disc:
  - Specify permanently in the knime.ini in the root of the KNIME installation:

    -Dknime.settings.passwords.forbidden=true

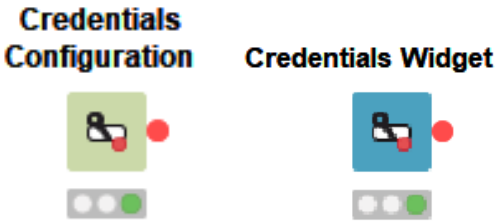


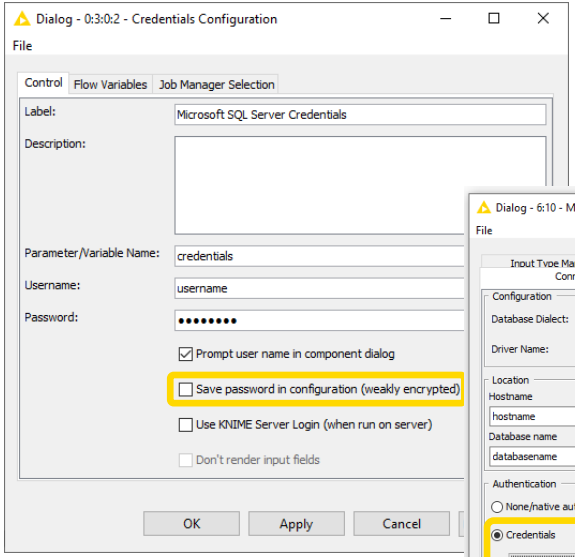


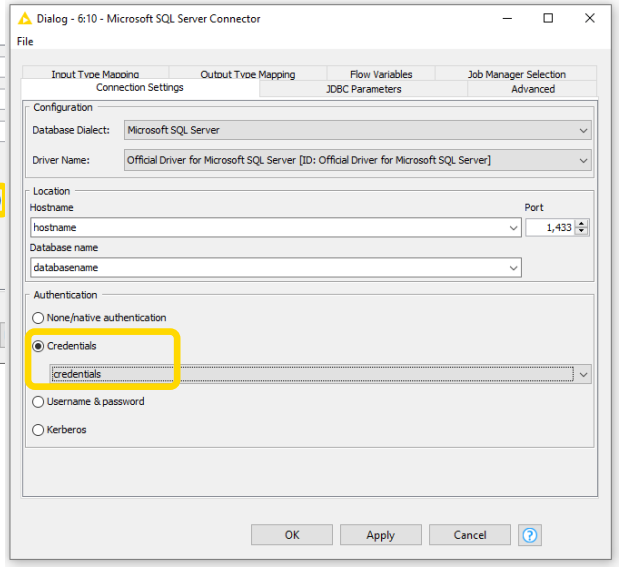


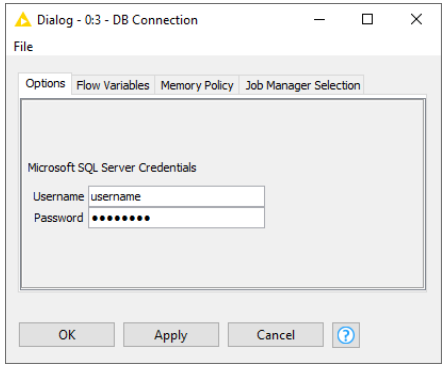
Handling Credentials

- Avoid saving credentials to the node
- Encrypt the credentials with the Credentials Configuration or Credentials Widget nodes
- Use for database connector, authentication, and REST nodes



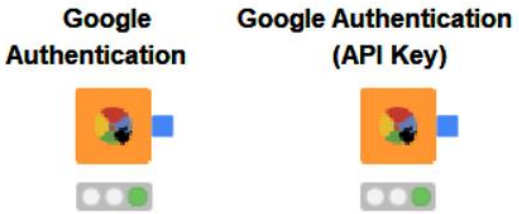


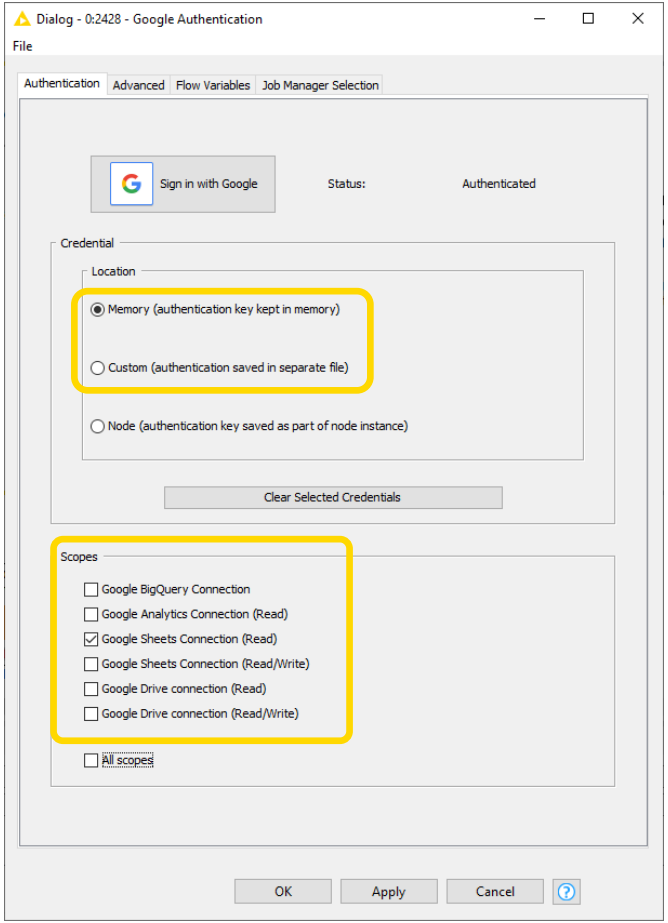




Handling Cloud Credentials

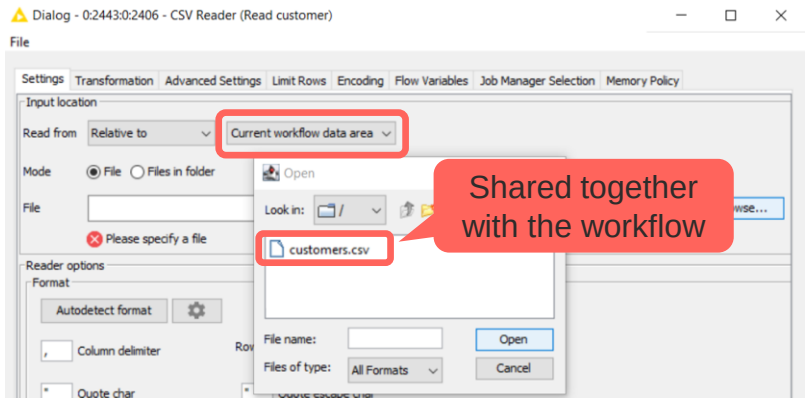
- Avoid saving the authentication credentials to the node (Node option)
- If you use the node more than once, make use of authentication files (Custom option)
- If you use the node only once, keep the authentication credentials in memory (Memory option)
- When possible, limit the available scopes

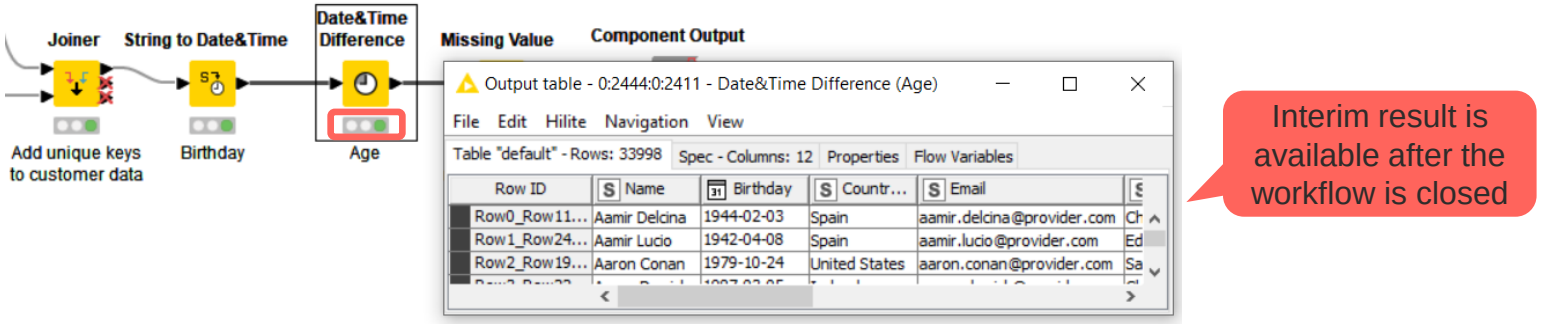




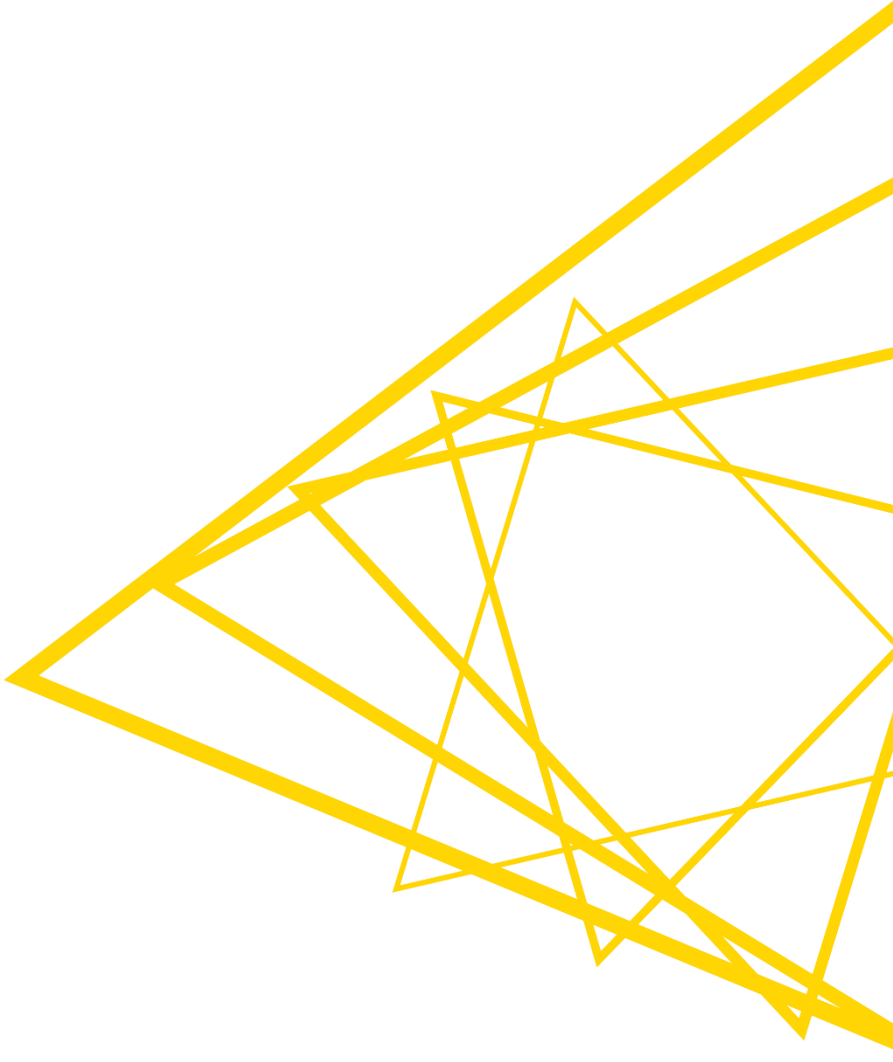
Handling Sensitive Data

- Do not save confidential data inside your workflow
- Reset your workflow before sharing it
  - unless there is a specific reason not to



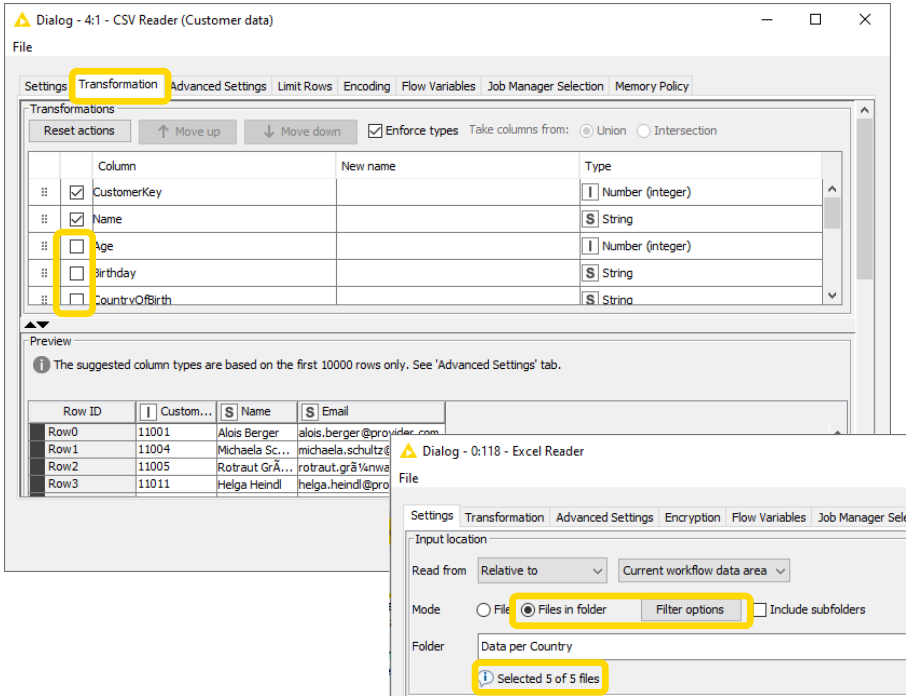


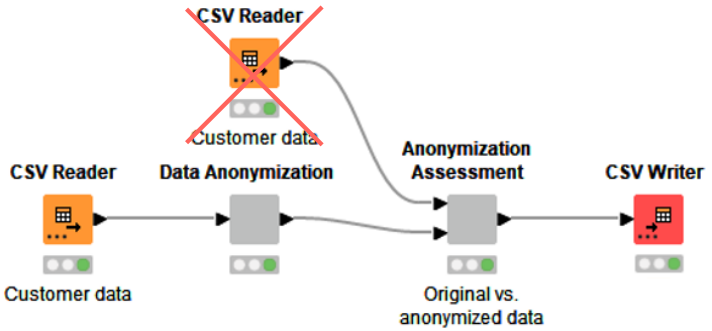
Efficiency

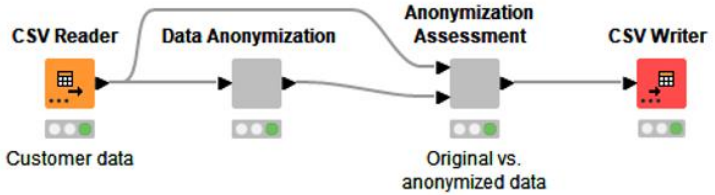


Efficient Reading of Files

- Remove redundant and unused columns in the “transformation” tab of a reader node
- Avoid reading the same file multiple times
- Read multiple files with the same structure with one Reader node using the “Files in folder” option



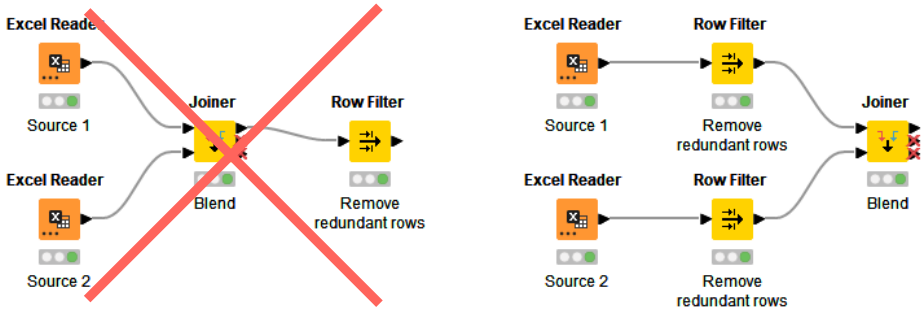




Efficient Workflow Design

- Not every manipulation operation in your data will be equally expensive
  - Joining two tables with a Joiner node will be more expensive than replacing a few rows with a Cell Replacer node
  - Only use loops when absolutely needed – use Multi Column nodes instead

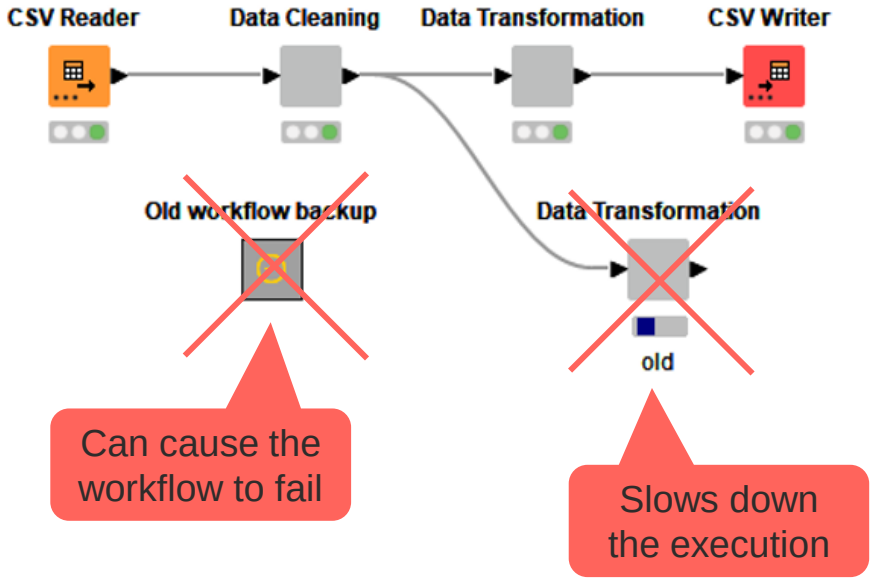
- Not every data type in KNIME uses the same ‘memory space’:
  - Strings occupy more space in memory than integers
  - Consider removing columns with constant values if they are not needed
- Remove redundant data early
  - Filter redundant rows or columns before dragging them through the data pipeline and other data operation, e.g., before joining



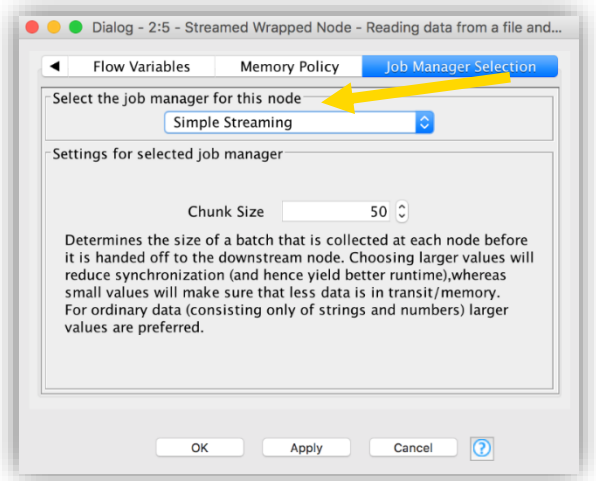
Efficient Workflow Design

Absolutely avoid:

- Disconnected nodes/components, especially if that workflow will run on a KNIME Business Hub or will be called by another workflow
- Workflow branches that are not actively needed
- Storing large unused files inside of the workflow
  - It will cause the workflow to take a long time to load/save

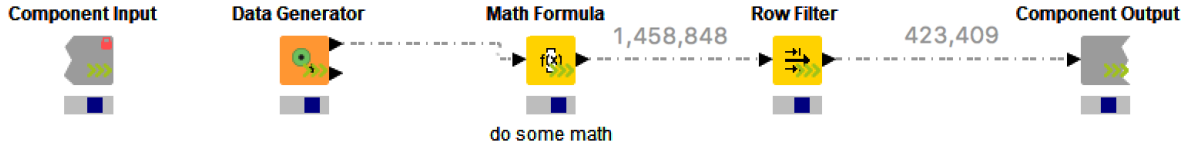


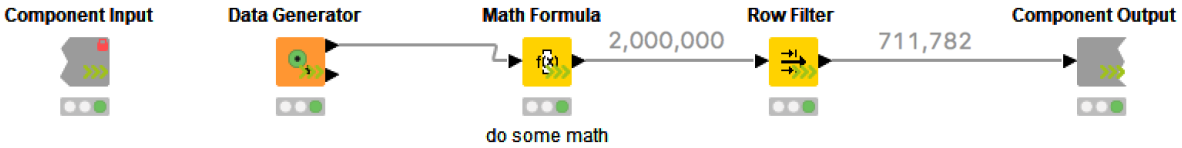
Additional Efficiency Tip: Streaming (Beta Feature)



Streamed Component - Reading data from a file and process

Sub Workflow as Component.
To open it:
- right click > Component > Open
- Ctrl + Double Click





Performance

- The execution of the deployment application must be fast
  - keep an eye on scalability
  - address the pain points in the execution
- Test for speed on large datasets (Timer Info node)
  - Find out which nodes take the longest to execute
  - Inspect what is KNIME-related and what could be related to other variables (i.e. slow network connection)
  - Measure how much you have improved your execution time – keep track of success

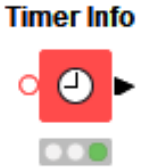
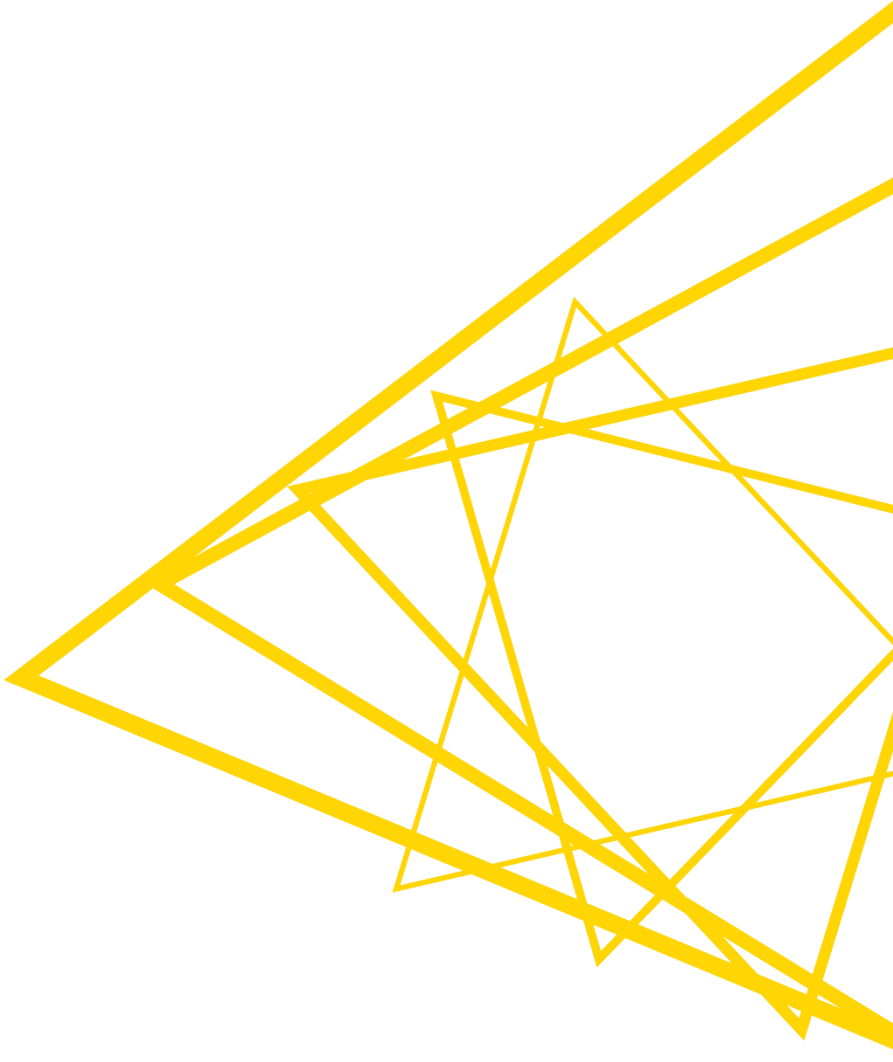


Table "default" - Rows: 8 | Spec - Columns: 8 | Properties | Flow Variables

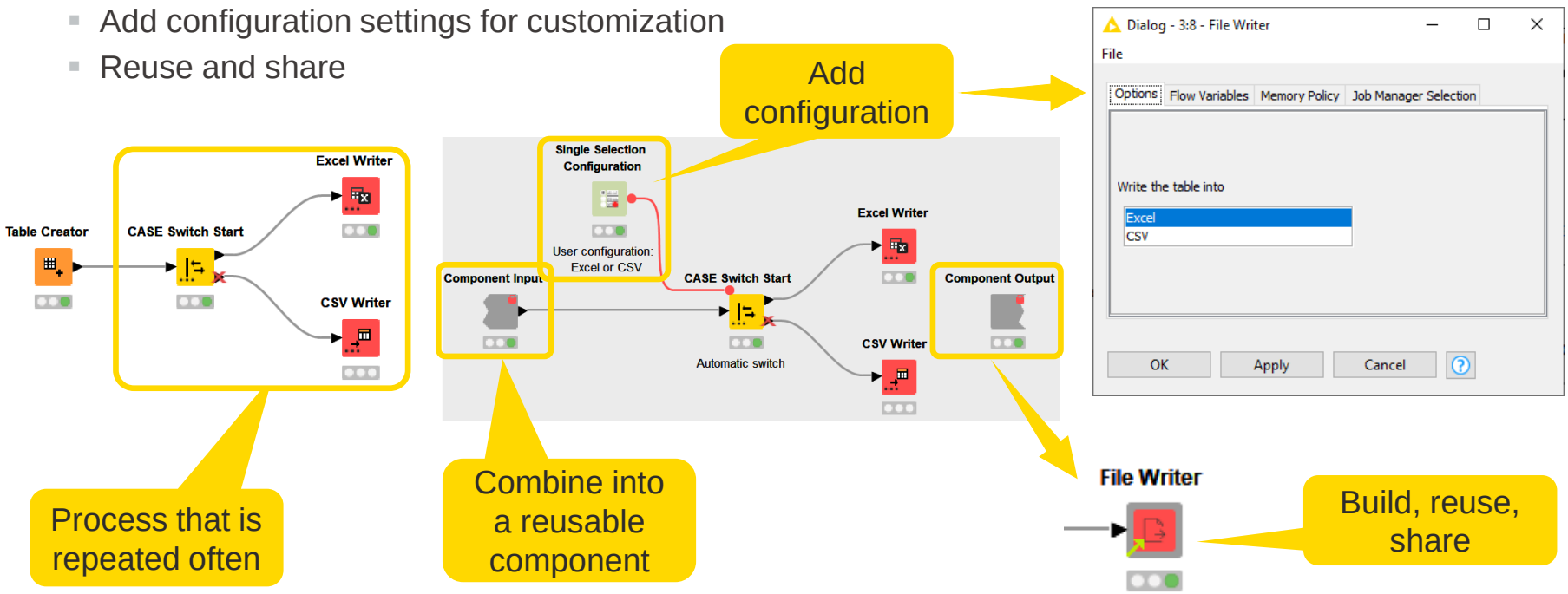
Row ID	Name	Execution Time	Execution Time since last Reset	Execution Time since Start	Nr of E...	Nr of E...	NodeID	Classname
Node 1	CSV Reader	892	892	892	1	1	4:1	org.knime.filehandling.core.node.table.reader.TableReaderNodeModel
Node 2	Excel Reader	832	832	832	1	1	4:2	org.knime.ext.poi3.node.io.filehandling.excel.reader.ExcelTableRea...
Node 3	Cell Replacer	17	17	17	1	1	4:3	org.knime.base.node.preproc.cellreplace.CellReplacerNodeModel
Node 18	SQLite Conn...	274	274	274	1	1	4:18	org.knime.database.extension.sqlite.node.connector.SQLiteDBConn...
Node 19	DB Table Sel...	31	31	31	1	1	4:19	org.knime.database.node.utility.tableselector.DBTableSelectNodeMo...
Node 20	DB Reader	107	107	107	1	1	4:20	org.knime.database.node.io.reader.DBReadNodeModel
Node 21	Joiner	91	91	91	1	1	4:21	org.knime.base.node.preproc.joiner3.Joiner3NodeModel
Node 22	Timer Info	?	0	0	0	0	4:22	org.knime.base.node.util.timerinfo.TimerinfoNodeModel

Reusability



Reusability via Components

- Logic blocks and repeatable processes can be written once and reused
- In KNIME Analytics Platform – by means of reusable components
  - Wrap a process into a component
  - Add configuration settings for customization
  - Reuse and share



Benefits of Using Components within an Organization

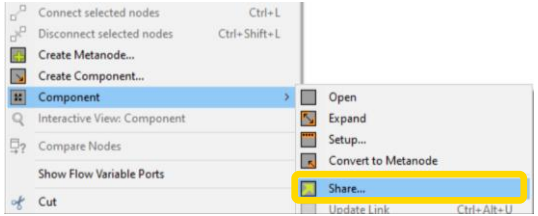
- Standardization
  - You own organization's best practices and procedures can be implemented as components
  - Component-driven design & development helps promote creation of reusable processes with a single responsibility. For example…
    - Organization-specific database connectors
    - Standardized ETL cleanup operations
    - Standardized logging operations
    - Model API interfaces (i.e., schema validation and transformation of JSON input/output)
- Adherence to modern testing and design principles
  - Components help teams follow the Don't Repeat Yourself (DRY) principle
  - Components are small, independently testable units of functionality.
- Workflow organization
  - Components can be nested inside other components, allowing complex workflows to be comprised of small, maintainable parts.

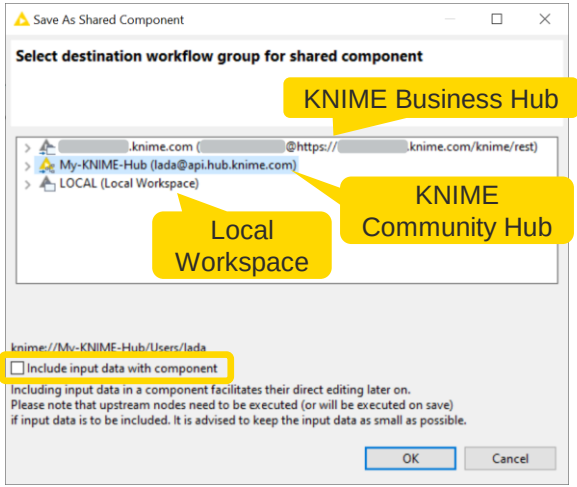
Best Practices for Components Building

- Single function or responsibility
- Small set of consistent inputs and outputs
  - Input should be validated, e.g., expected number of columns, rows, column types, value ranges, etc.
  - Breakpoint node for unexpected input – stop execution with a custom error message
  - Similar output for similar input unless randomness is needed
- Should not rely on the workflows that use them in order to function properly
- Avoid dependencies on specific data sources
  - Exception: components whose purpose is connecting to a data source
- Should be self-contained
  - By default, variables from the workflow aren‘t available inside the component
  - By default, variables created inside are only available locally inside the component
  - Configure Component Input/Output to pass flow variables from/to outside the component
- Should be properly documented
  - Describe the function, set of inputs, configuration settings, and set of outputs

Best Practices for Components Sharing

- Include input data with component
  - When selected, everyone with access to the component will access it together with its current input data
  - Make sure that no sensitive data are shared
- Destination
  - Make sure that everyone who uses the component can access it and update its instances in their workflows

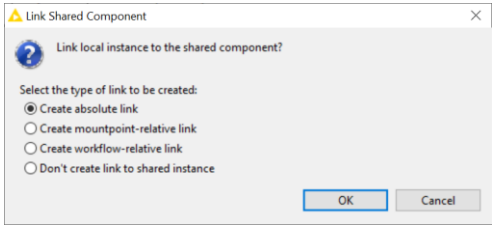




Local Workspace	Business Hub	Community Hub
Access from the local KNIME Analytics Platform installation	Access from the KNIME Business Hub	· Private: access logging in to your KNIME Community Hub profile · Public: available to everyone
Share for own usage	Share with colleagues who have access to the hub instance	Share for own usage, with colleagues who have access to the space, or publicly

Best Practices for Components Sharing

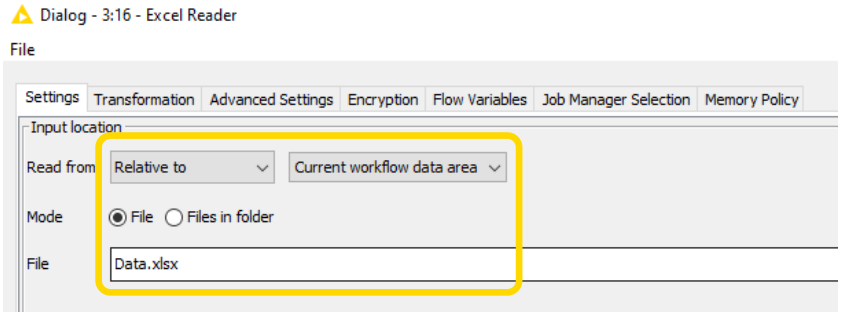
- Link
  - Make sure a workflow always can find a component when update is needed



	Description	Usage
Absolute	Absolute location of the shared component **knime://LOCAL/Destination/Component**	Local Workspace: own usage Hub: share with others
Mountpoint-relative	Path is relative to the current mountpoint Similar folder structure w.r.t. the mountpoint is required **knime://knime.mountpoint/Destination/Component**	For sharing the components used in the same mountpoint
Workflow-relative	Path is relative to the current workflow Similar folder structure w.r.t. the workflow is required **knime://knime.workflow/../Destination/Component**	For sharing the components used in the same workflow or workflow group
Share but continue to work on the unlinked instance	Shared component has an absolute link Current instance isn't linked to the shared component **knime://LOCAL/Destination/Component**	To create a component but continue to work on the current instance

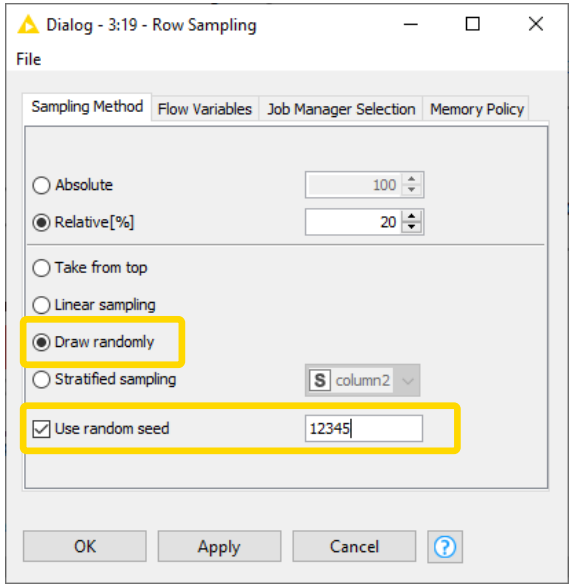
Replication vs. Security

- Include data in the workflow data area to allow for replication when sharing
  - BUT Don’t include sensitive data that shouldn’t be shared
  - BUT Keep large files outside the workflow to avoid slow loading / saving
- Use relative paths instead of absolute paths
  - Absolute paths aren’t optimal for sharing, reusability, and security
  - With the appropriate relative path, the workflow will work everywhere
- Do not hardcode the credentials in connector nodes
  - Make use of the Credentials Configuration node

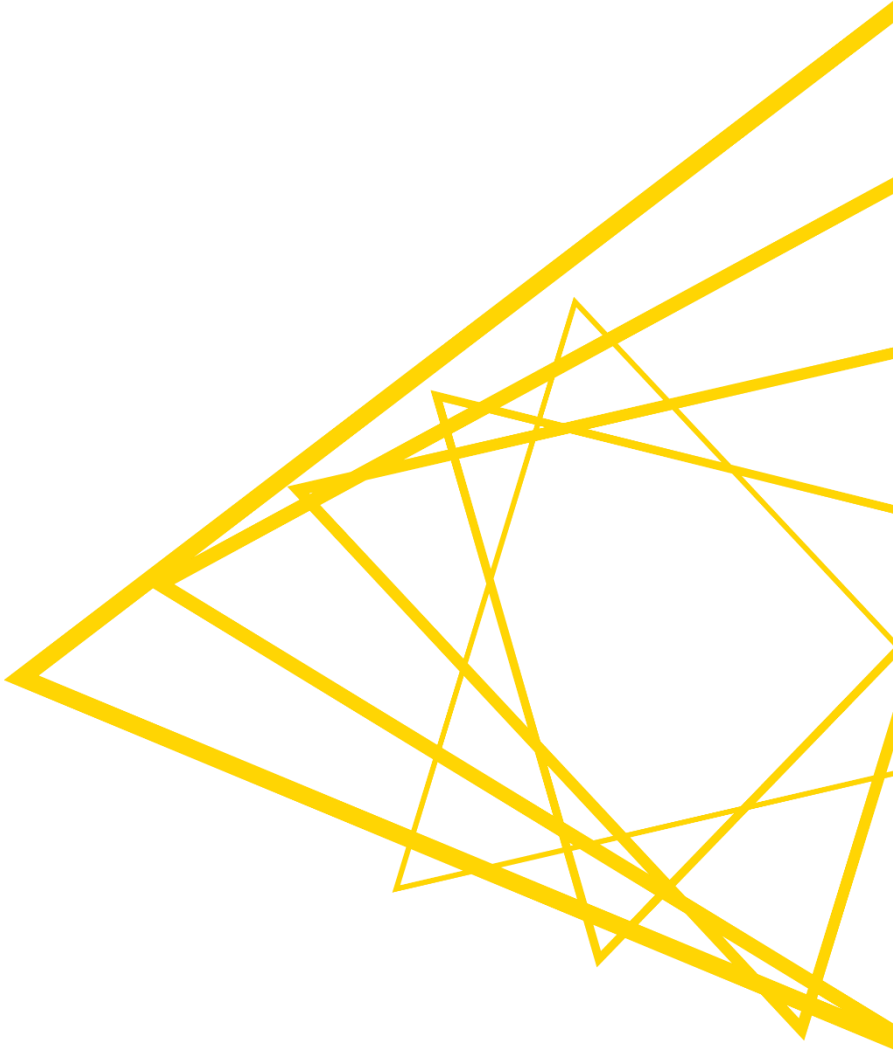


Replication

- Use seeds in the nodes that use (pseudo)randomness to be able to replicate the results



Demo

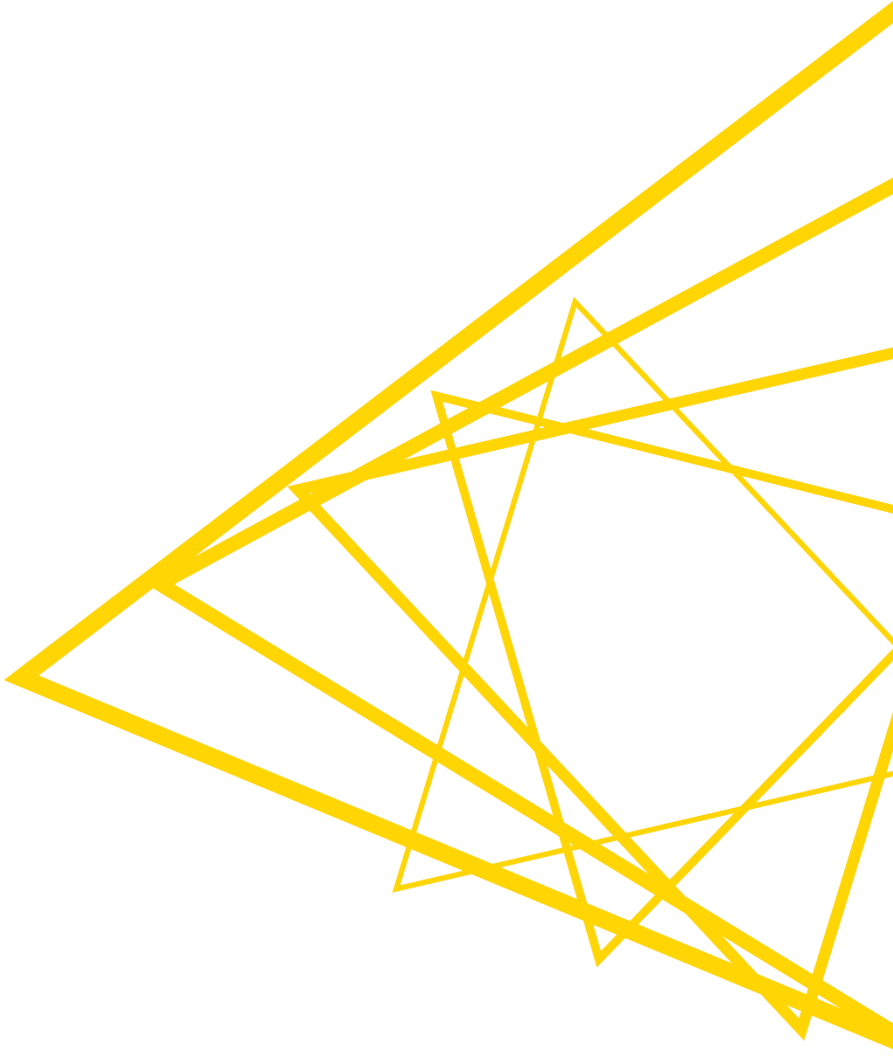


Session 1: Summary

Now you should be able to:

- Recognize various KNIME connector nodes
- Access, validate, and parse data from a web service and a data lake
- Apply best practices regarding security, efficiency, reusability, and data validation
- Build the first part of the application that accesses, validates, processes, and blends the new customer data from two different sources

Session 2: ETL, Data anonymization, Databases

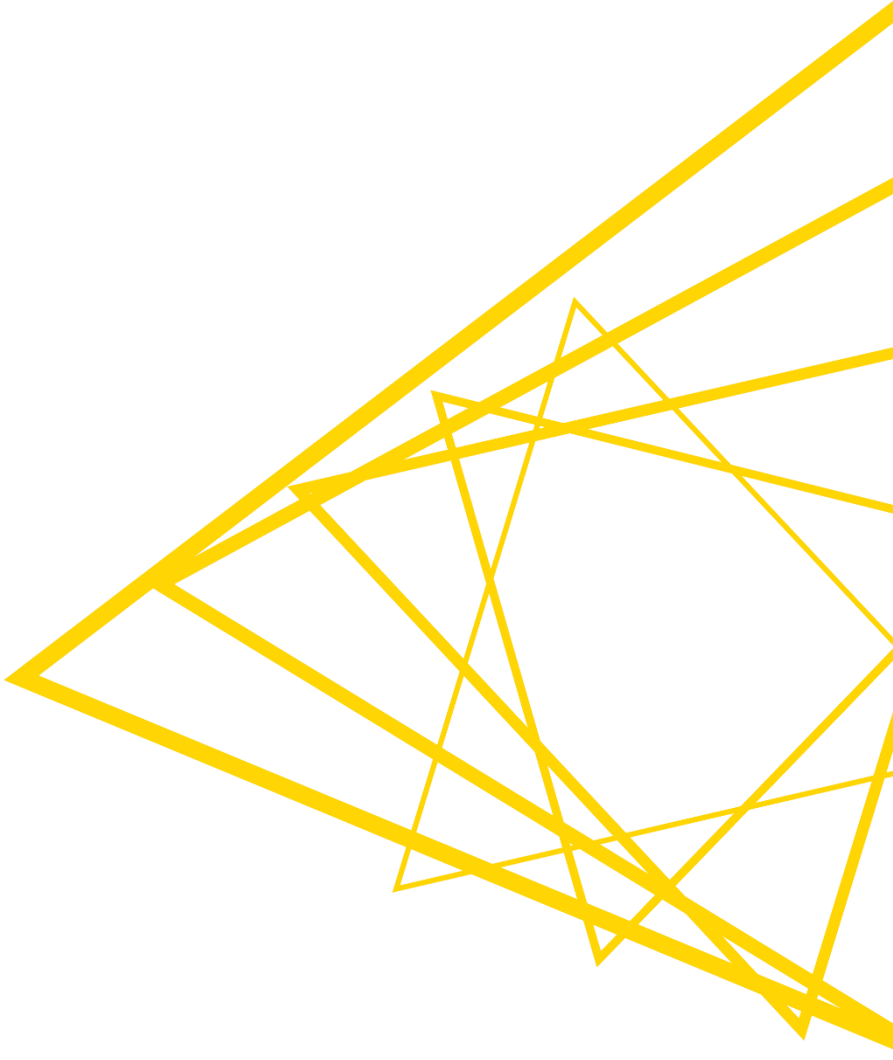


Session 2: Learning Outcomes

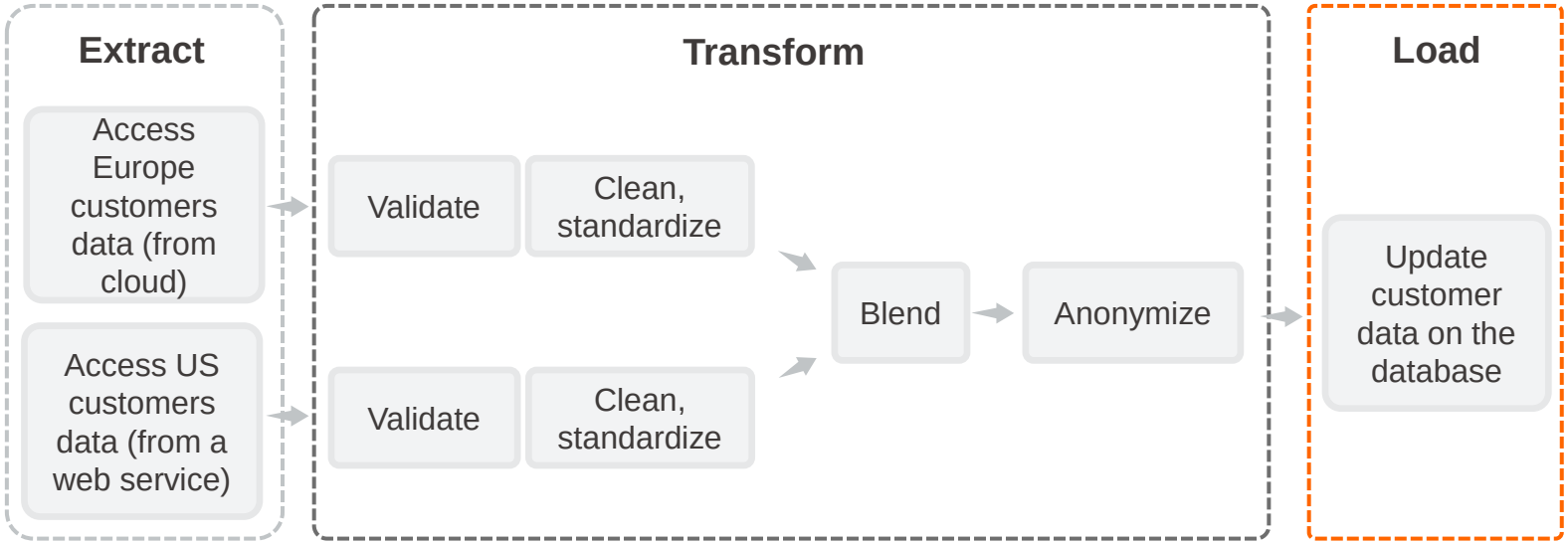
At the end of this session, you will be able to:

- Apply various data anonymization techniques
- Recognize advanced KNIME Database extension functionality
- Query and update a database records
- Build the application that anonymizes the new customer data and uploads it to the database

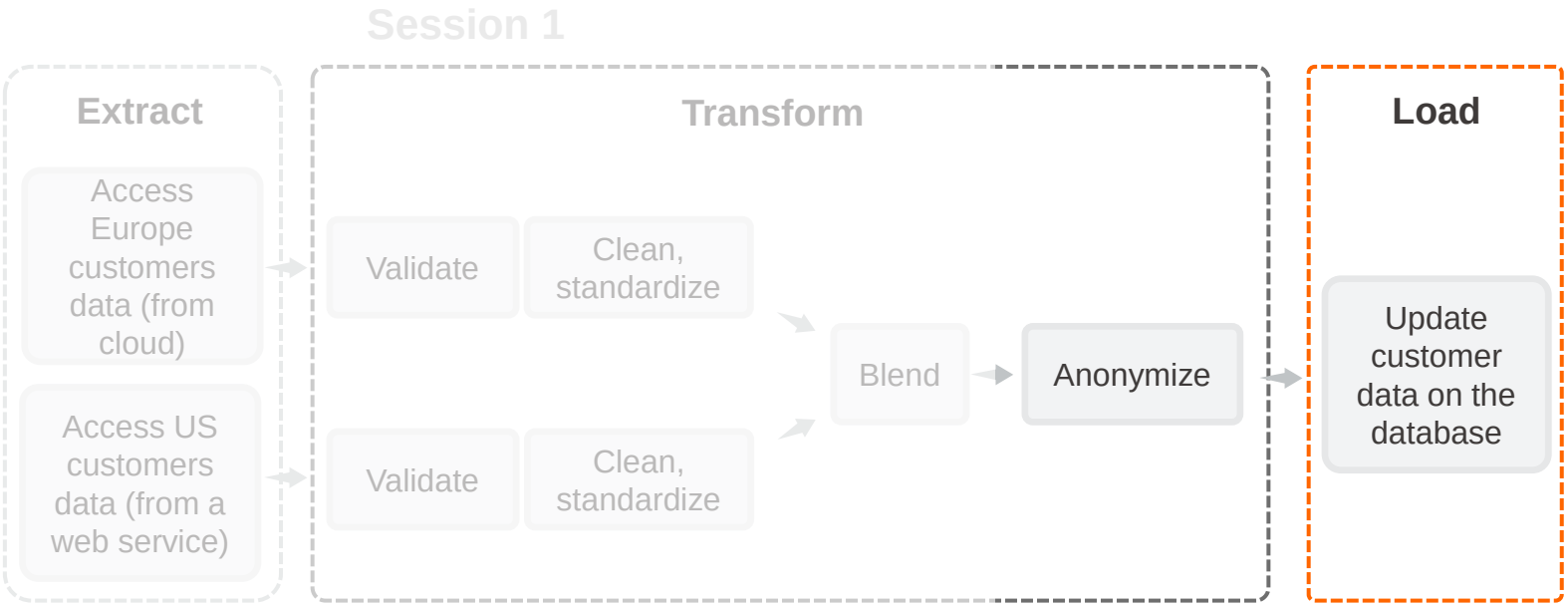
Today‘s example: ETL on Customers data – Part II



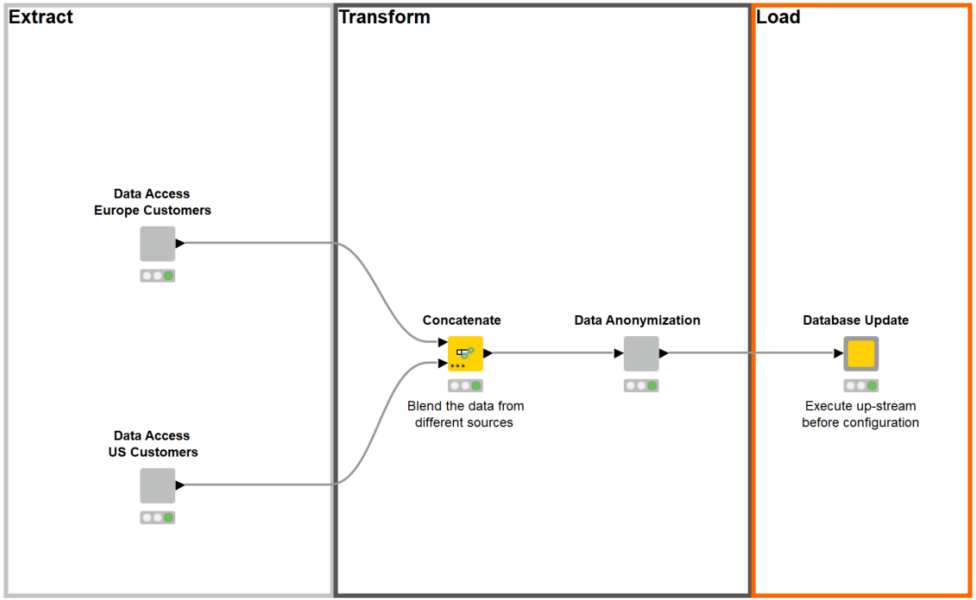
Today’s Example: ETL on Customers Data



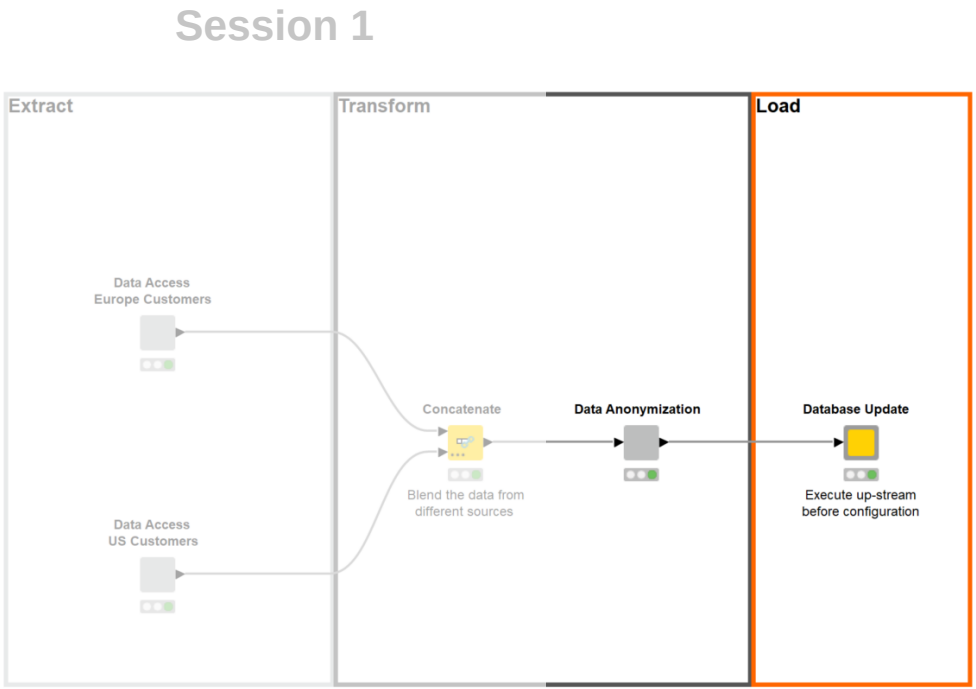
Today’s Example: ETL on Customers Data



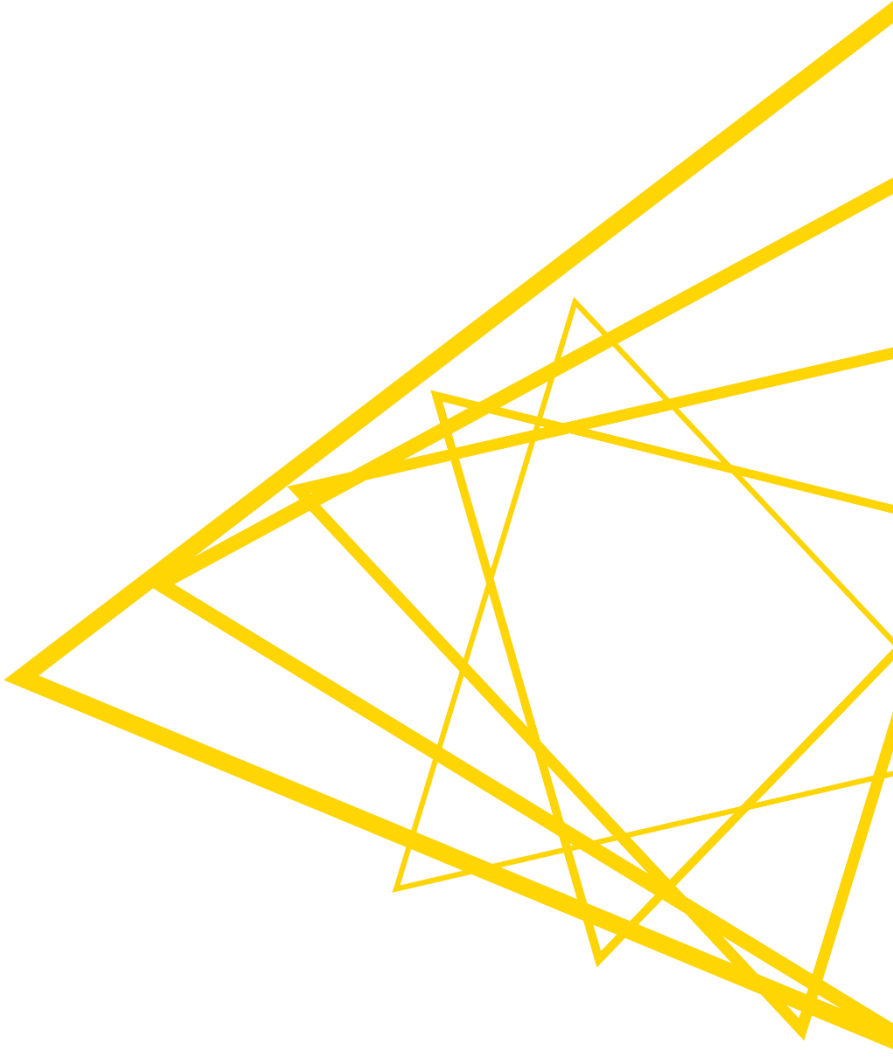
Today’s Example: ETL on Customers Data



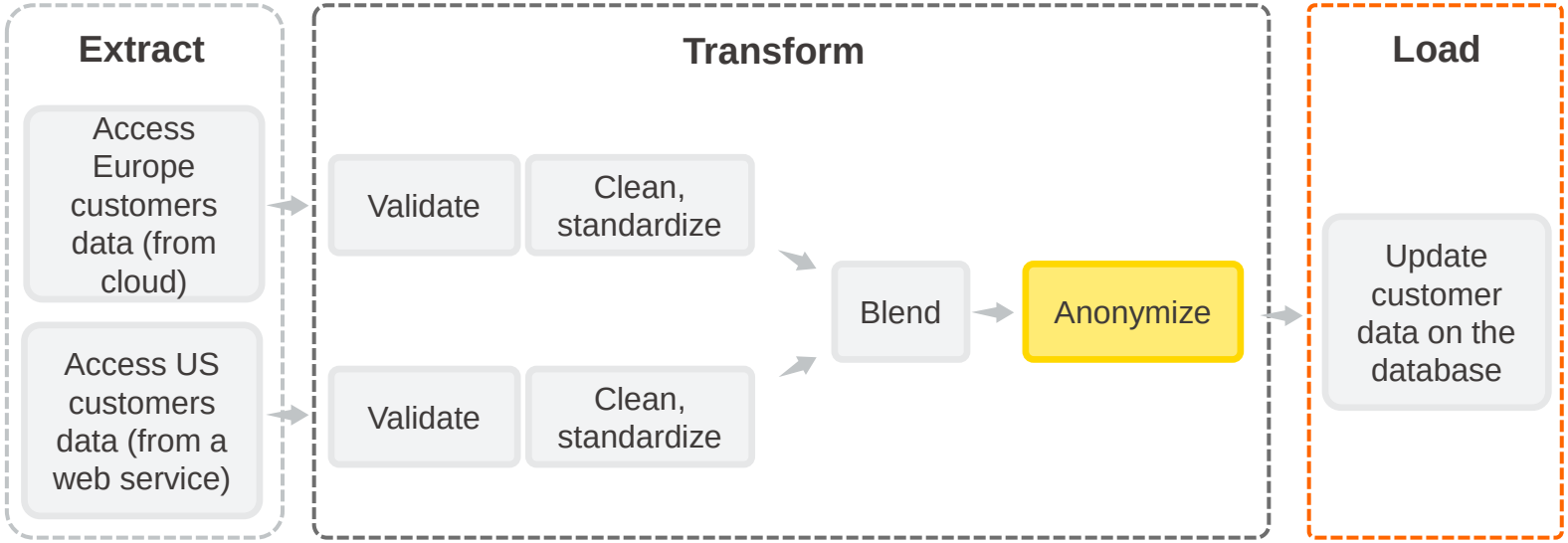
Today’s Example: ETL on Customers Data



Data Anonymization

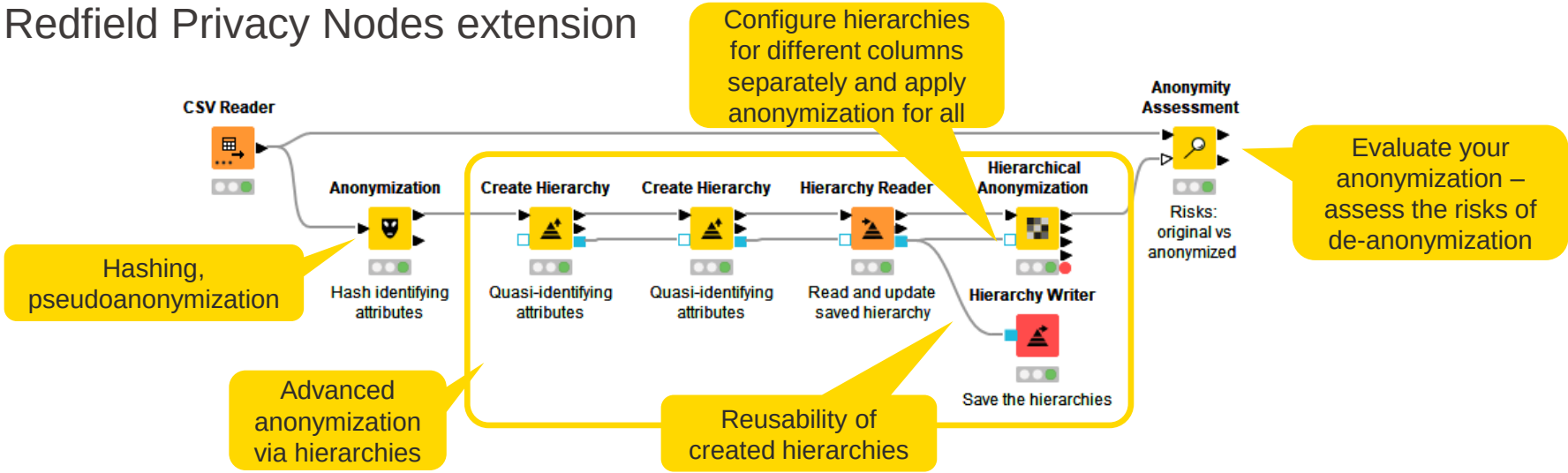


Data Anonymization



Data Anonymization

- Sensitive / personal data needs to be handled with caution
- GDPR is stipulating that only anonymized data may be used extensively
  - Transform the data so that it is impossible to re-identify the individuals in the dataset
  - Assess the risks of de-anonymization
- Redfield Privacy Nodes extension



- https://www.knime.com/blog/data-anonymization-in-knime-a-redfield-privacy-extension-walkthrough
- https://www.youtube.com/watch?v=5e6hd0LWNDo

Attribute Types & Anonymization Forms

- Attribute types
  - **Identifying** – attributes that identify a person precisely, e.g., name, email, ID number, etc.
  - **Quasi-identifying** – attributes that identify a person indirectly, e.g., date of birth + gender + zip code + additional information
  - **Sensitive** – attributes that should not be matched to a specific person, e.g., medical diagnoses, sexual orientation, religious views.
  - **Non-sensitive** – all the other attributes that can be useful for data analysis
- Anonymization forms (available in KNIME Analytics Platform)
  - **Suppression** – entire removal of values in specific columns, e.g., identifying information
  - **Character masking** – partial modification of values with non-meaningful characters (e.g., “x” or “*”)
  - **Pseudoanonymization** – replacement of values with values that do not contain any useful information, e.g., hashing and tokenization
  - **Generalization** – aggregated of original values, e.g., mean, mode, or bins, instead of original values

Pseudoanonymization via Hashing

- Hashing is transforming a given key into another value
- SHA-1 (Secure Hash Algorithm 1) is a cryptographic hash function which takes an input and produces a 160-bit hash value (40 digits long)
- Problems
  - Enumerating all the possible original values and hashing them can lead to deanonymization
  - Possibility of collision – two different keys are hashed into a similar value
  - Other columns and datasets can be used to re-identify the person
- Solution: Salting
  - Concatenate original string with random value, values from the other column, or a timestamp
- Anonymization node from Redfield Privacy Nodes extension

Name	Name_salted	Name_anonymized
L. Messi	L. Messi-5106534569952410475	6ae704fdfdfa8d5a19672364ee9f6c3d9d20efeb
Cristiano Ronaldo	Cristiano Ronaldo-167885730524958550	6ea5a126ae64717440a34e26f72cf65172922727
Neymar Jr	Neymar Jr4672433029010564658	16786d48048655658e3a1c93292eadb45d3c8649
De Gea	De Gea-7216359497931550918	476ed16fd2f7f70085c8cb498eb8bce095f9cfc0
K. De Bruyne	K. De Bruyne-3581075550420886390	d4a31af79bd24d80052f11604ed0547fda2c4959

Anonymization Node

- Hashes the values with SHA-1 function

Name	Age
Giuseppina Nitsch	64
Natalija WeiÃŸe	63
Victor Koll	61
Stavros Selle	57

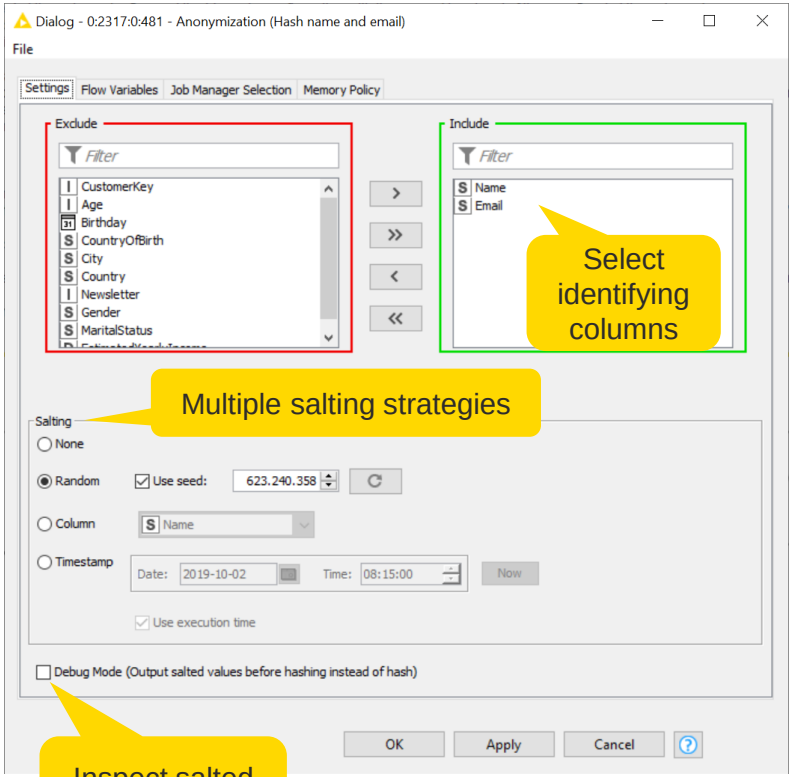
Anonymization

Original table with hashed values

Name	Age
49badac74b4c13150cf112c0cd0a0b58d1ff62f0	64
66f2debbba4e97547de56954f87fa5dac19a7...	63
588b78cf946e0f4e68204b81eba3068bd5c08...	61
93ab48f8fef4cba94763257a48af47babb02950e	57

Name	Name_anonymized
Giuseppina Nitsch	49badac74b4c13150cf112c0cd0a0b58d1ff62f0
Natalija WeiÃŸe	66f2debbba4e97547de56954f87fa5dac19a7...
Victor Koll	588b78cf946e0f4e68204b81eba3068bd5c08...
Stavros Selle	93ab48f8fef4cba94763257a48af47babb02950e

Return dictionary of original and hashed values – should be properly secured

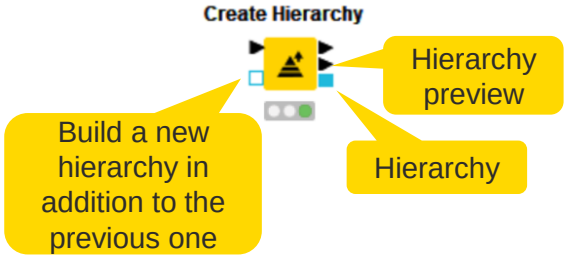


Generalization – Creating Hierarchies

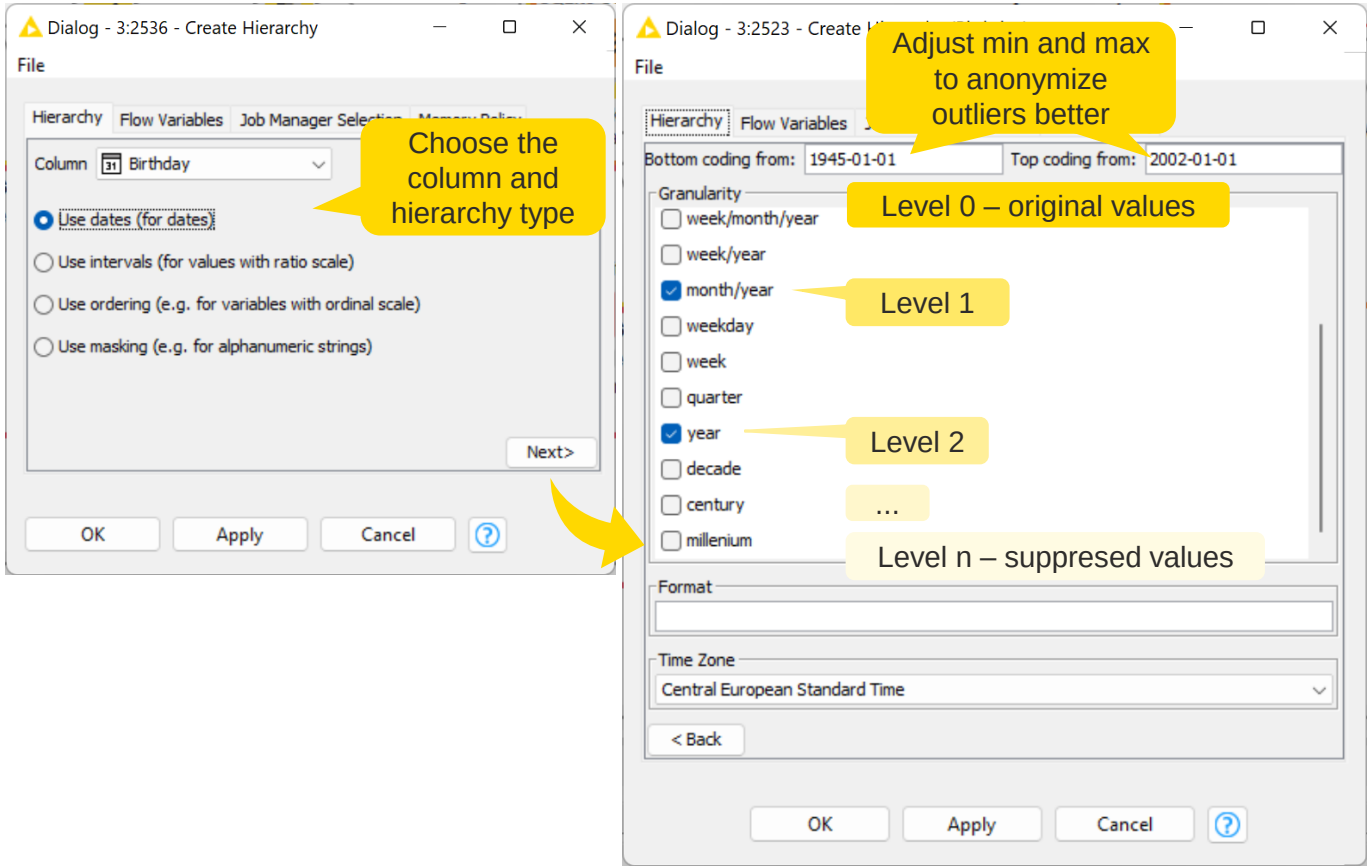
- Define complex binning rules with multiple layers that go from original data to less and less accurate, and finally to completely suppressed data
  - Date-based
  - Interval-based
  - Order-based
  - Mask-based
- Create Hierarchy node from Redfield Privacy Nodes extension

Accurate → Suppressed

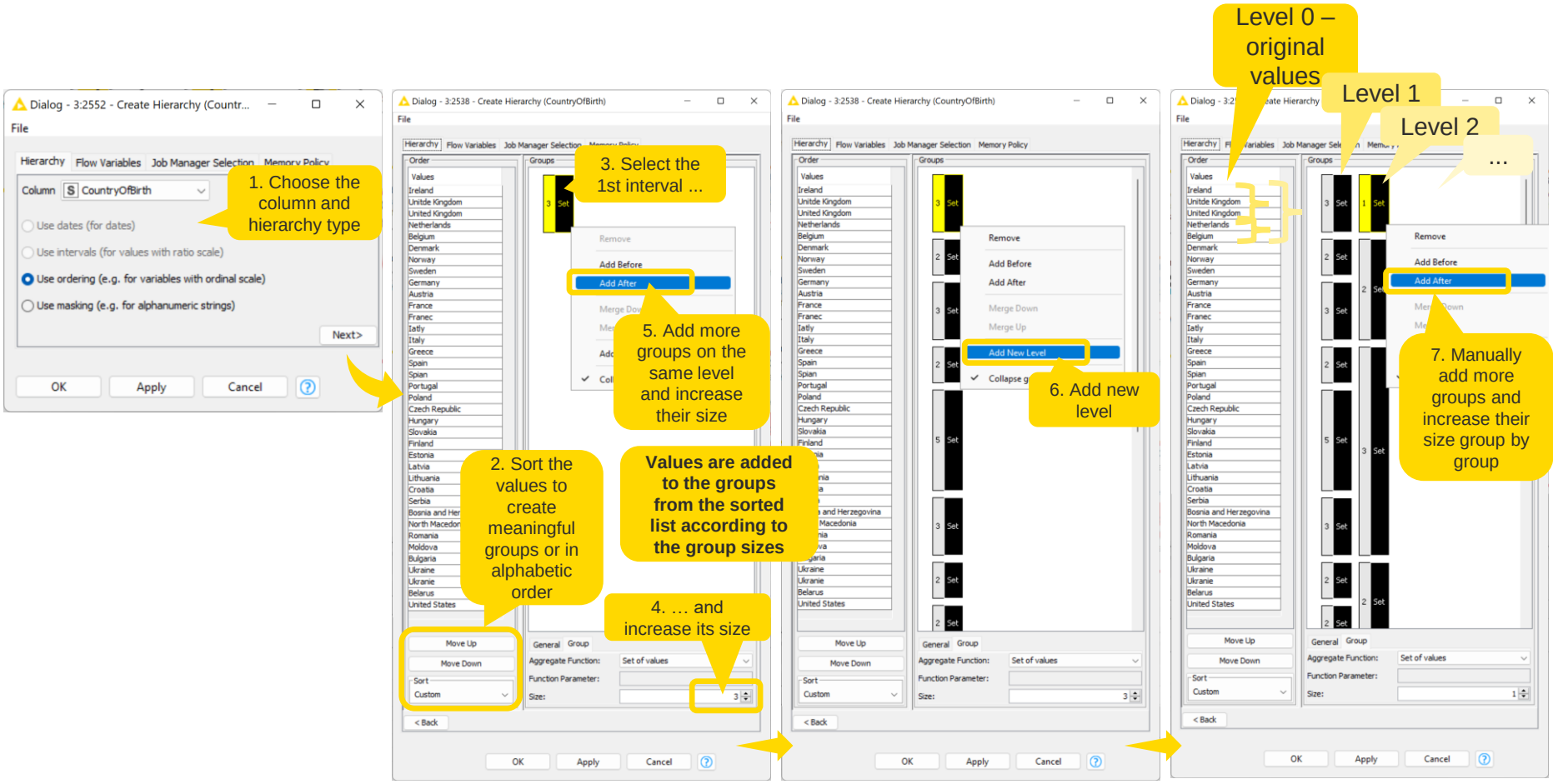
Level-0	Level-1	Level-2	Level-3
Spain	{Spain, Spian, Portugal}	{Germany, Austria, France, Franec, Iatly, Italy, Greece, Spai...	*
Slovakia	{Hungary, Slovakia}	{Poland, Czech Republic, Hungary, Slovakia}	*
Serbia	{Croatia, Serbia, Bosnia and H...	{Croatia, Serbia, Bosnia and Herzegovina, North Macedonia, ...	*
Romania	{Romania, Moldova, Bulgaria}	{Croatia, Serbia, Bosnia and Herzegovina, North Macedonia, ...	*
Portugal	{Spain, Spian, Portugal}	{Germany, Austria, France, Franec, Iatly, Italy, Greece, Spai...	*
Poland	{Poland, Czech Republic}	{Poland, Czech Republic, Hungary, Slovakia}	*
Norway	{Denmark, Norway, Sweden}	{Netherlands, Belgium, Denmark, Norway, Sweden}	*



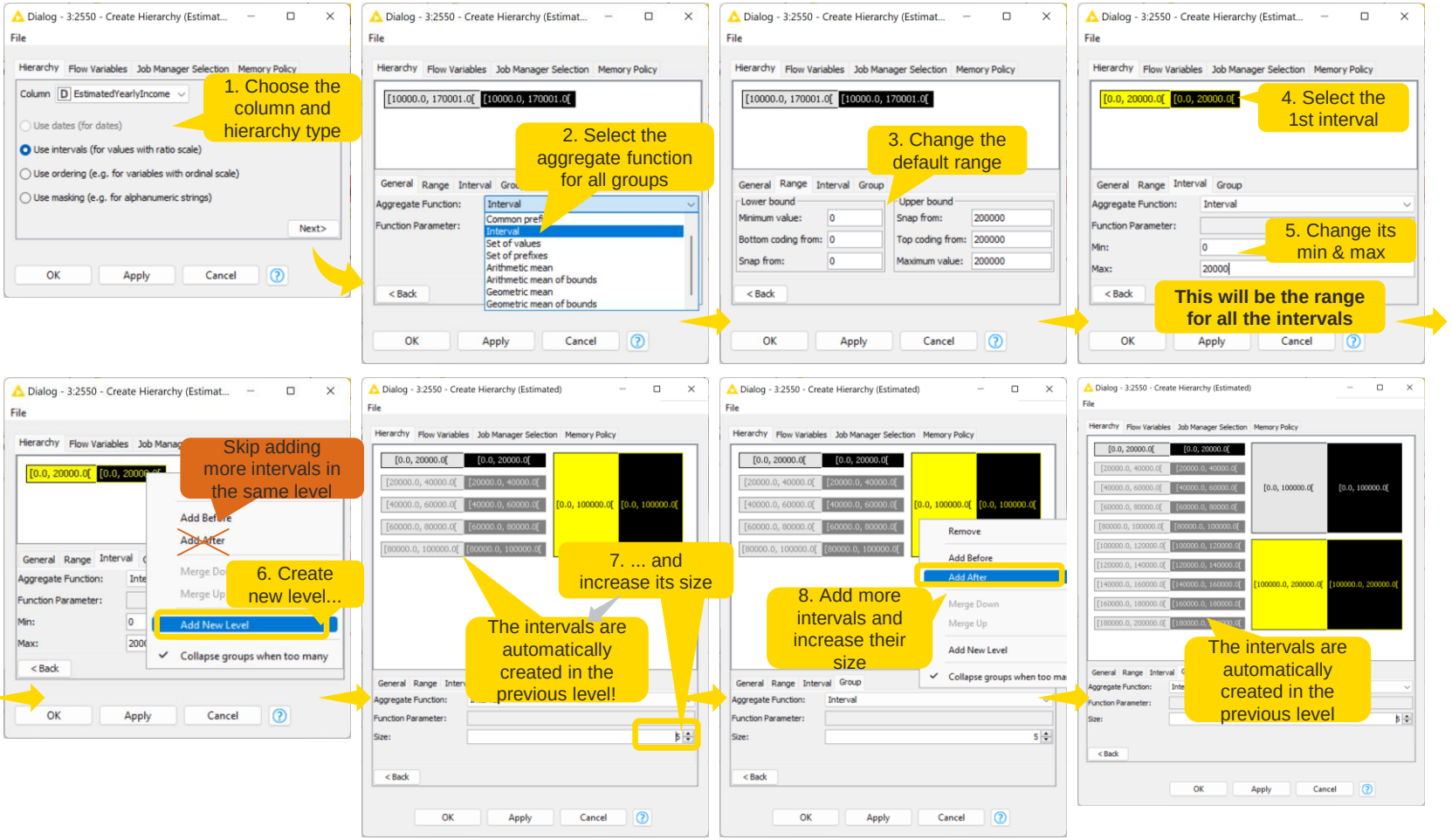
Create Hierarchy Node – Date and Time Data



Create Hierarchy Node – Categorical Data

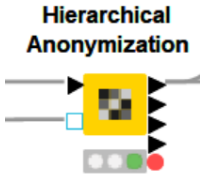


Create Hierarchy Node – Numeric Data



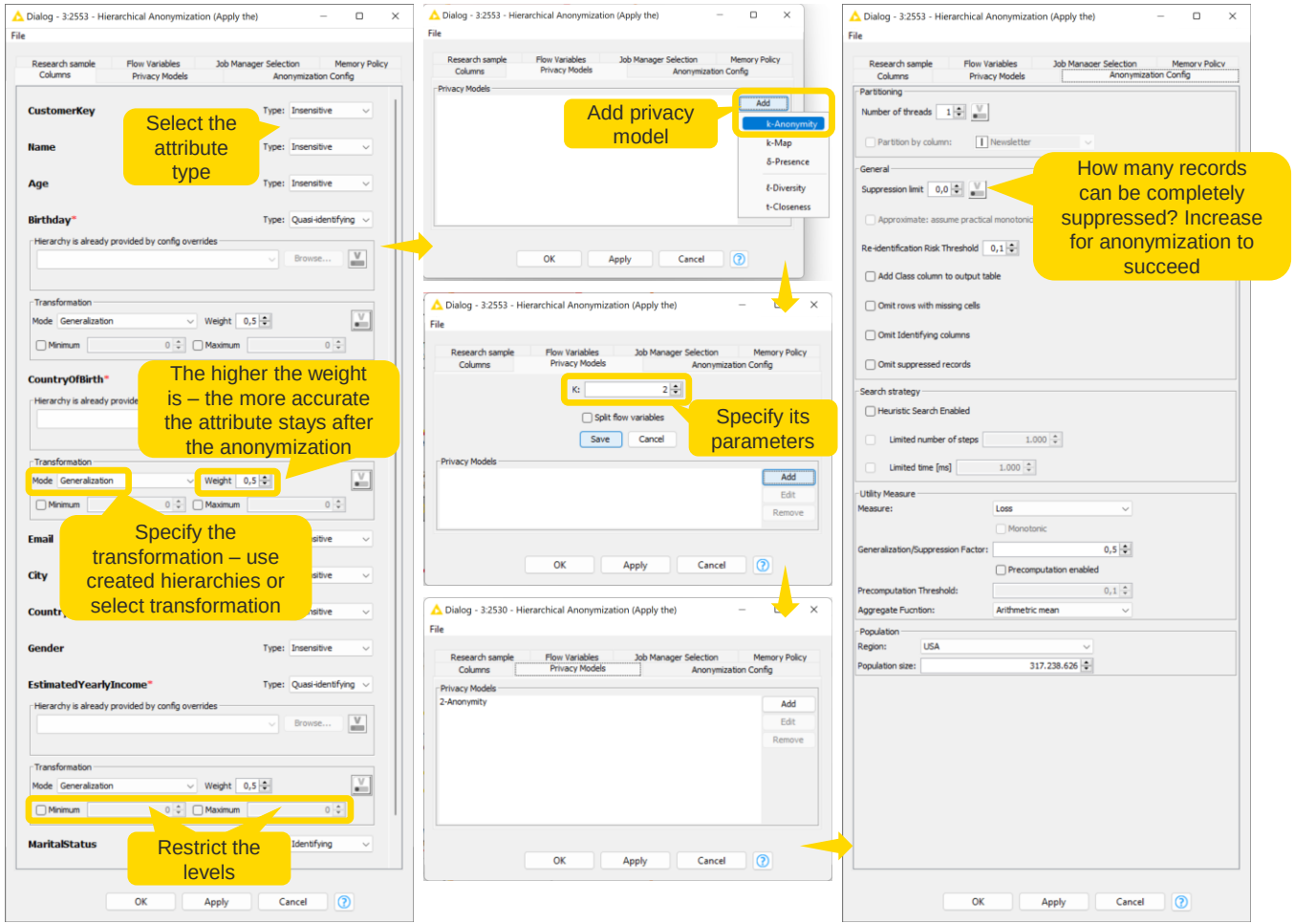
Generalization – Privacy Models

- Which level of hierarchy to use? – Let a **privacy model** find out
- Privacy model specifies conditions that the data set must satisfy to keep disclosure risk acceptable according to the user-specified parameters
- k-anonymity model
  - “A dataset is k-anonymous if each record cannot be distinguished from at least k-1 other records regarding the quasi-identifiers.”
  - “Each group of indistinguishable records in terms of quasi-identifiers forms a so-called equivalence class.”
  - E.g., in 2-anonymity model, each person should be indistinguishable from at least one other person
- Hierarchical Anonymization node from Redfield Privacy Nodes extension
  - Applies the privacy model to anonymize the data



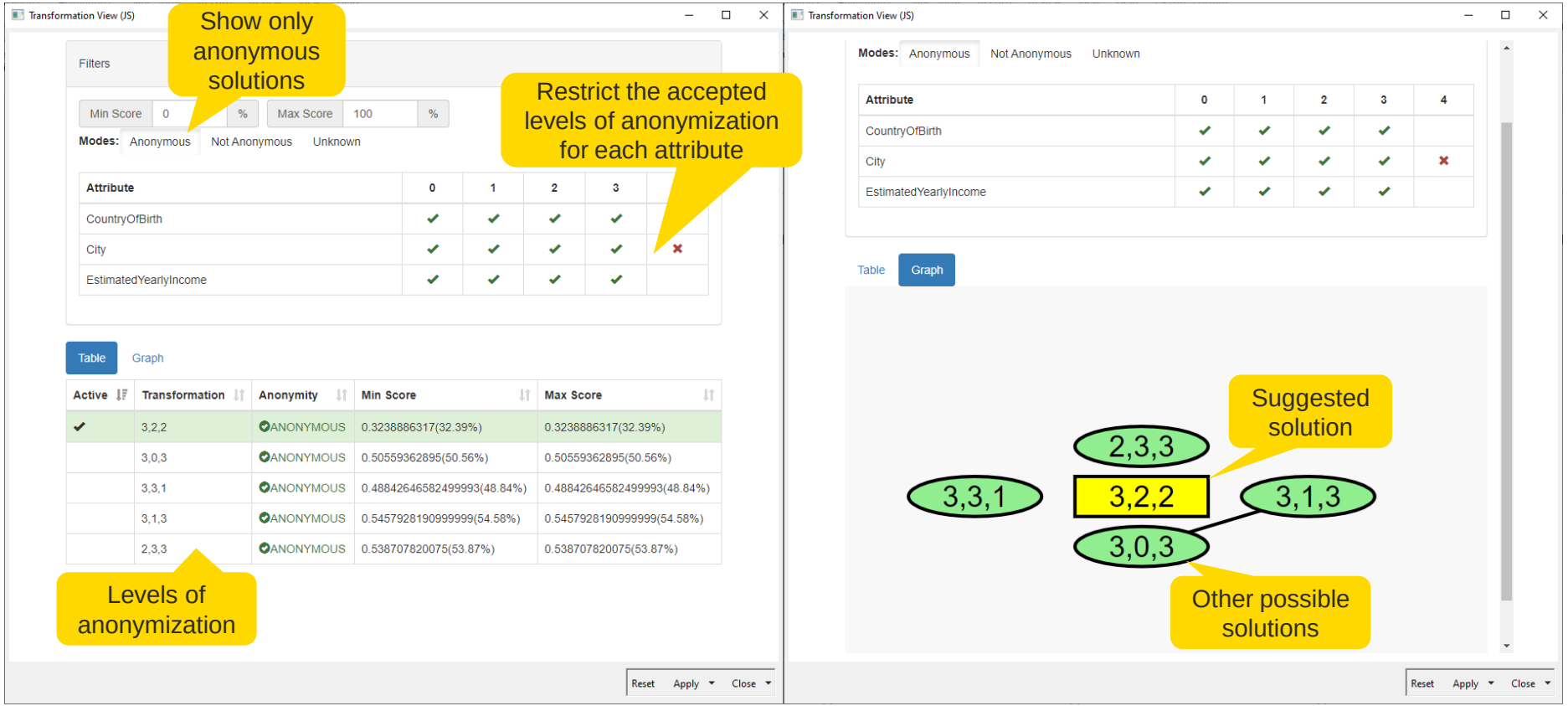
Sweeney, L. (2002). k-Anonymity: A Model for Protecting Privacy. Int. J. Uncertain. Fuzziness Knowl. Based Syst., 10, 557-570.

Hierarchical Anonymization Node



Hierarchical Anonymization Node

- Use automatically suggested levels of anonymization or control them

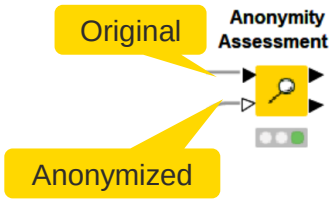


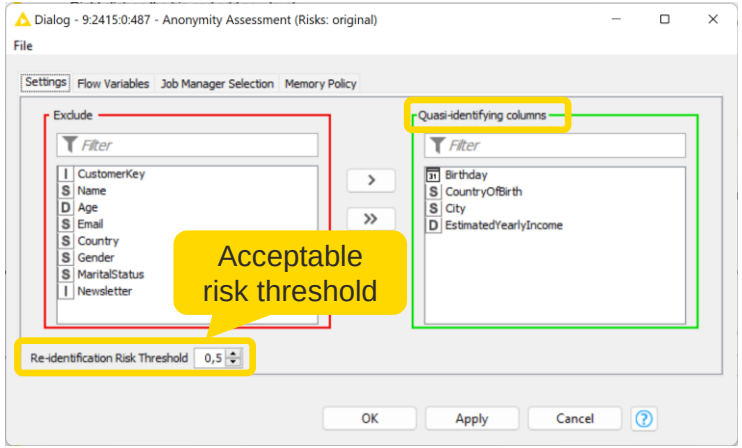
De-anonymization Risk Assessment

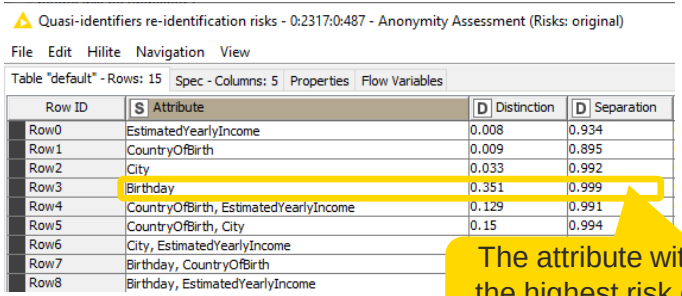
- Quasi-identifiers risks
  - Measure the distinction and separation of the quasi-identifiers and their combinations to find out which attributes have the biggest diversity
    - Separation defines the degree to which combinations of variables separate the records from each other
    - Distinction defines to which degree the variables make records distinct
- Attacker model risks
  - Estimates the probability of re-identification and success rate for it
    - Prosecutor: tries to identify a specific person in the dataset
    - Journalist: tries to identify any person in the dataset, to show that the dataset is compromised
    - Marketer: tries to identify as many people in the data set as possible
- Individual risk for each person to be re-identified is 1/k,
  - where k = 1 + the number of people with the same values in the quasi-identifying columns
    - e.g., the highest risk for the 2-anonymity model is 0.5

Anonymity Assessment Node

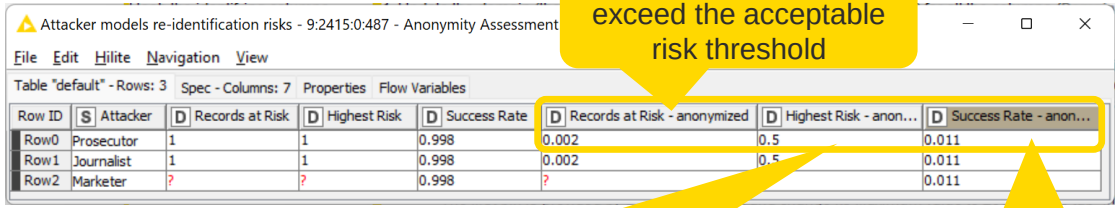
- Before anonymization
  - Provide only the original table to check which attributes or combinations are the most risky
  - Decide which attributes to anonymize
- After anonymization
  - Compare original data vs. anonymized data
  - Assess the de-anonymization risks







The attribute with the highest risk of identification

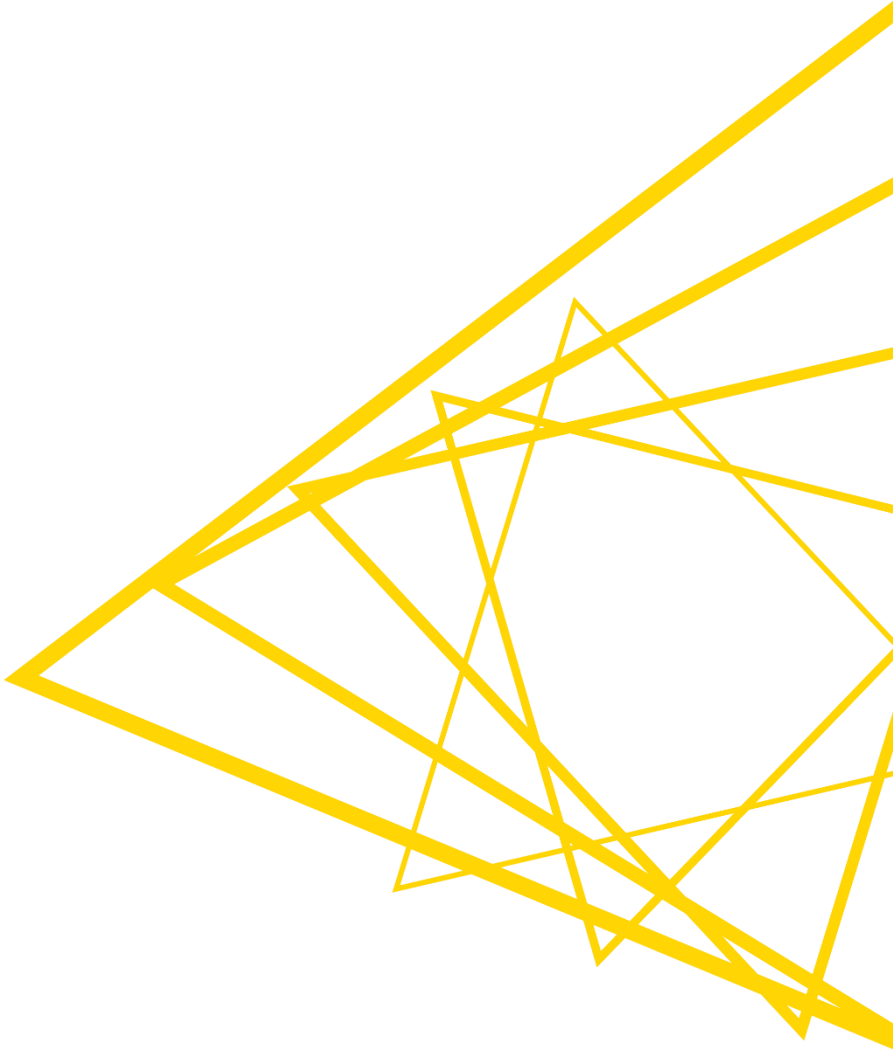


Ratio of records that exceed the acceptable risk threshold

For each person in the dataset there is at least (1 / highest risk) people with the same values in three quasi-identifying columns

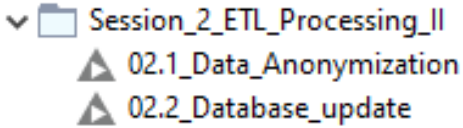
Weighted average risk of individual risks, avg(1/k)

Demo



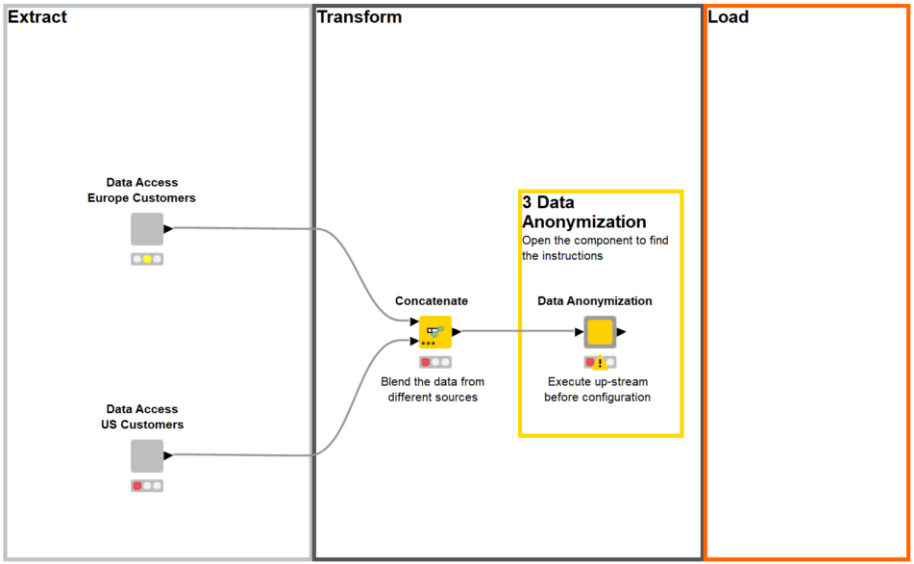
Exercises – Session 2

- Before starting the exercise **(skip if you performed these steps for session 1)**
  - Install local instance of PostgreSQL
  - Download the training workflows from the KNIME Community Hub
  - Install necessary extensions (open 00.1_Extensions_setup)
  - Execute workflow 00.2_Setup_PostgreSQL_Database
    - Use the credentials for your local instance of PostgreSQL

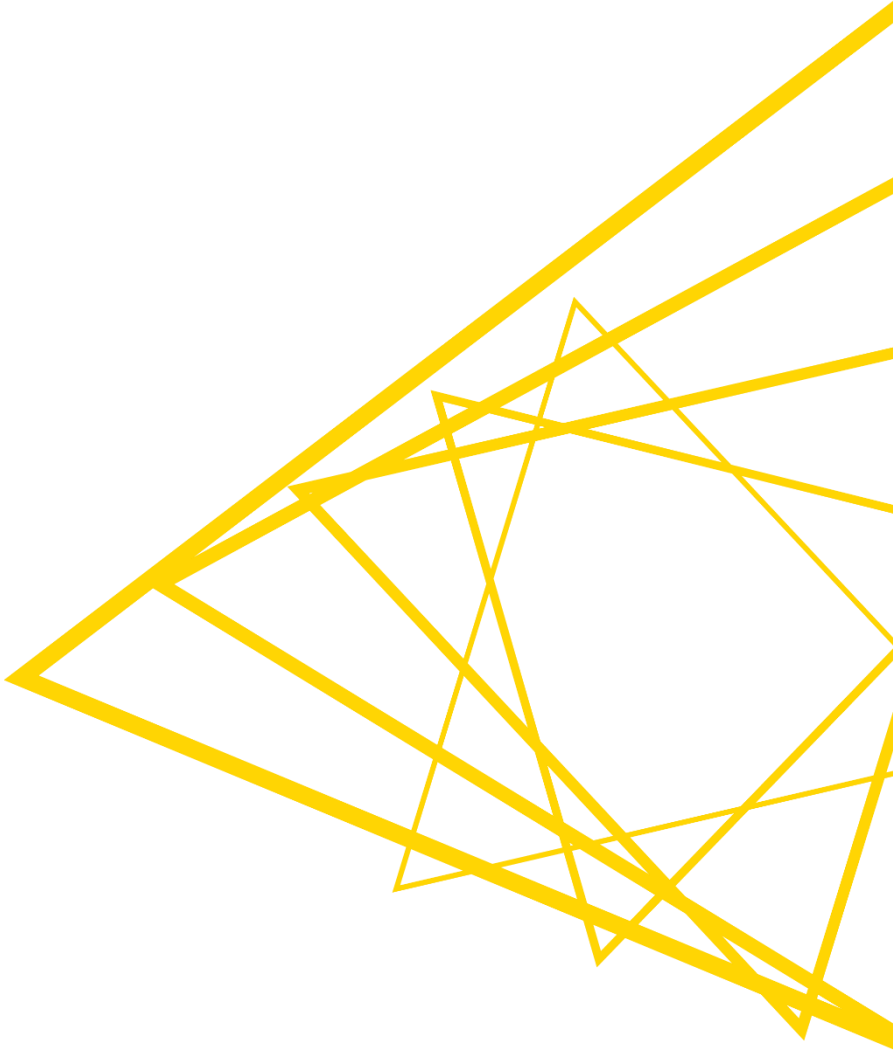


Exercise – 02.1_Data_Anonymization

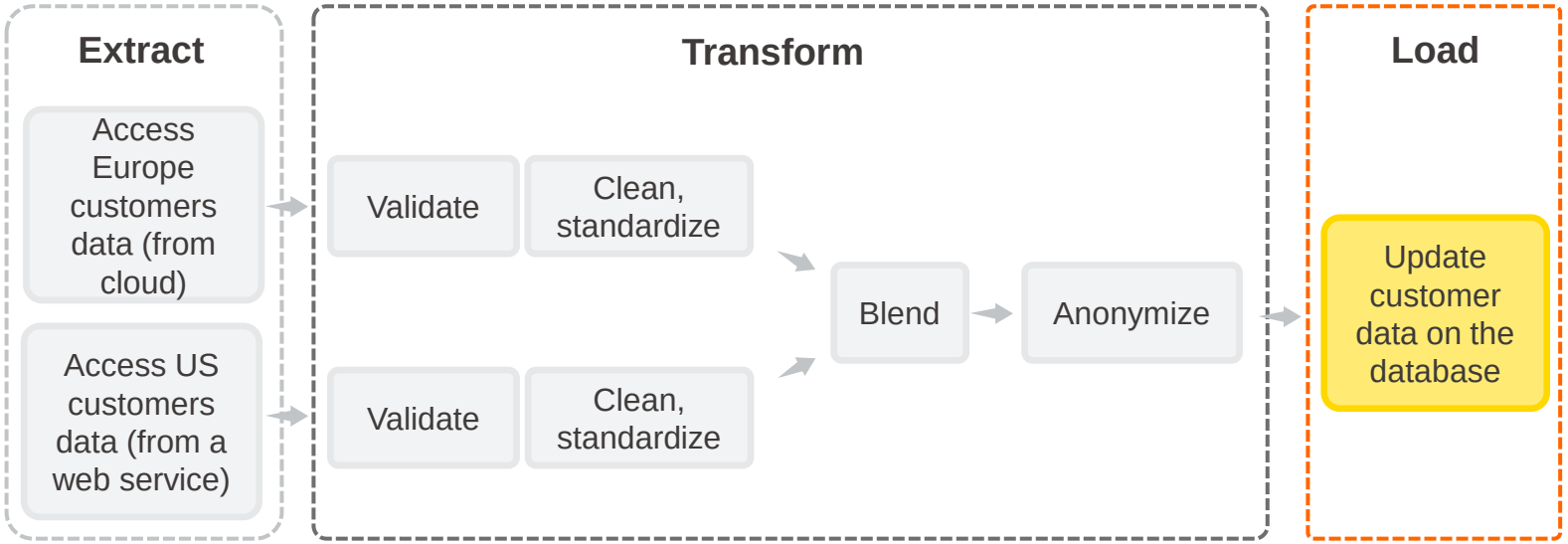
- This exercise is the third step to build application “ETL on Customers Data”
  - The solution to the previous exercises is already in the workflow
  - 3 Data Anonymization
  - Find detailed instructions in the workflow



Relational Databases

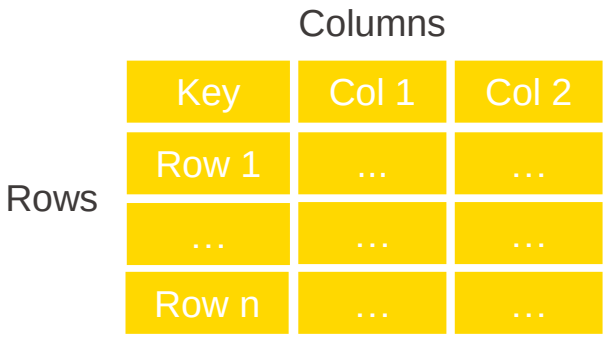


Relational Databases



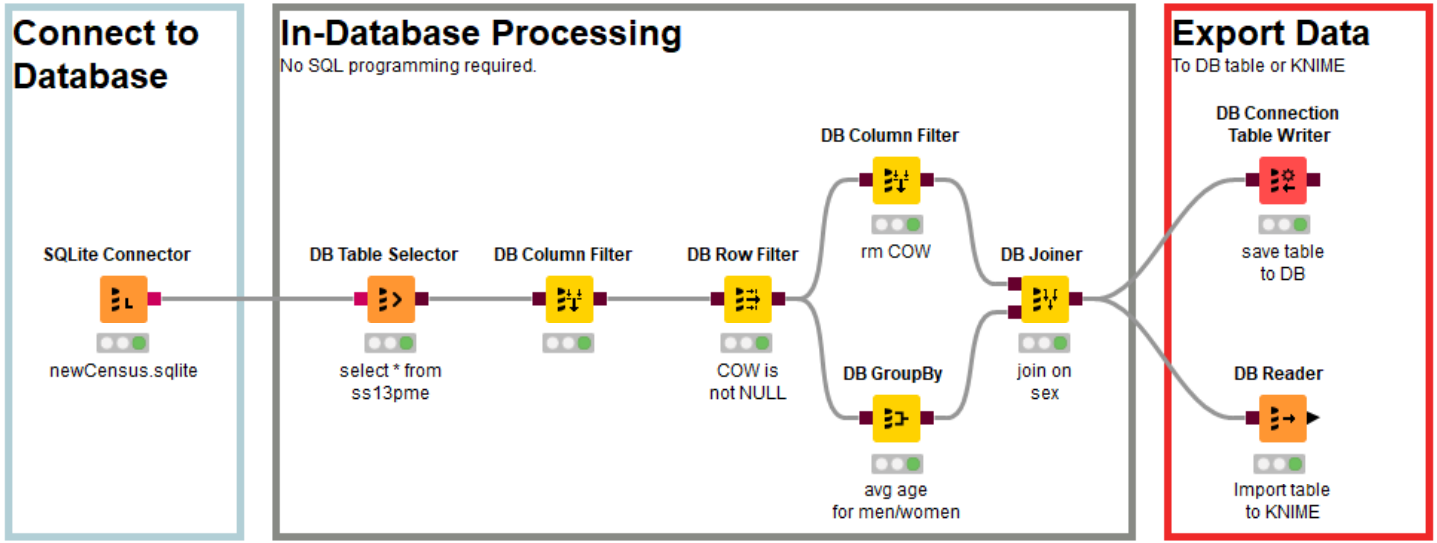
Relational Database

- Relational Database
  - Based on the relational model of data
  - Data are organized into tables – "relations“ – of columns and rows
  - Columns are attributes
  - Each row (record) contains a value for each attribute
  - Can be queried and maintained with SQL

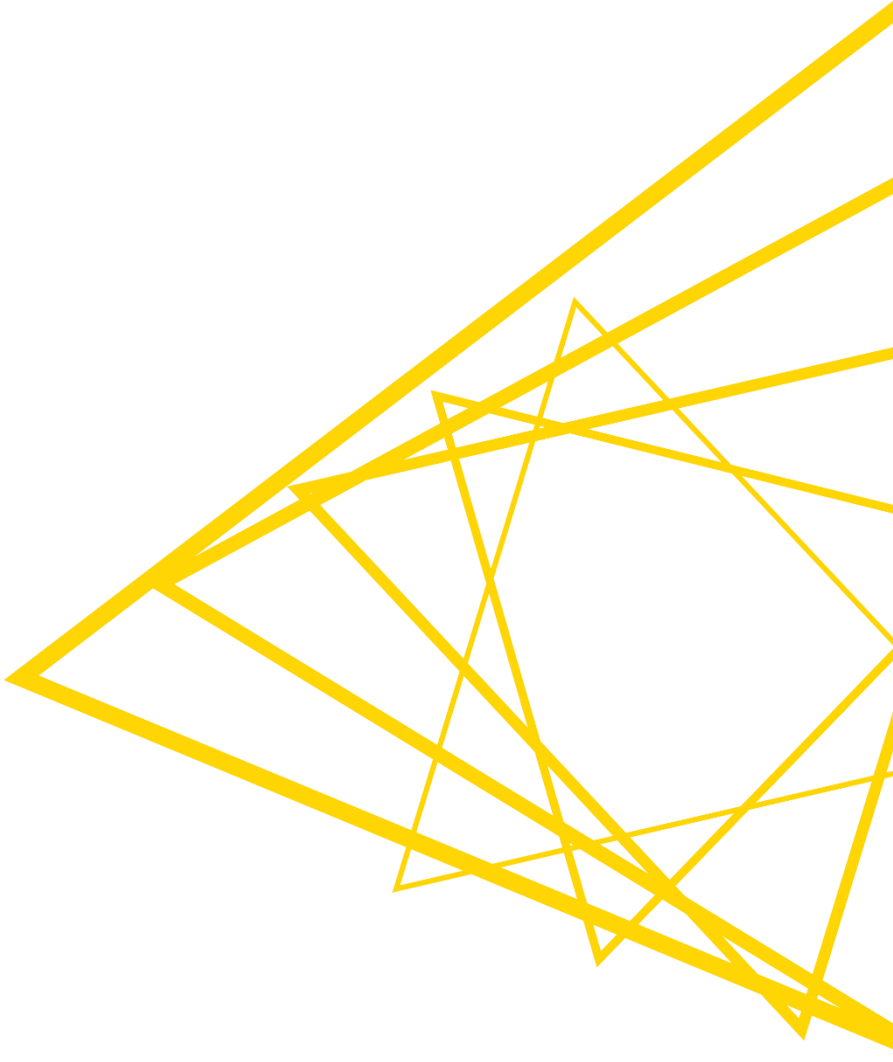


KNIME Database Extension

- Connect to all JDBC-compliant databases
- Visually assemble complex SQL statements (no SQL coding needed)
- Harness the power of your database within KNIME

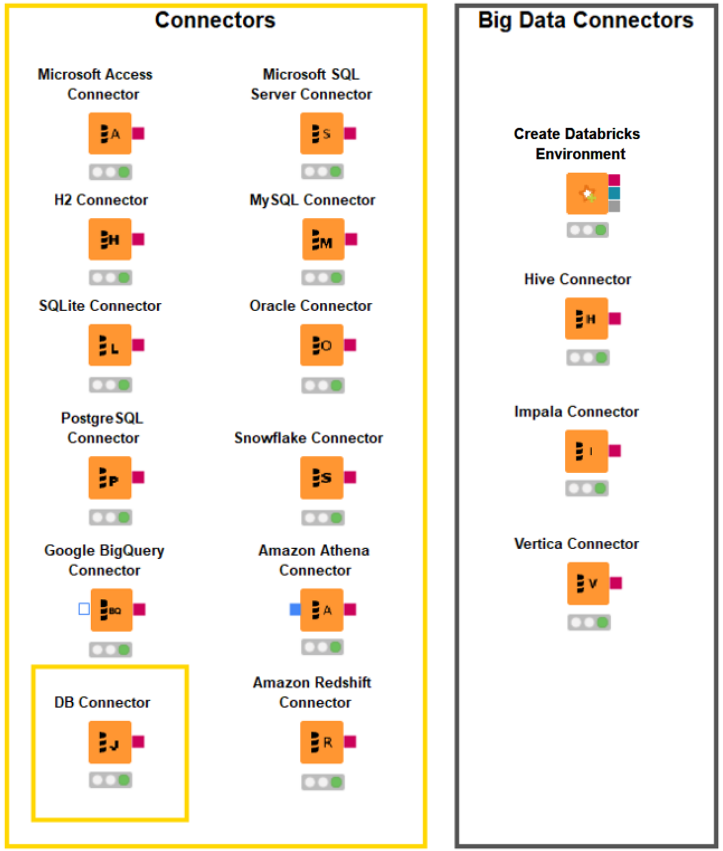


Connect to a Database



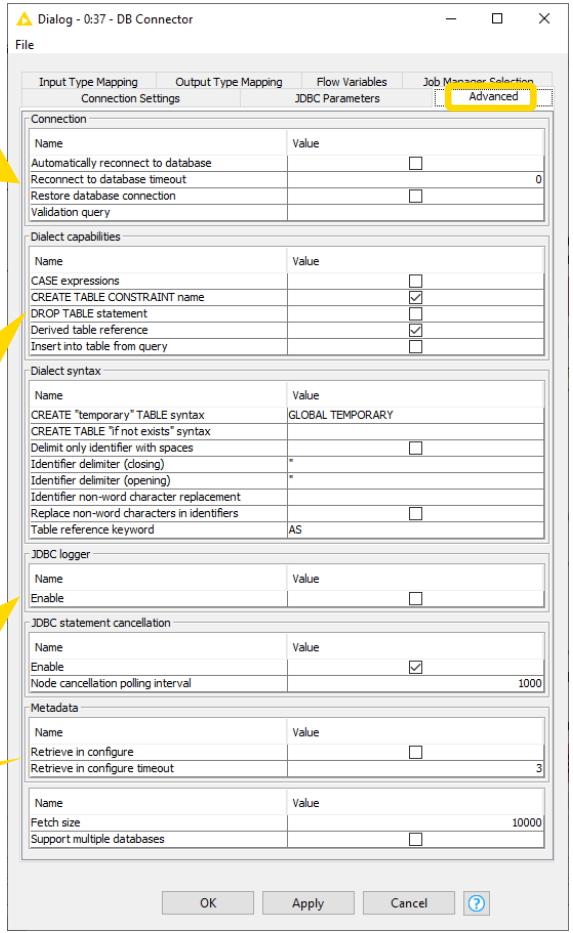
Database Connectors

- Dedicated nodes to connect to specific Databases
  - Necessary JDBC driver included
  - Easy to use
  - Import DB specific behavior/capability
- Hive and Impala connector part of the KNIME Big Data Connectors extension
- General Database Connector
  - Can connect to any JDBC source
  - Register new JDBC driver via File -> Preferences -> KNIME -> Databases



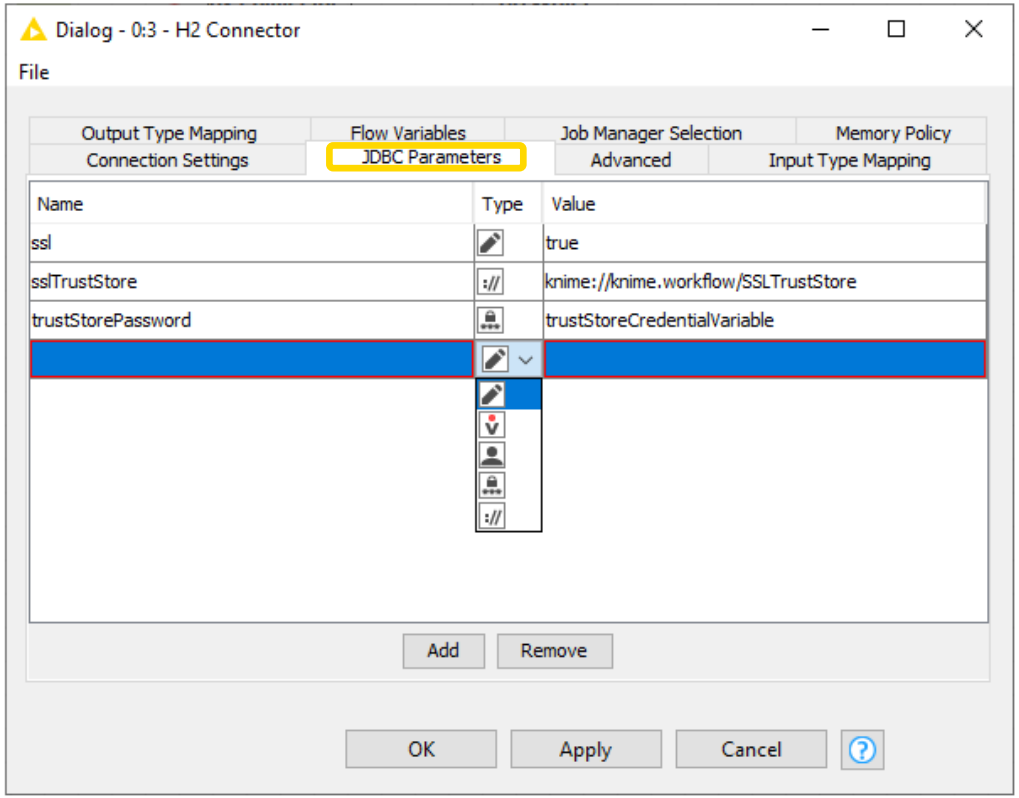
Advanced Database Options – Advanced Tab

- Define KNIME framework properties and interaction with the database
- Dedicated connectors show only a subset of options
- Check documentation for more details on each option

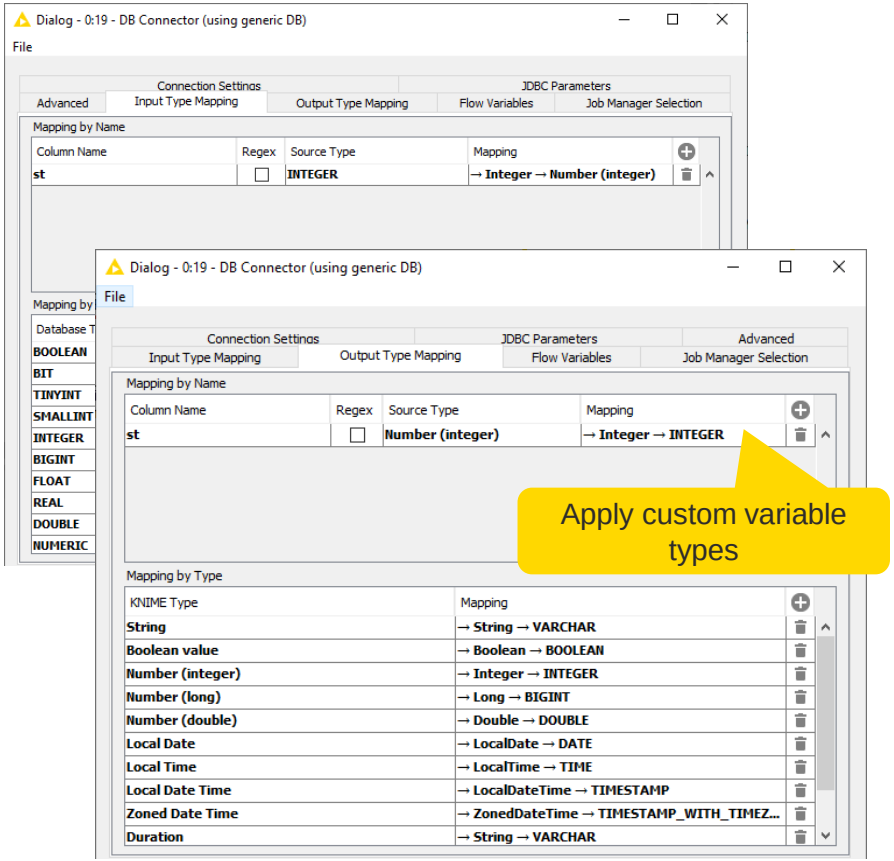


Advanced Database Options – JDBC Parameters

- Define custom JDBC driver connection parameters
  - constant,
  - variable,
  - credential user,
  - credential password,
  - KNIME URL (files)



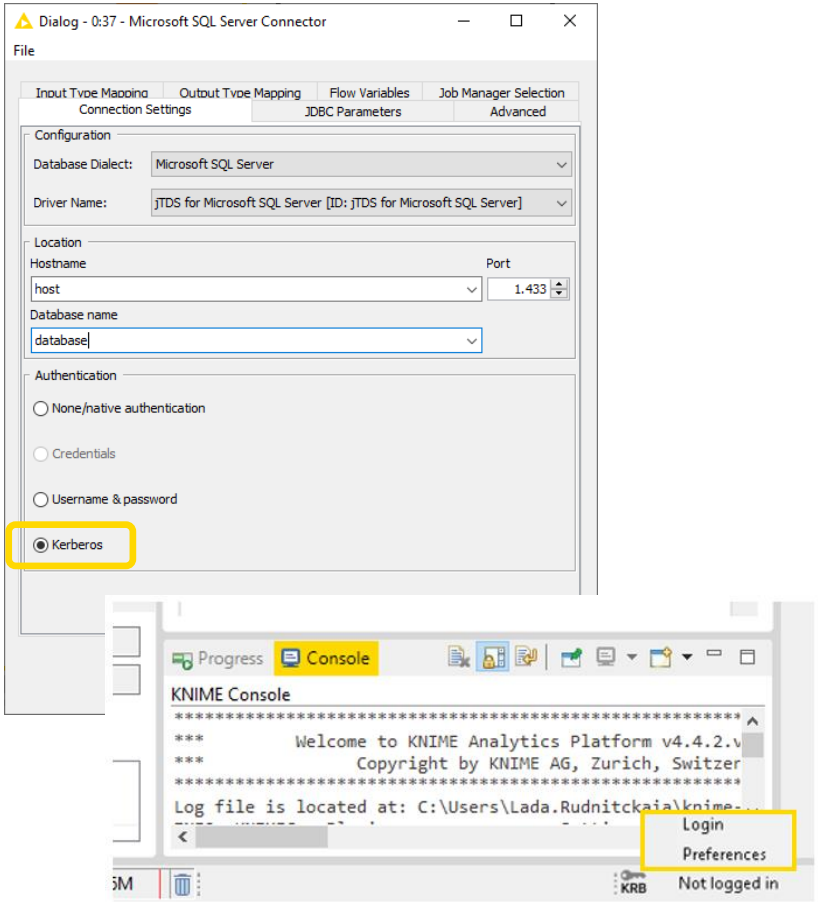
Advanced Database Options – Type Mapping



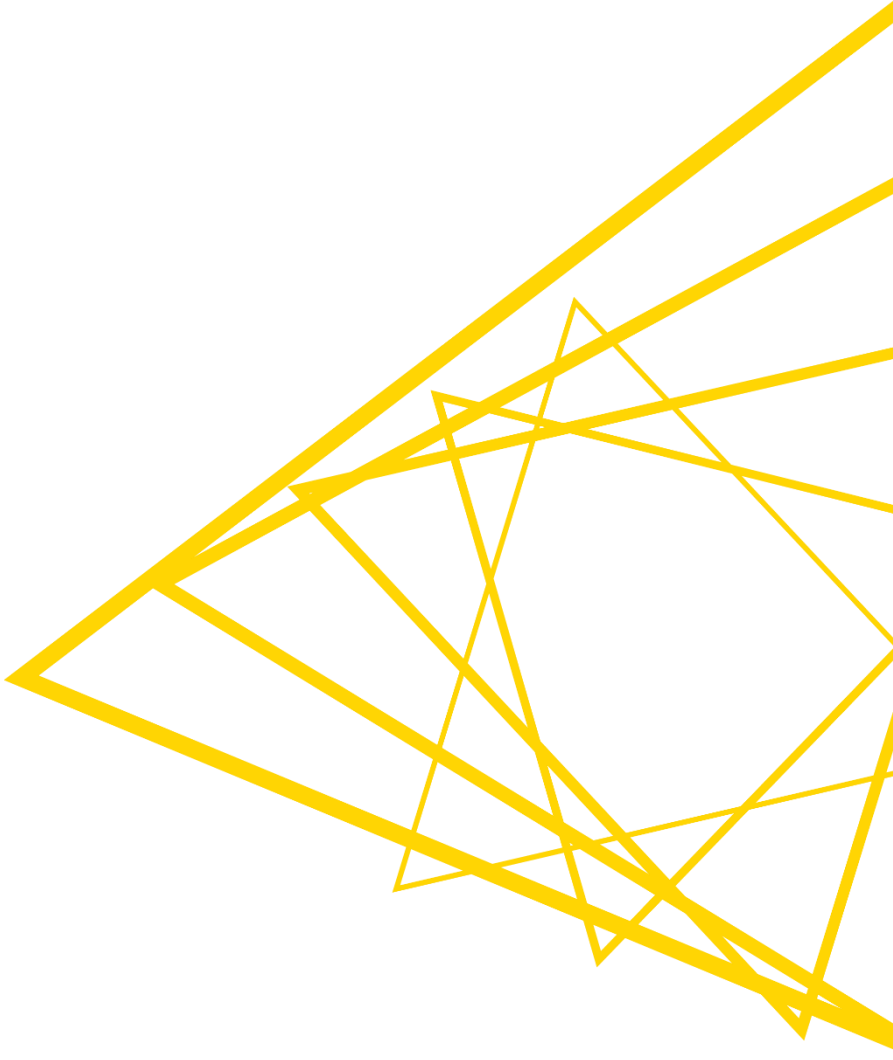
- KNIME will do its best to guess what type mappings are appropriate based on what it knows about your database
- Specify type mappings manually
  - By name, for individual fields – or groups of fields using RegEx
  - By type
- Two separate tabs to govern input (from a database) and output (from KNIME) type mappings
- DB Type Mapper Node

Authentication via Kerberos

- A network authentication protocol for distributed applications
  - Configure in File > Preferences > KNIME > Kerberos
  - Enable / disable Kerberos logging to get more information about Kerberos setup
  - Log in in the bottom right part of KNIME workbench and check the status in Preferences
  - In the connector node, select Kerberos as an authentication option and provide necessary parameters in the JDBC Parameters tab
- Provided for connectors that support it
  - DB Connector, Microsoft SQL Server, Oracle, PostgreSQL, Hive, Impala, Vertica, HDFS, SMB, Create Spark Context (Livy), REST nodes, SAP Reader (Theobald Software)
- Full documentation available

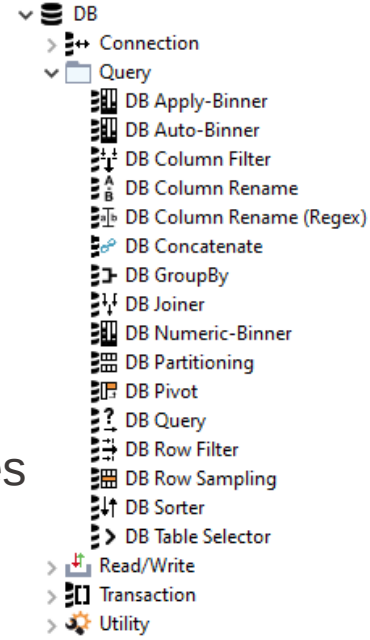


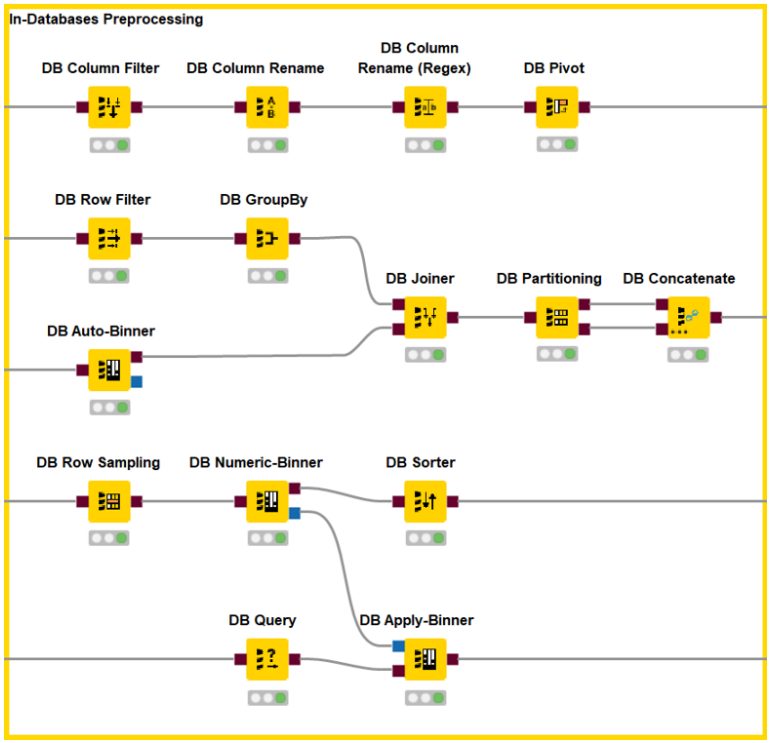
In-Database Processing



Query Nodes

- Various manipulations
  - Filter rows and columns
  - Join and concatenate tables
  - Extract samples
  - Bin numeric columns
  - Sort
  - Aggregate
  - Write your own query
- Configuration is similar to KNIME Manipulation nodes (in most cases)
- No SQL coding
- The nodes construct and output a SQL query

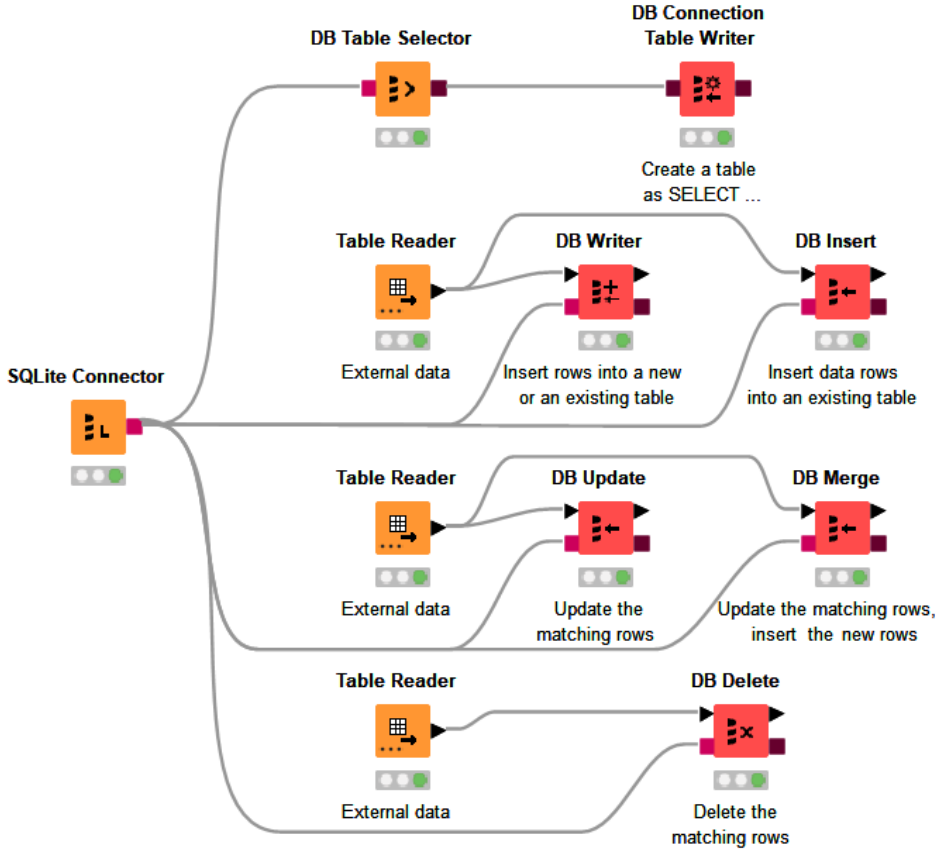




Write/Load Data into a Database

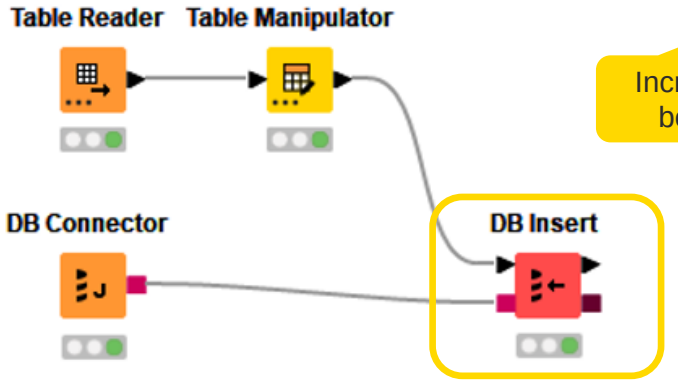
Database Writing Nodes

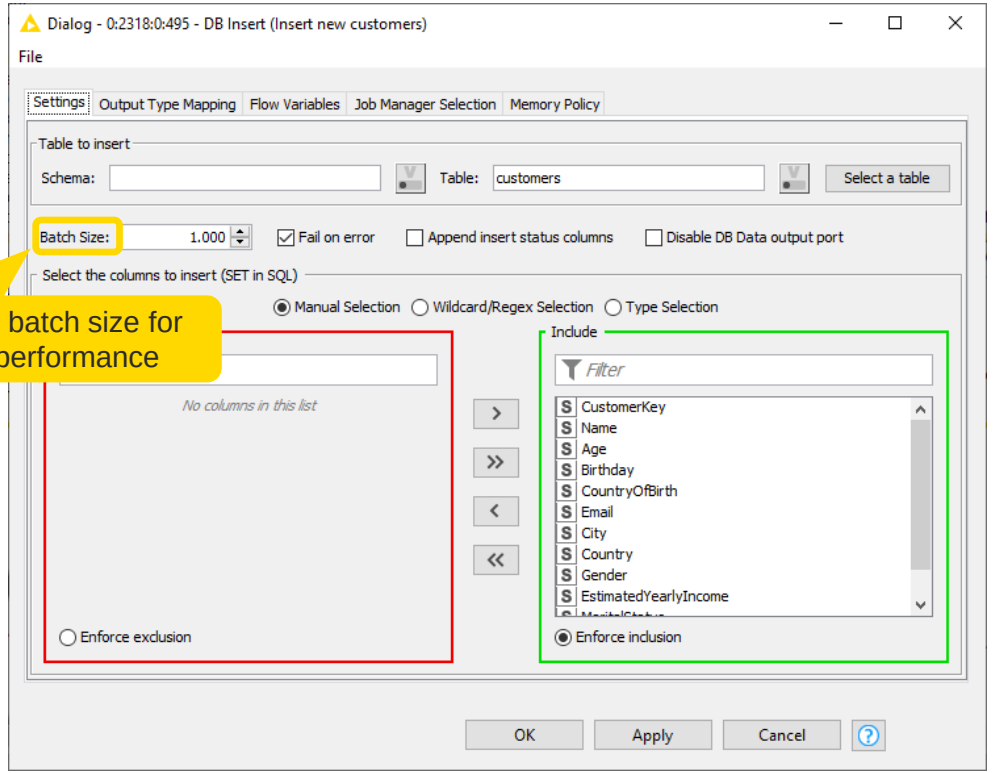
- Create table directly in the datab
- Insert/append data
- Update values in table
- Delete rows from table



DB Insert Node

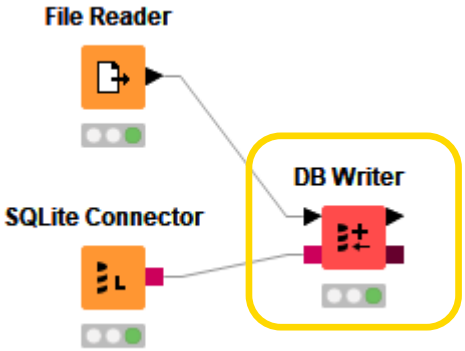
- Inserts data rows into an existing database table
  - Column names need to match exactly

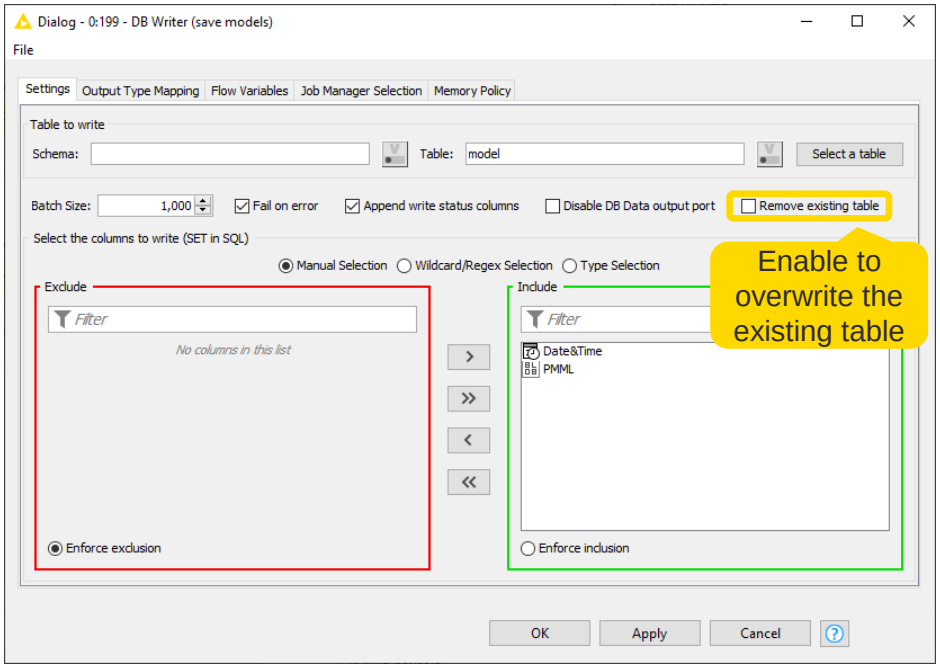




DB Writer Node

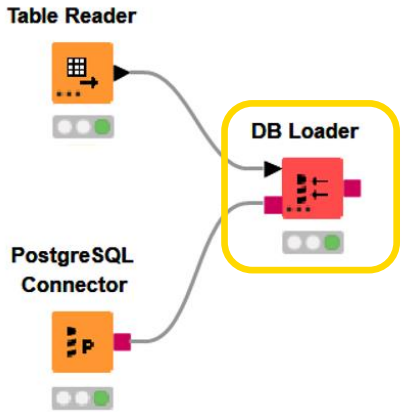
- If the table exists
  - Inserts data rows into a database table
    - Column names need to match exactly
  - Or overwrites the existing table
- If the table doesn't exist
  - Creates it according to the input table spec
  - To control new table's properties, use DB Table Creator and DB Insert nodes instead

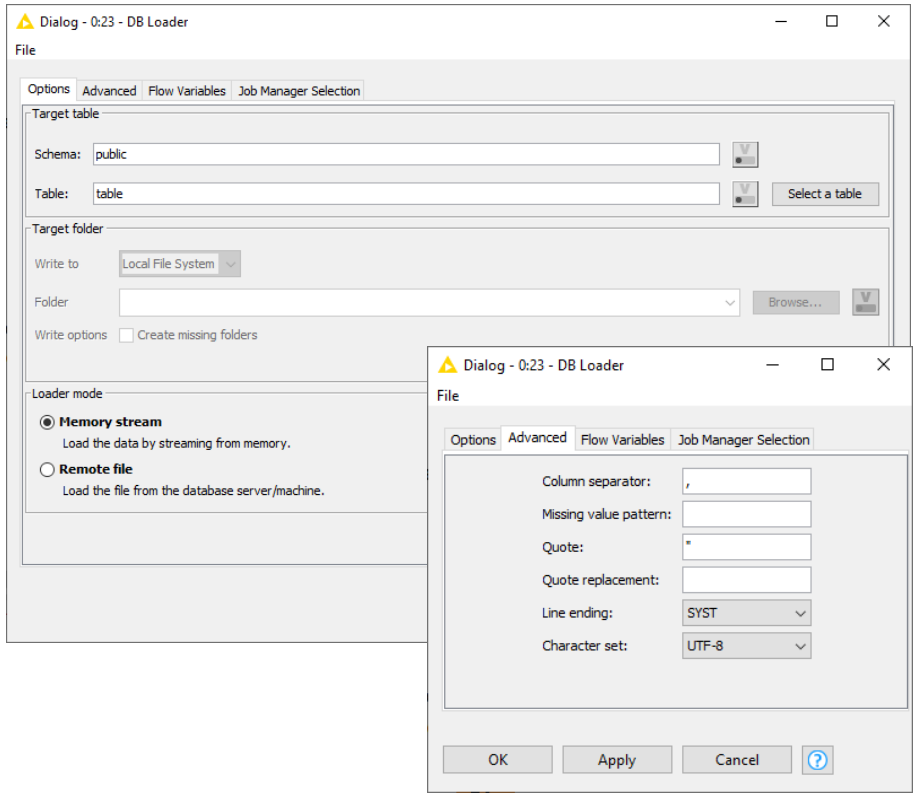




DB Loader Node

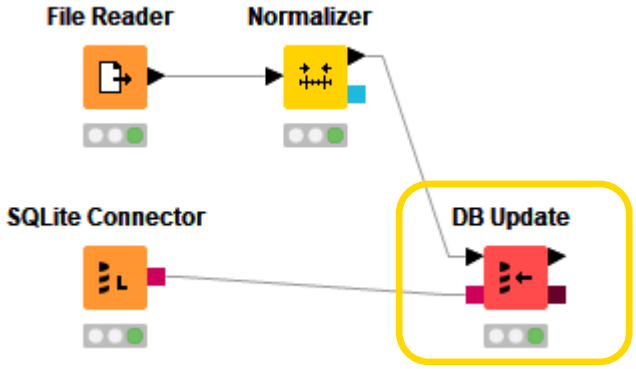
- Inserts data rows into an existing database table
  - Uses **fast** bulk loading for large amounts of data
  - Column names and order need to match exactly
  - Doesn't check the column type compatibility or the values itself
    - Can lead to a corrupt data table
  - Supported by the limited number of databases

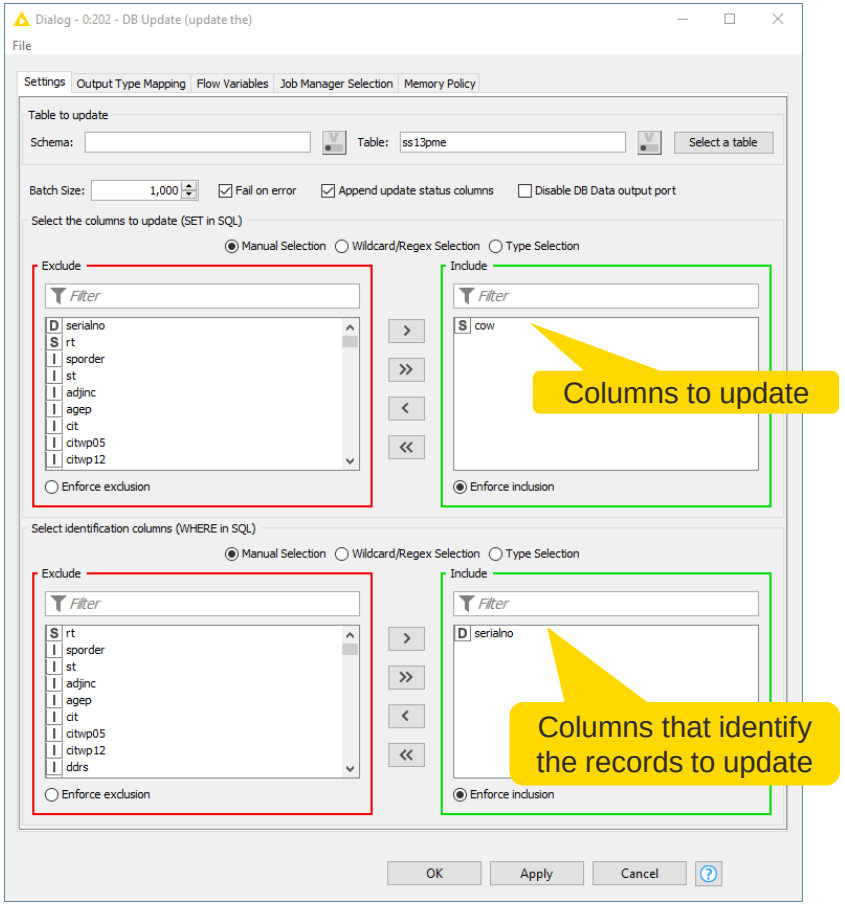




DB Update Node

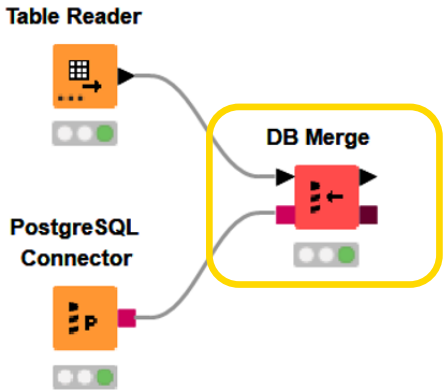
- Updates the records in the existing database table that match the update criteria
  - Column names need to match exactly

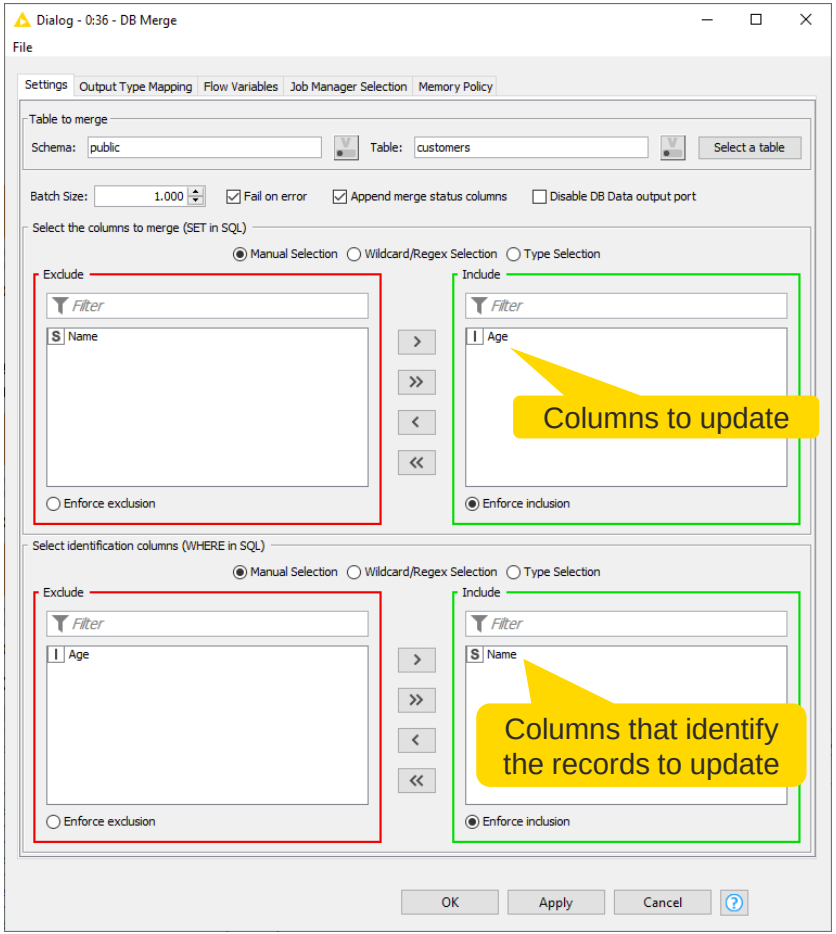




DB Merge Node

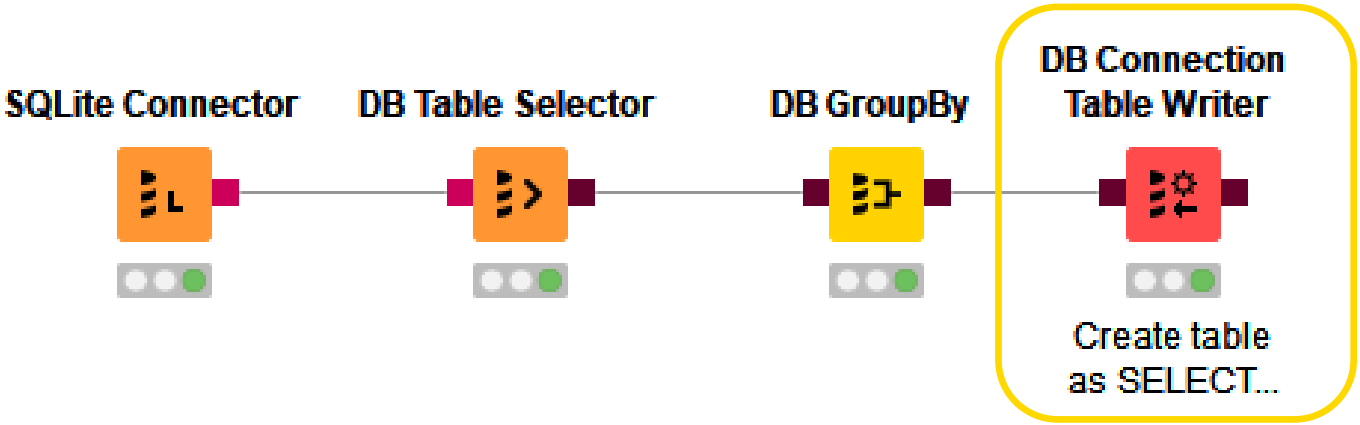
- Updates the records in the existing database table that match the update criteria
- Inserts new records if there are no matching rows
  - Column names need to match exactly





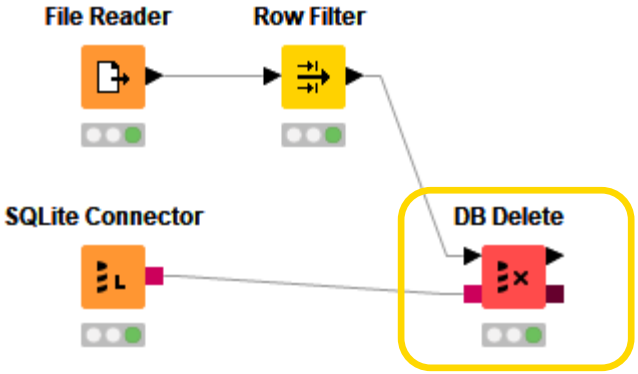
DB Connection Table Writer Node

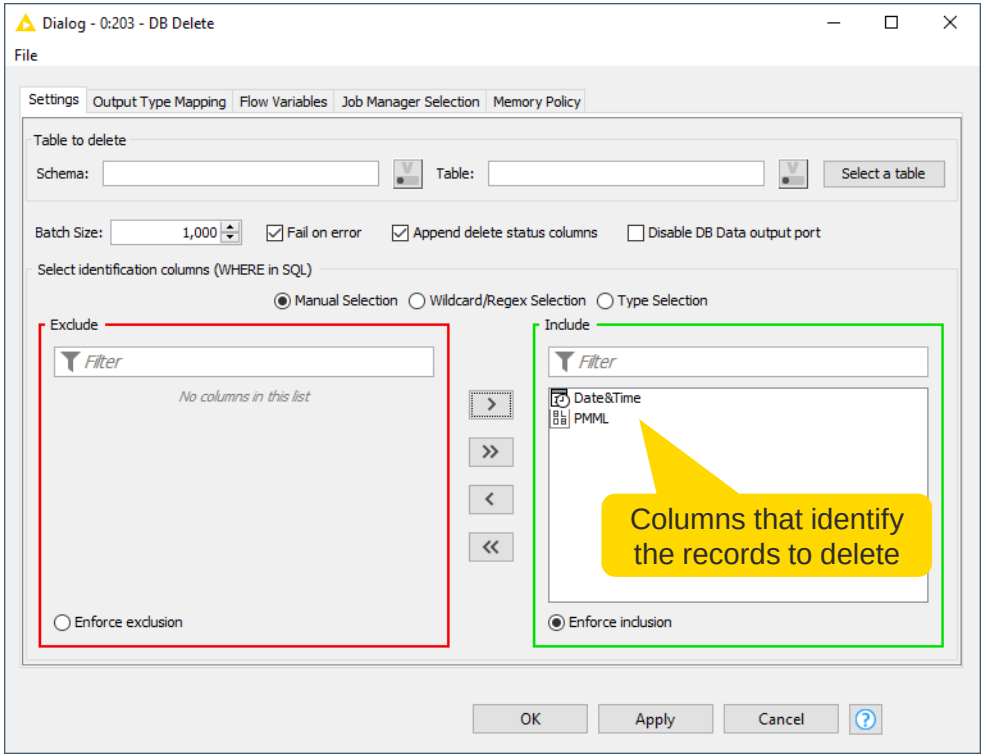
- Creates a new database table **based on the input SQL query**



DB Delete Node

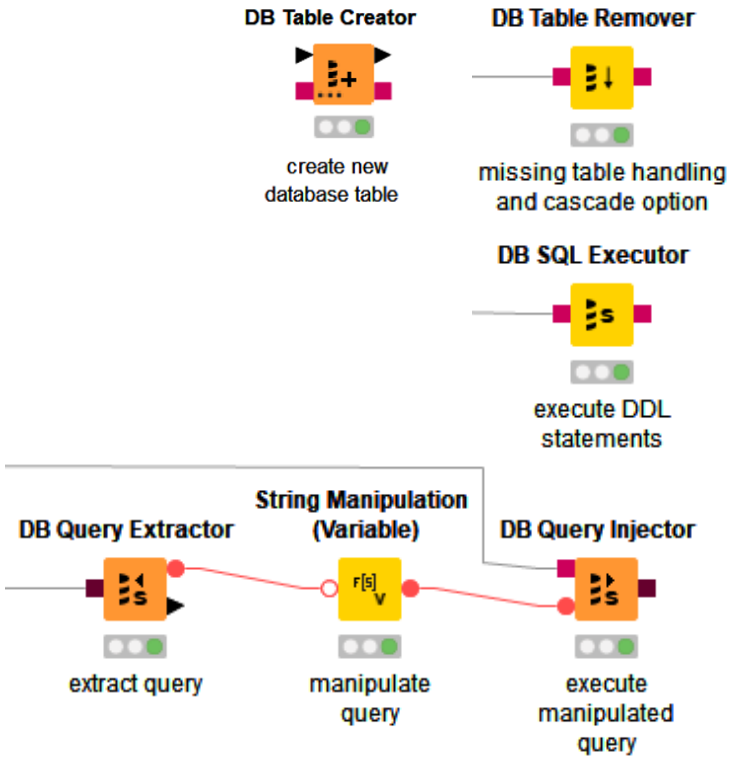
- Deletes the records that match the values of the selected columns in the existing database table
  - Column names need to match exactly





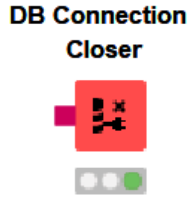
Utility

- Create new table
- Drop table
  - missing table handling
  - cascade option
- Execute any SQL statement e.g. DDL
- Manipulate existing queries



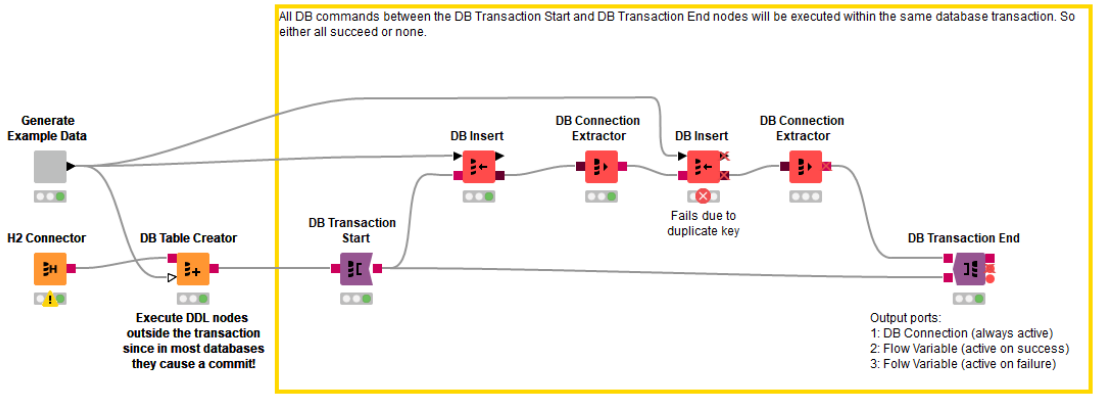
DB Connection

- The DB Session lifecycle is managed by the Connector nodes
  - Executing a Connector node will create a DB Session
  - Resetting the node or closing the workflow will destroy the corresponding DB Session
- DB Connection Closer
  - Closes the input database connection
  - Always use at the end of the DB processing
    - Free up DB resources as soon as possible
    - No need to wait till the end of the workflow to close the DB connection if it's used in the beginning of the workflow only
    - On the Business Hub it might take some time to really close a workflow
- DB Connection Extractor
  - Extracts the output database connection from the input database data connection
    - Use when a node that requires a DB Session input port executes after a node that outputs a DB Data port
    - E.g., when modeling the database transactions



Transaction Support

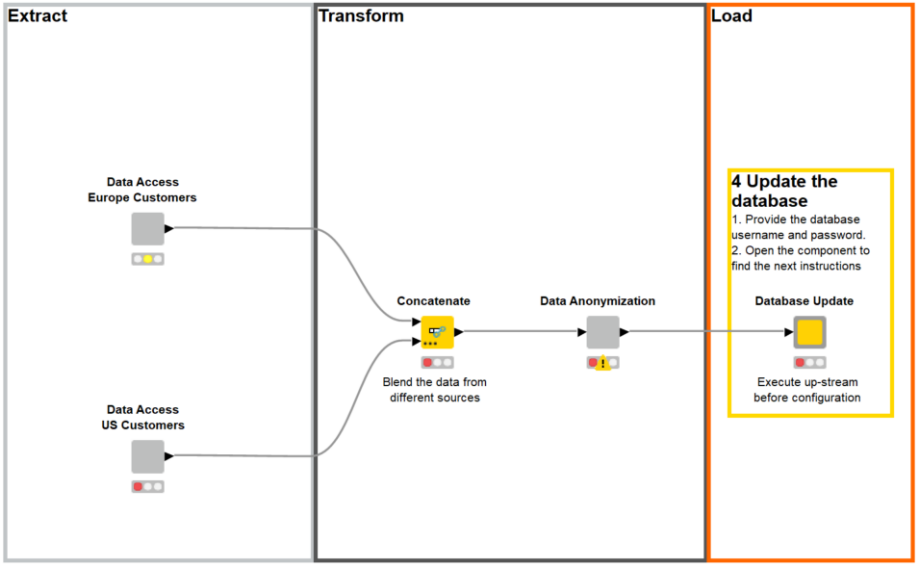
- DB Transaction Start/End
  - Group several database data manipulation operations into a single unit of work – transaction
  - The transaction either completes entirely or not at all
    - If successful, the transaction ends with a commit that makes all changes visible to other users
    - If not, the transaction ends with a rollback returning the database to the state at the beginning of the transaction
  - Uses the default isolation level of the connected database



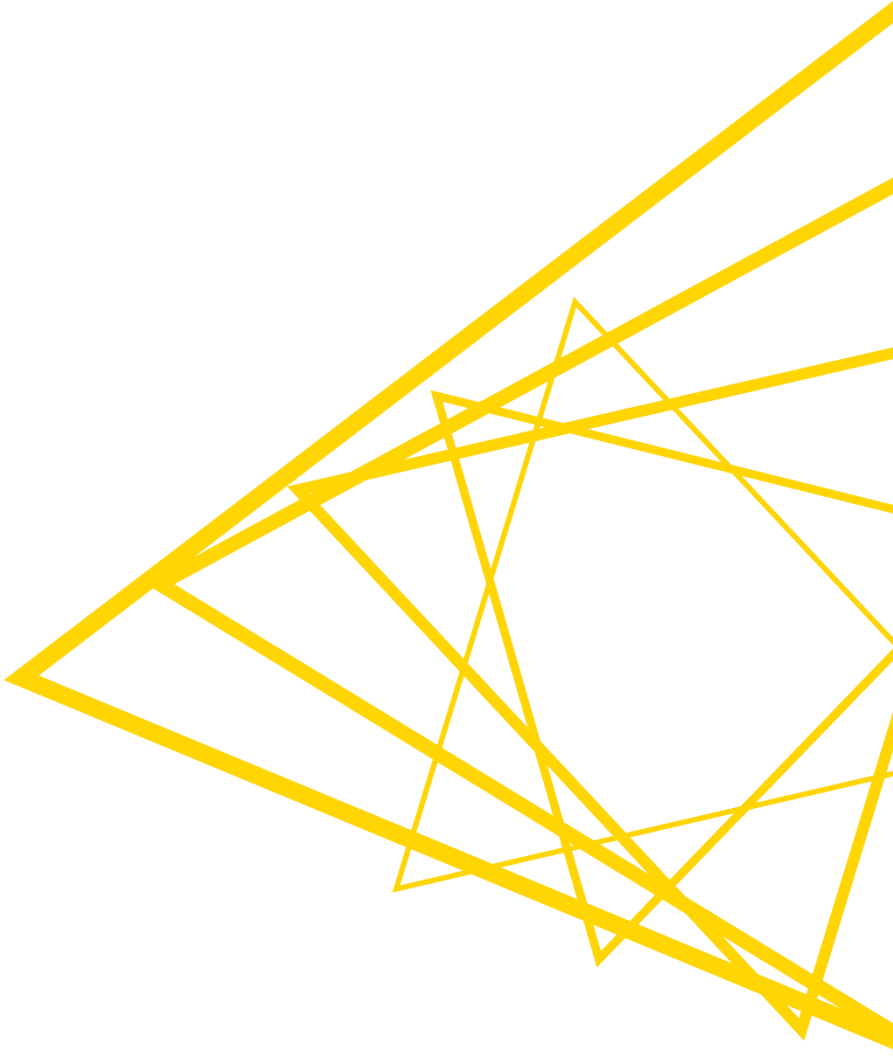
Workflow is available on the KNIME Community Hub:
https://kni.me/w/kWP1OhaY4DXK444n

Exercise – 02.2_Database_update

- This exercise is the fourth step to build application “ETL on Customers Data”
  - The solution to the previous exercises is already in the workflow
  - 4 Update the database
    - Use the credentials for your local instance of PostgreSQL
  - Find detailed instructions in the workflow



NoSQL Databases



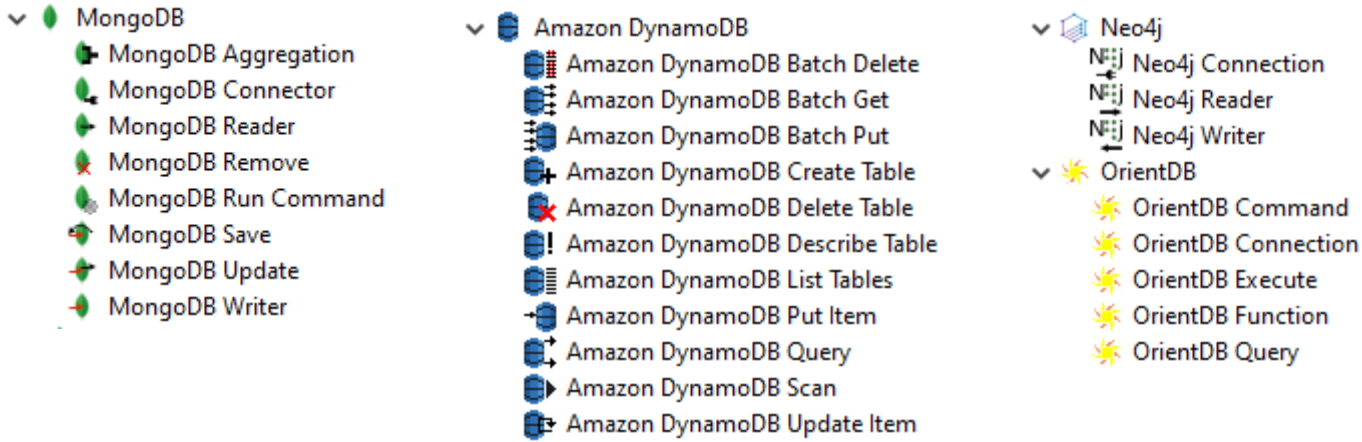
NoSQL Databases

- Non-SQL or non-relational database design
- Store and retrieve data other than in tabular form
- Different data models (structures)
  - key-value, document, column, graph, object, multimodel, etc.
- Used in big data and real-time web applications
  - Flexible data model (No ETL, no need to design schema first)
  - Quick insights
- Example: key-value NoSQL database
  - Keys identify data records uniquely
  - Each record can have different fields of different types
  - Queries are expressed in terms of keys
  - Memory efficient – gaps aren't stored

Key	Value
k1	A, B, C
k2	A, B
k3	B, D
k4	A, 4, 2

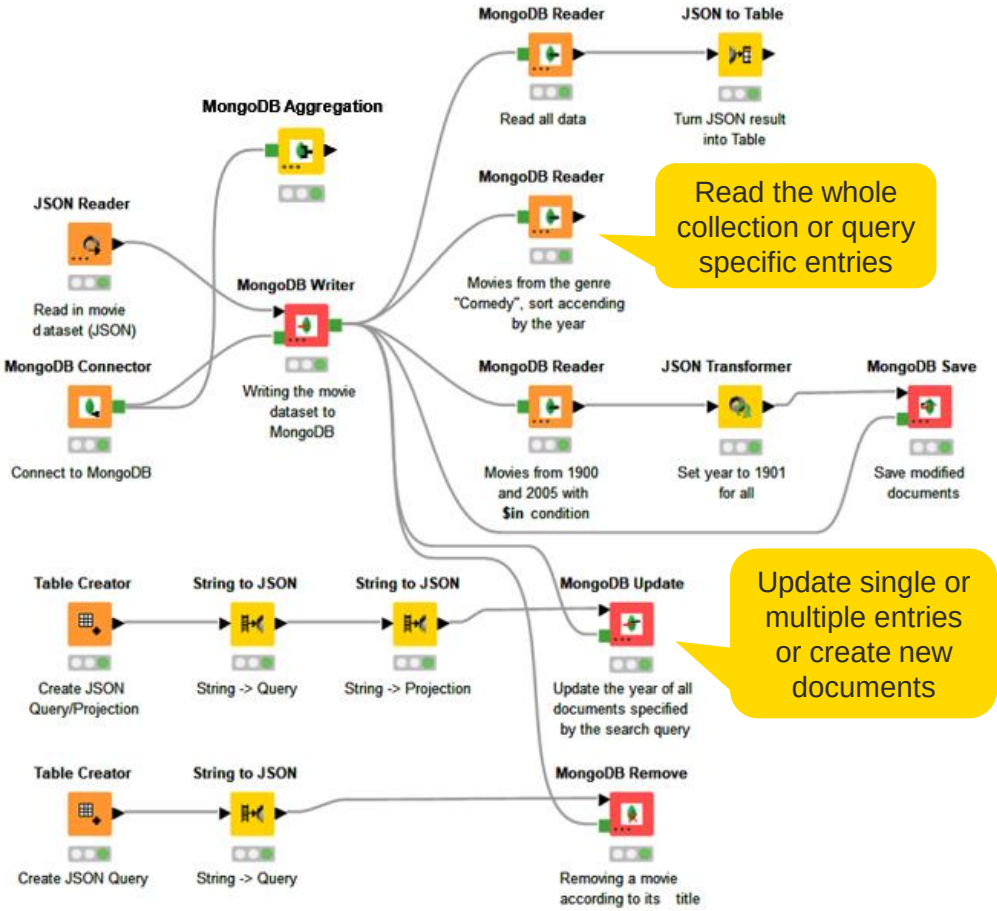
NoSQL Databases in KNIME Analytics Platform

- Supported NoSQL databases:
  - MongoDB (document)
  - DynamoDB (key-value, document)
  - Neo4j (graph)
  - OrientDB (multi-model: graph, document, key/value, and object models)



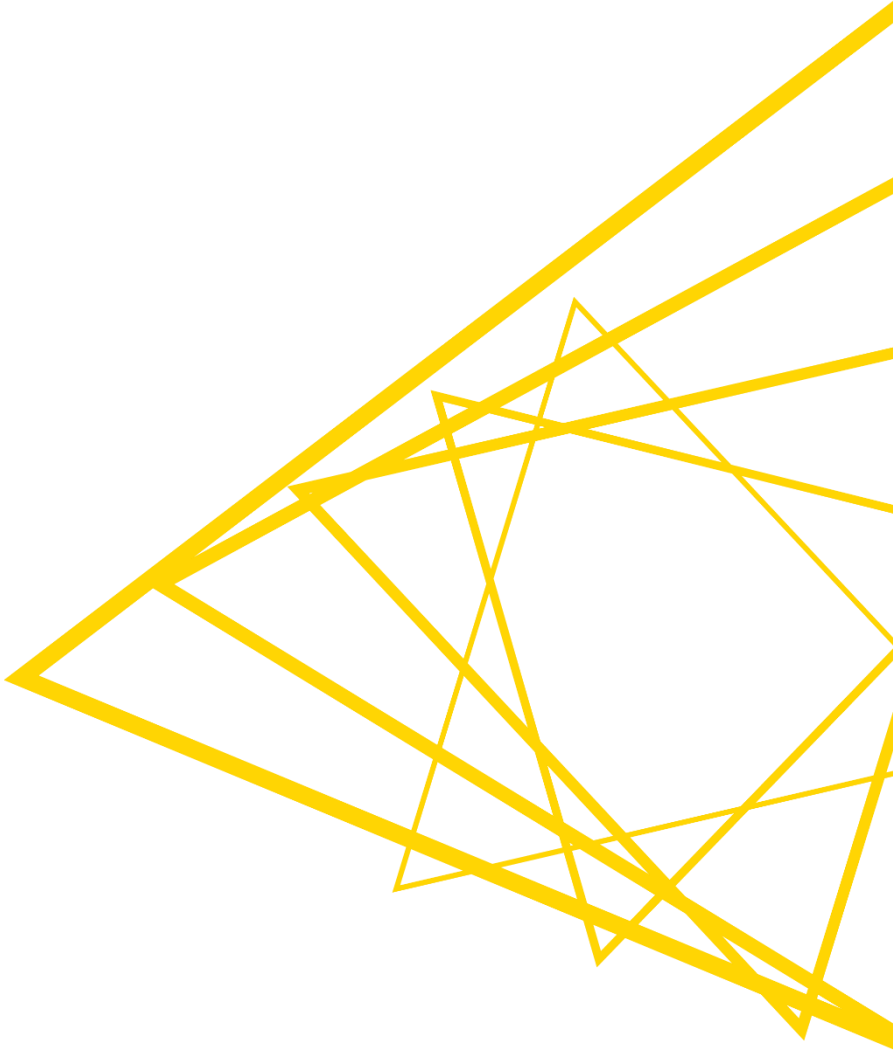
MongoDB in KNIME Analytics Platform

- Dedicated nodes in KNIME MongoDB Integration
  - Connect
  - Read all JSON data
  - Query specific entries
  - Write new dataset to MongoDB
  - Update, insert, remove specific entries
  - Execute database management controls in MongoDB from within KNIME
- Querying
  - Selecting the database and collection
  - Specify query, projection, & sorting
  - Aggregate within MongoDB



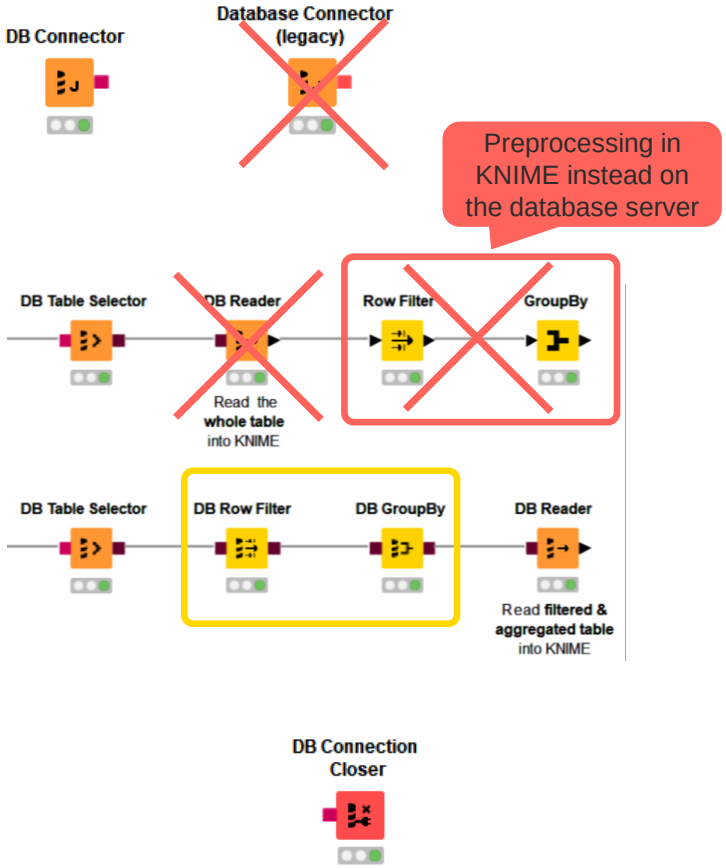
Best Practices: ETL, Databases

Efficiency

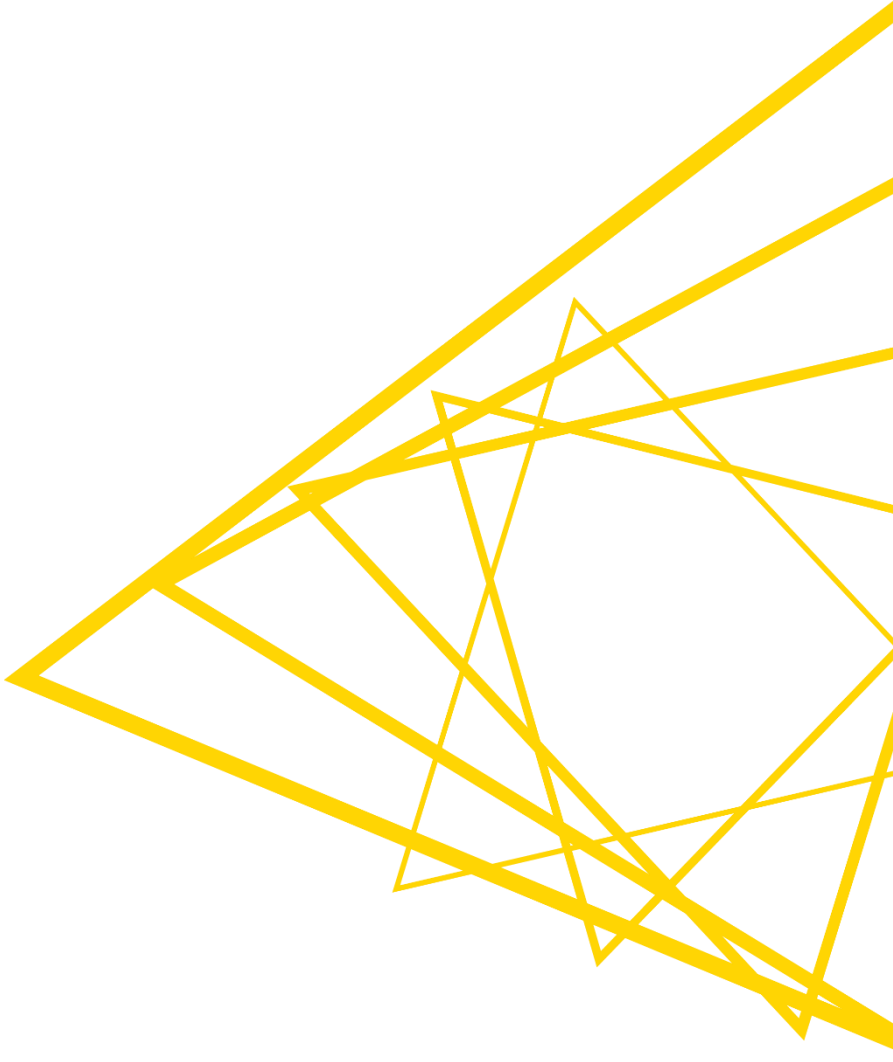


Efficient Usage of Databases

- Make sure to use the new Database framework instead of the legacy version
  - “DB nodes” are new
  - “Database nodes” are legacy
- Avoid using the DB connection node multiple times if connecting to the same database
  - Unless you need to open up parallel connections to speed up execution of parallel branches
- Push processing to database server when possible – don’t do it locally
- Always use the DB Connection Closer node at the end of your DB processing

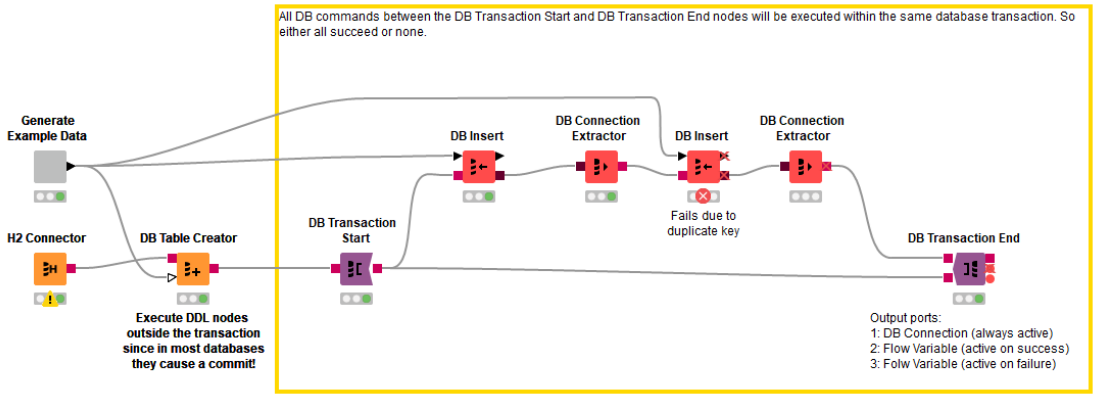


Error Handling



Recap: Transaction Support

- DB Transaction Start/End
  - Group several database data manipulation operations into a single unit of work – transaction
  - The transaction either completes entirely or not at all
    - If successful, the transaction ends with a commit that makes all changes visible to other users
    - If not, the transaction ends with a rollback returning the database to the state at the beginning of the transaction
  - Uses the default isolation level of the connected database



Workflow is available on the KNIME Community Hub:
https://kni.me/w/kWP1OhaY4DXK444n

Session 2: Summary

Now you should be able to:

- Apply various data anonymization techniques
- Recognize advanced KNIME Database extension functionality
- Query and update a database records
- Build the application that anonymizes the new customer data and uploads it to the database

Session 3: ELT, Big Data, Hadoop, Spark

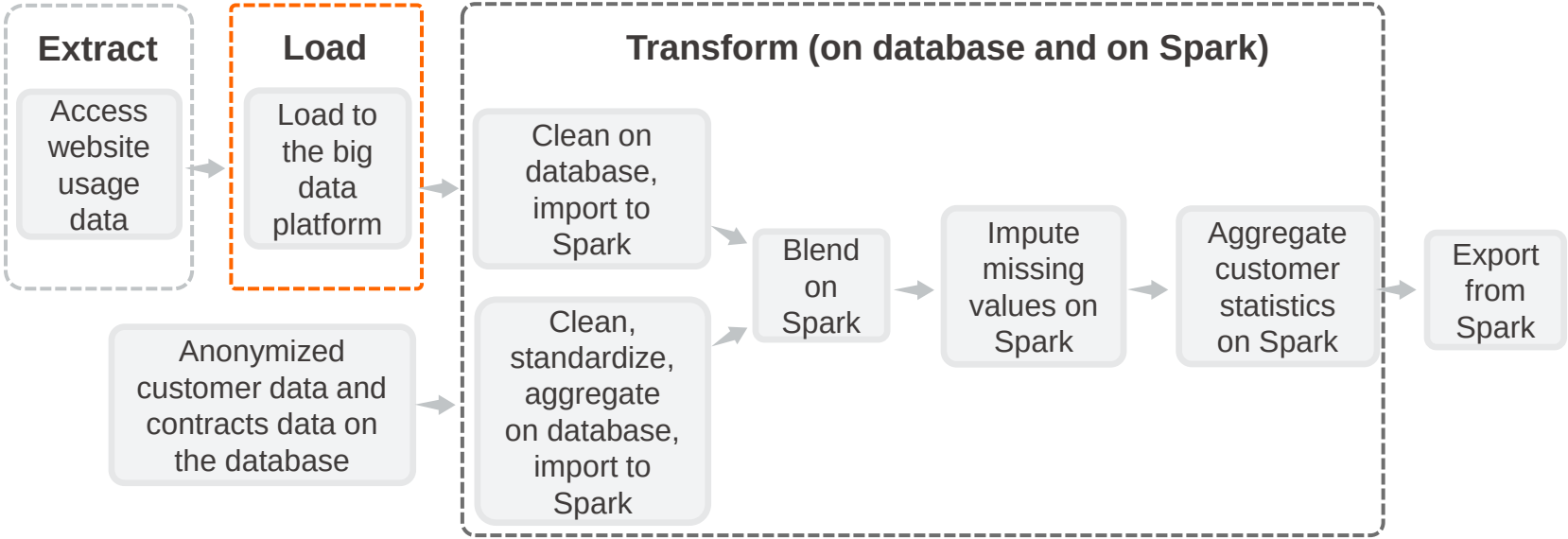
Session 3: Learning Outcomes

At the end of this session, you will be able to:

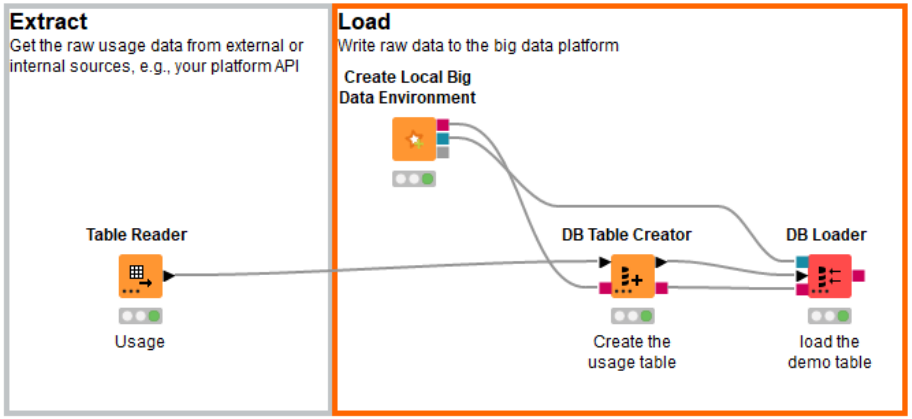
- Integrate Hadoop applications, such as Hive, HDFS, and Spark into KNIME Analytics Platform
- Process relatively large data on a database server and on Spark
- Train and apply a machine learning model on Spark
- Impute missing values on Spark
- Build the application that aggregates website usage data, personal data, and contract data into a customer statistics table

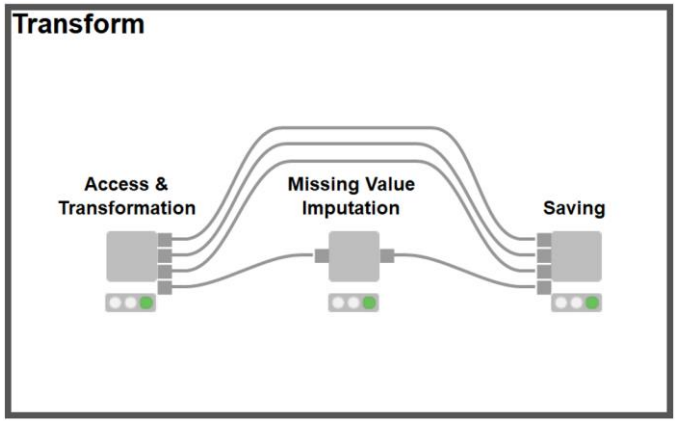
Today‘s example: ELT on Usage data

Session 3: ELT on Usage Data



Today’s Example: ELT on Usage Data





Data Set

Usage

Custom...	SessionID	IP	Property	Start	End	Duration	NrClicks	NrPage...	AvgScr...	NrSear...	Search...	AvgPag...	AvgRe...	NrCras...	NrLikes	NrDislikes	NrShares	SessionSatisfactionScore
11001	SID0000000	192.0.2.1	?	2016-08-17T17:44...	2016-08-17T17:50...	6	188	6	19	9	9	594	594	4	9	4	2	1.0
11001	SID0000001	192.0.2.1	?	2016-08-18T23:36...	2016-08-18T23:53...	17	503	16	50	25	25	1,527	1,527	10	25	10	5	3.0
11001	SID0000002	192.0.2.1	?	2016-10-02T20:48...	2016-10-02T21:07...	19	466	15	47	23	23	1,419	1,419	9	23	9	5	3.0
11001	SID0000003	192.0.2.1	?	2016-11-28T21:12...	2016-11-28T21:28...	16	658	20	66	33	33	1,985	1,985	13	33	13	7	4.0
11001	SID0000004	192.0.2.1	?	2017-03-15T09:12...	2017-03-15T09:32...	20	987	30	99	49	49	2,963	2,963	20	49	20	10	6.0
11001	SID0000005	192.0.2.1	?	2017-05-16T06:48...	2017-05-16T06:53...	5	826	25	83	41	41	2,483	2,483	17	41	17	8	5.0
11001	SID0000006	192.0.2.1	?	2017-07-12T23:44...	2017-07-13T00:00...	16	373	12	37	19	19	1,142	1,142	7	19	7	4	?
11001	SID0000007	192.0.2.1	?	2017-09-23T17:52...	2017-09-23T18:10...	18	567	17	57	28	28	1,718	1,718	11	28	11	6	3.0

Contracts

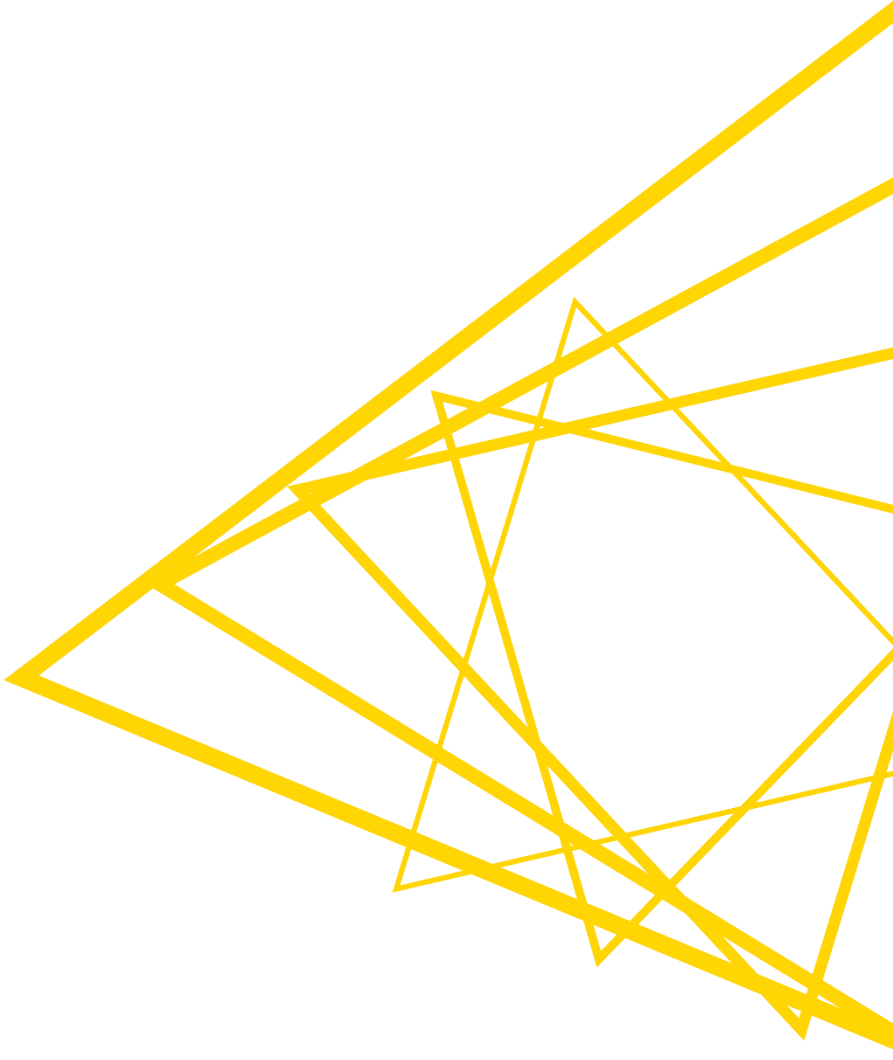
CustomerKey	ContractID	Products	Value	Date
11005	C000003	Private Investment	860.395	2016-01-15
11013	C000006	Private Investment	1,180.312	2016-03-23
11022	C000008	Fund Manager+	690.215	2016-11-11
11023	C000009	Private Investment	912.719	2016-10-03
11029	C000011	Private Investment	1,747.002	2016-08-01
11030	C000012	Private Investment	193.873	2016-01-04

Statistics

Custom...	TotalActivity	AverageDuration	LastActivity	VisitFrequency	AverageSessionSatisfactionScore	TotalContractValue	NumberOfContracts	DaysSinceLastActivity
23300	33	16.5	2017-10-16T11:28...	2	3.5	2,456.139	2	532
23307	255	11.087	2018-06-04T18:59...	23	3.13	1,001.355	1	301
23310	283	10.885	2017-05-26T14:32...	26	3.346	1,501.292	1	675
23316	178	13.692	2017-01-26T18:52...	13	2.923	2,608.389	2	795
23337	275	9.483	2017-09-24T10:42...	29	2.897	1,208.75	1	554
23345	325	12.037	2018-03-26T12:41...	27	2.593	1,163.283	1	371
23346	146	10.429	2018-07-01T10:51...	14	3	281.948	1	274

The dataset is generated randomly. Any reference to living persons or real events is purely coincidental

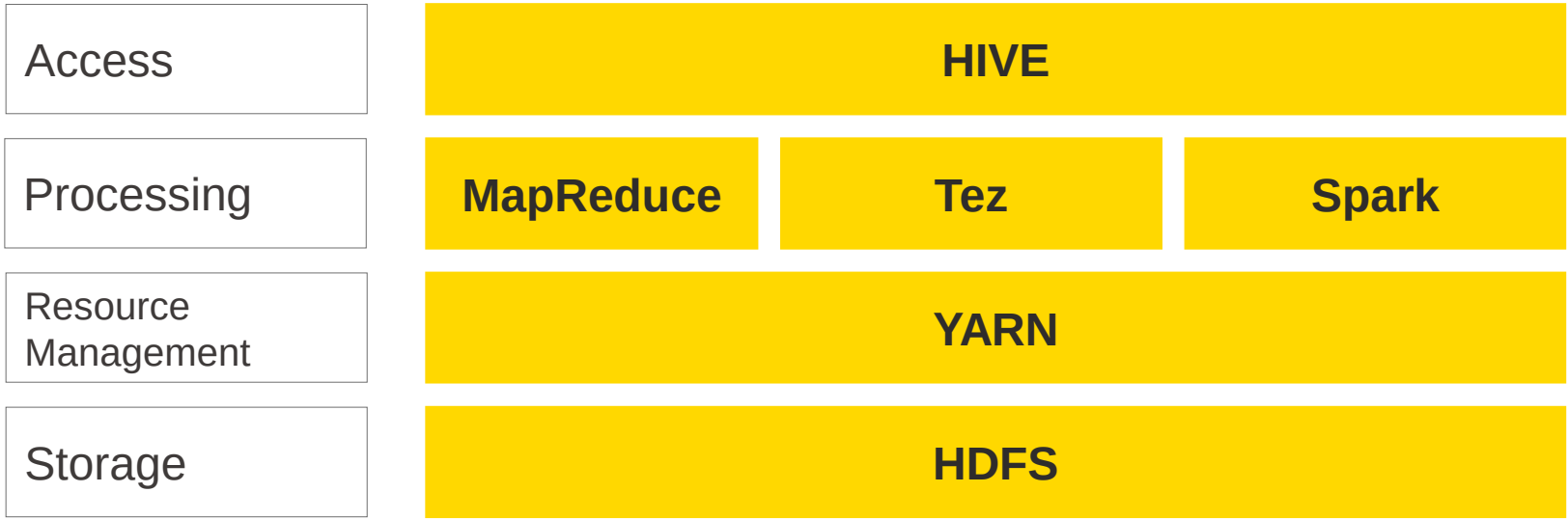
A Quick Intro to Hadoop



Apache Hadoop

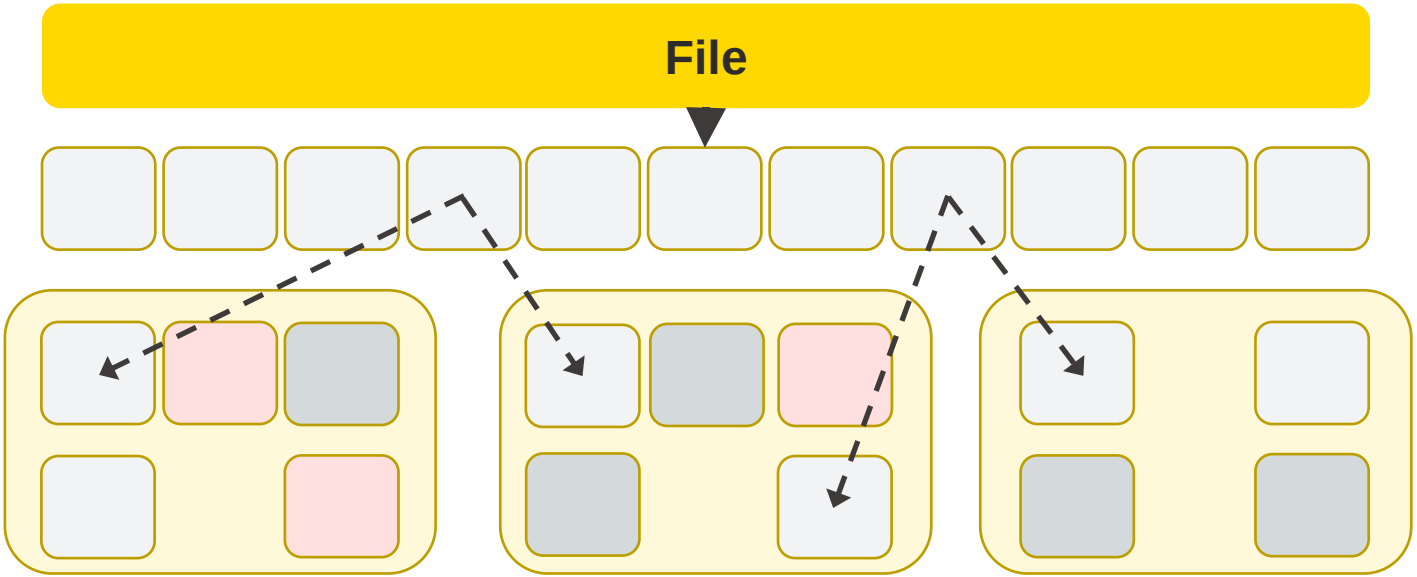
- Open-source framework for distributed storage and processing of large data sets
- Designed to scale up to thousands of machines
- Does not rely on hardware to provide high availability
  - Handles failures at application layer instead
- Spawned diverse ecosystem of products

Hadoop Ecosystem



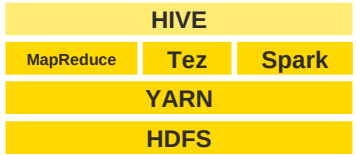
HDFS

- Hadoop distributed file system
- Stores large files across multiple machines
- Blocks of a file are replicated for fault tolerance
  - Aims: improve data reliability, availability, and network bandwidth utilization



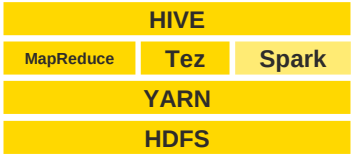
Hive

- **SQL-like database** on top of files in HDFS
- Provides data summarization, query, and analysis
- Interprets a set of files as a database table (schema information to be provided)
- Translates SQL queries to MapReduce, Tez, or Spark jobs
- Supports various file formats:
  - Text/CSV
  - SequenceFile
  - Avro
  - ORC
  - Parquet



Spark

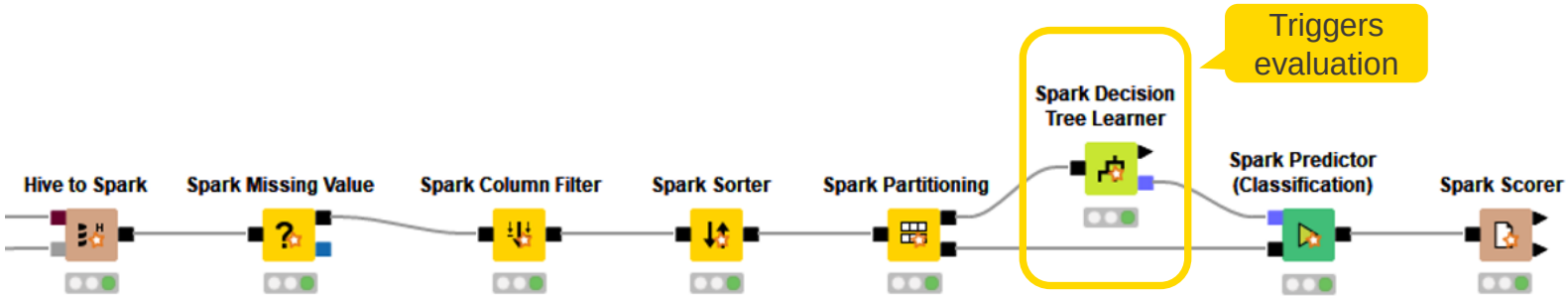
- Cluster computing framework for large-scale data processing
- Keeps large working datasets in memory between jobs
  - No need to always load data from disk -> much (!) faster than MapReduce
- Programmatic interface
  - Scala, Java, Python, R
  - Functional programming paradigm: map, flatmap, filter, reduce, fold, …
- Great for:
  - Iterative algorithms
  - Interactive analysis



Spark DataFrame

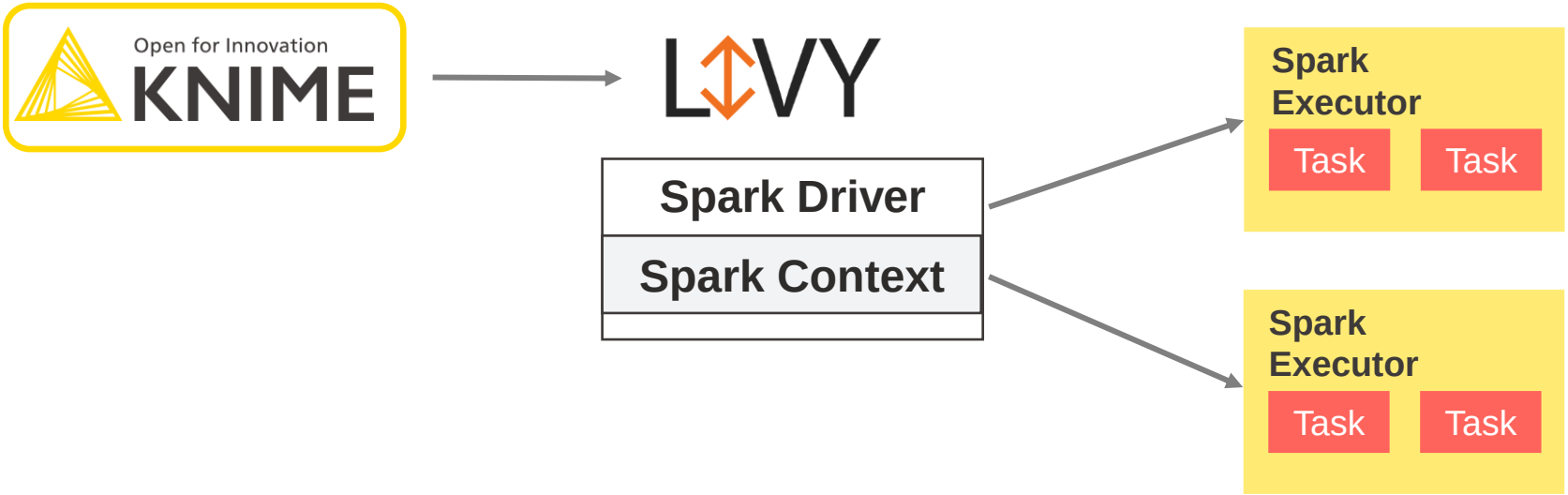
Name	Surname	Age
John	Doe	35
Jane	Roe	29
…	…	…

- Table-like
  - Collection of rows, organized in columns with names and types
- Immutable
  - Data manipulation = creating new DataFrame from an existing one by applying a function on it
- Distributed
  - Each row belongs to exactly one partition
  - Each partition is held by a Spark Executor
- Lazily evaluated
  - Functions are not executed until an action that requests to see the data is triggered

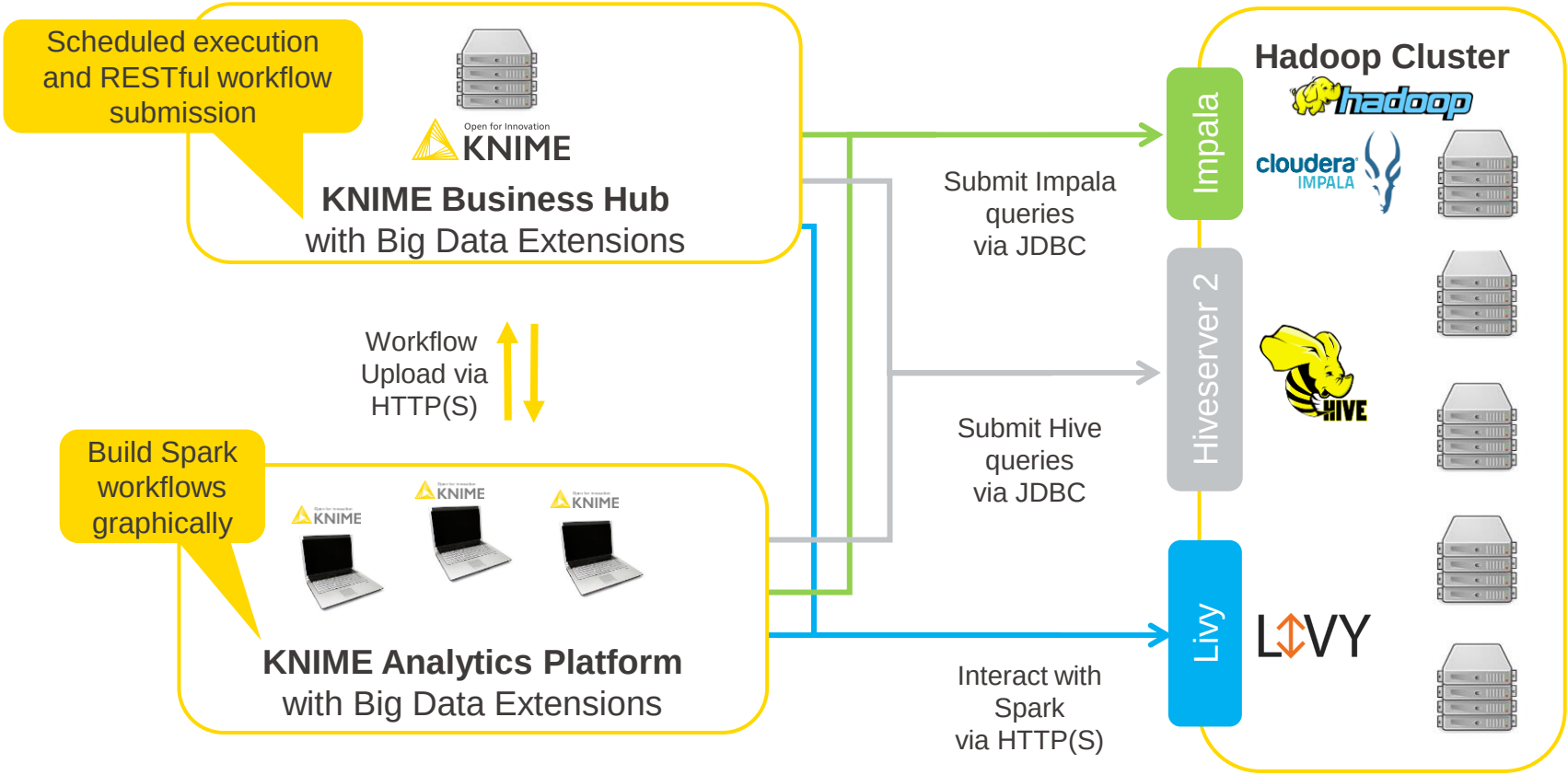


Spark Context

- Spark Context
  - Main entry point for Spark functionality
  - Represents connection to a Spark cluster
  - Allocates resources on the cluster

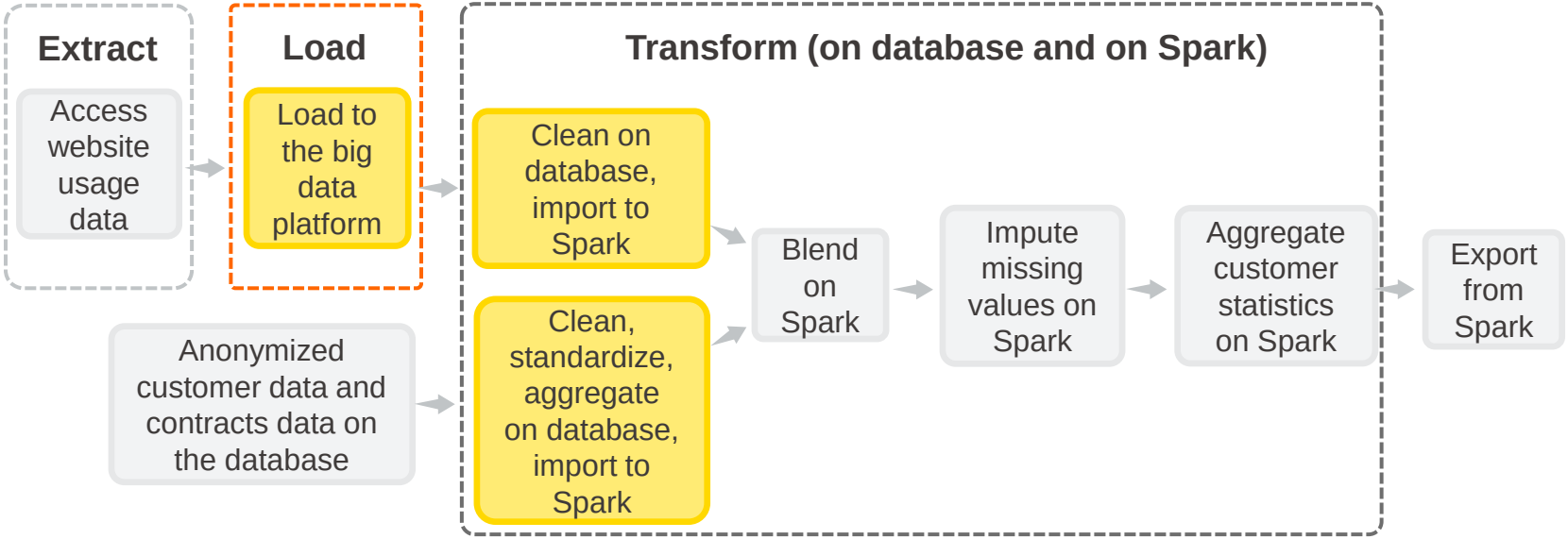


Big Data Architecture with KNIME



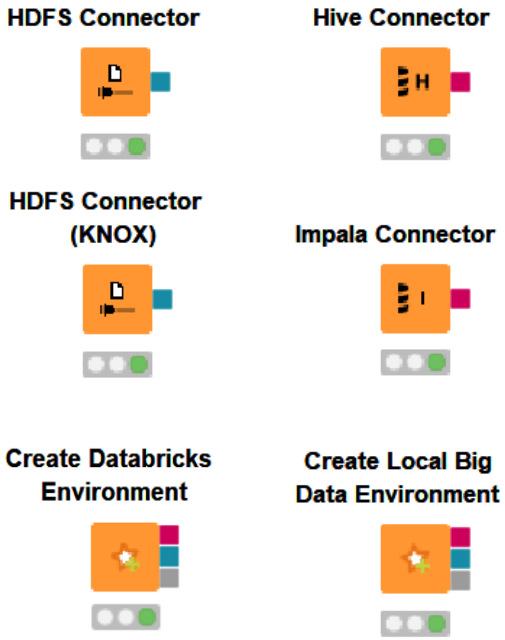
Big Data Connectors, IO & In-Database Processing on Hadoop

IO & In-Database Processing on Hadoop



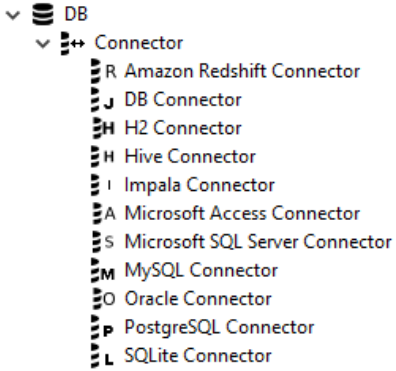
KNIME Big Data Connectors

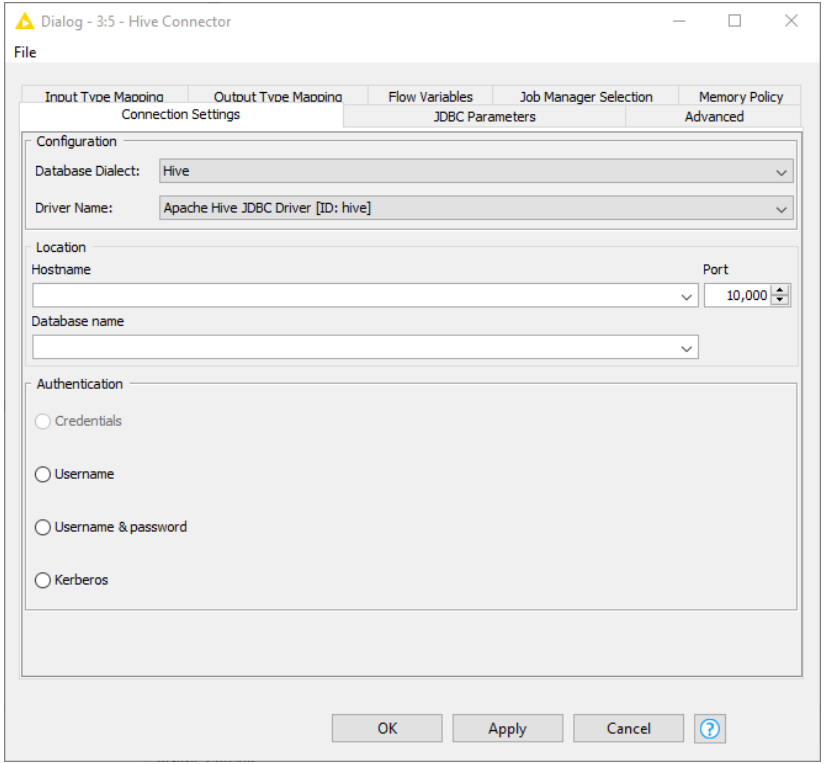
- Package required drivers/libraries
- Preconfigured connectors
  - HDFS, Hive, Impala, Vertica
  - Cloud: Amazon S3, Google BigQuery, Databricks, Amazon Athena, etc.
- Standard ports
  - Database and file system connectivity
  - Process data on Hadoop with regular KNIME Database extension nodes
  - Browse files and folders on distributed file systems with regular file handling framework



Hive Connector

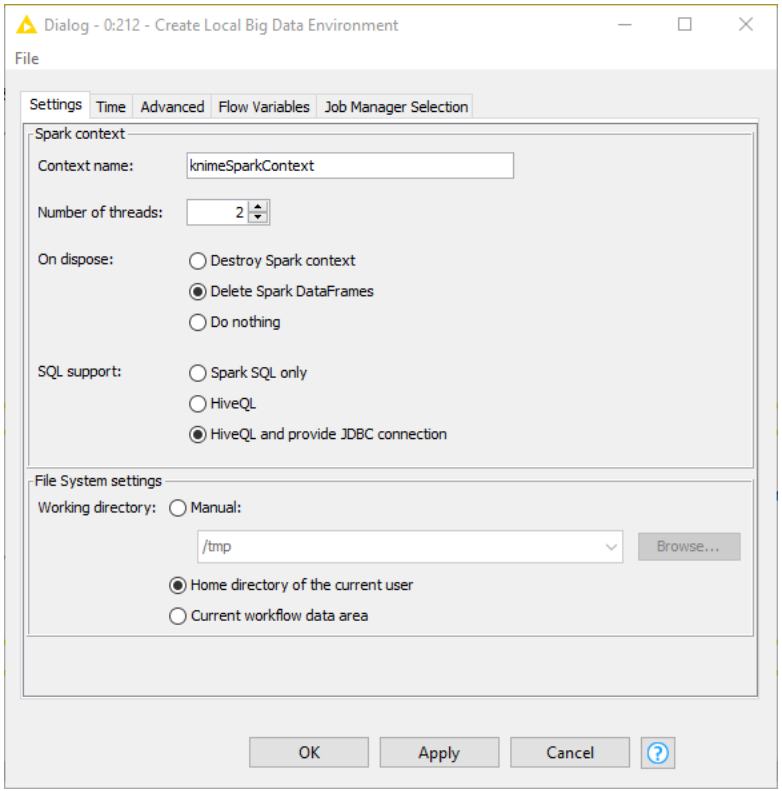
- Creates JDBC connection to Hive
- On unsecured clusters no password required





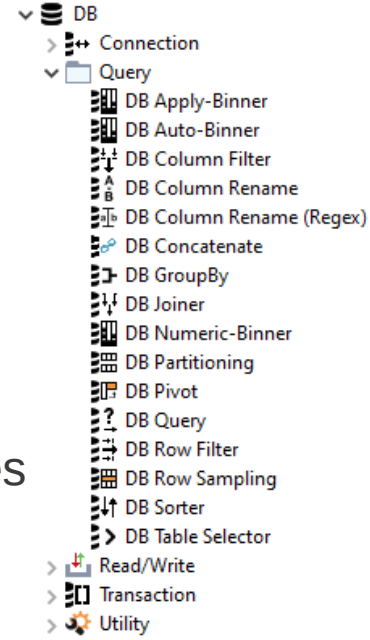
Create Local Big Data Environment Node

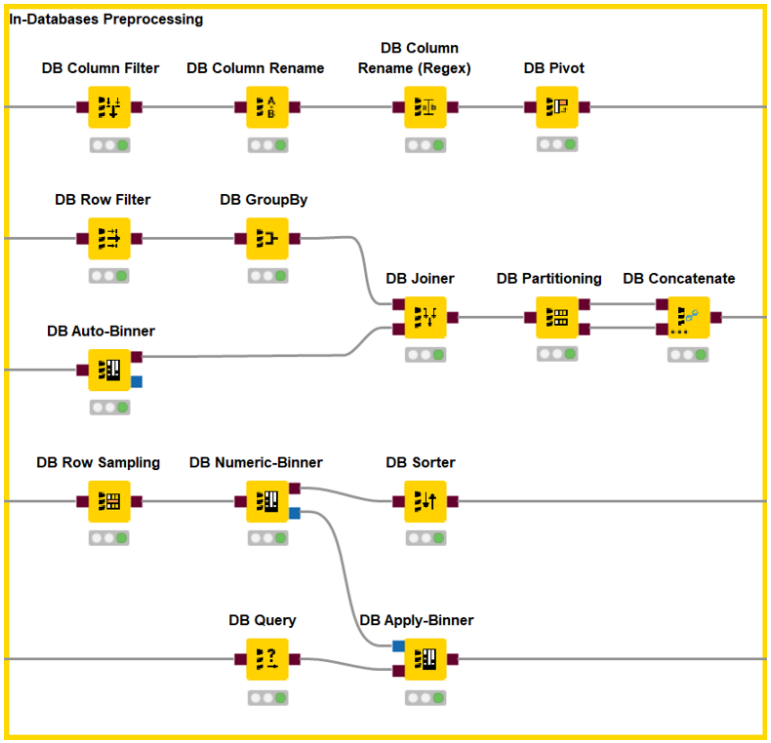
- Creates a fully functional big data environment on your local machine with
  - Apache Hive
  - HDFS
  - Apache Spark
- Try out Big Data nodes without Hadoop cluster
- Build and test workflows locally on sample data



Recap: Query Nodes

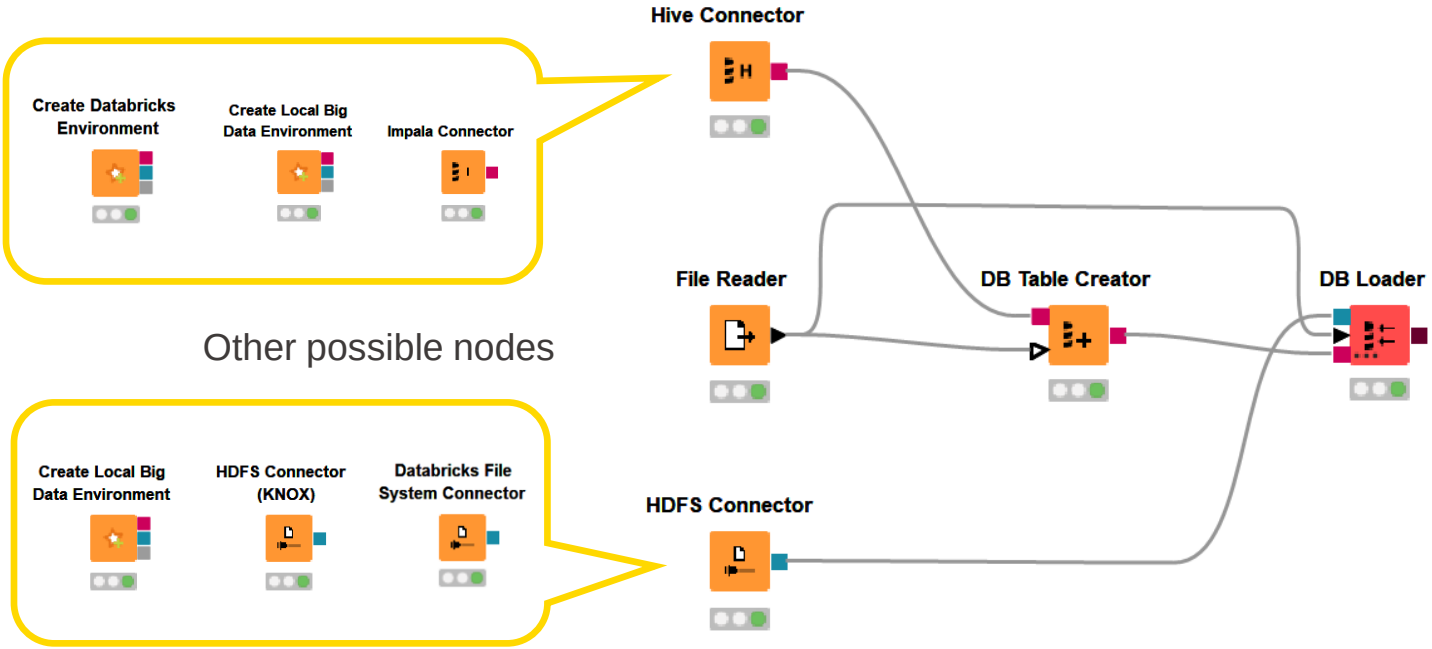
- Various manipulations
  - Filter rows and columns
  - Join and concatenate tables
  - Extract samples
  - Bin numeric columns
  - Sort
  - Aggregate
  - Write your own query
- Configuration is similar to KNIME Manipulation nodes (in most cases)
- No SQL coding
- The nodes construct and output a SQL query





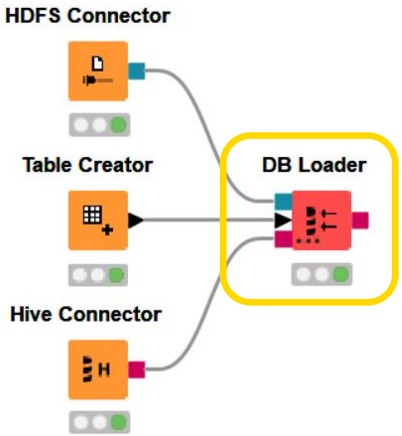
Loading Data into Hive/Impala

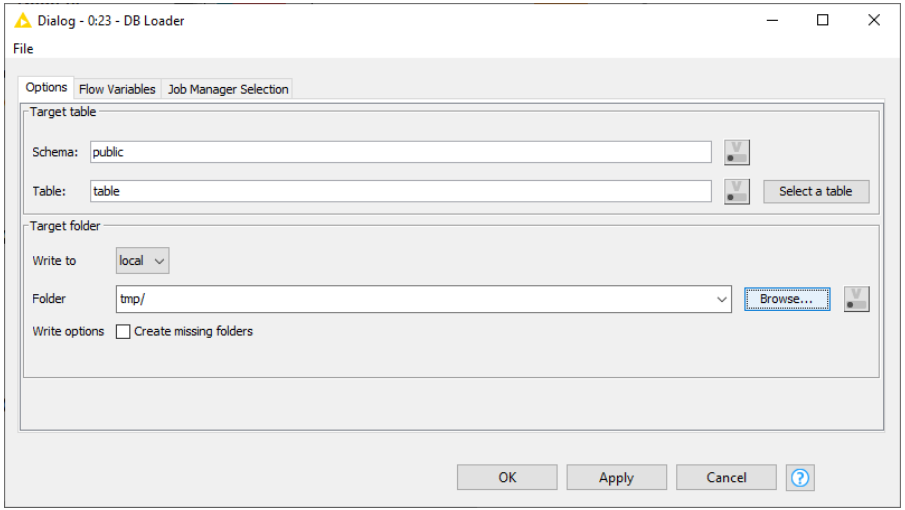
- Connectors are from KNIME Big Data Connectors Extension
- Use DB Table Creator and DB Loader from regular DB framework



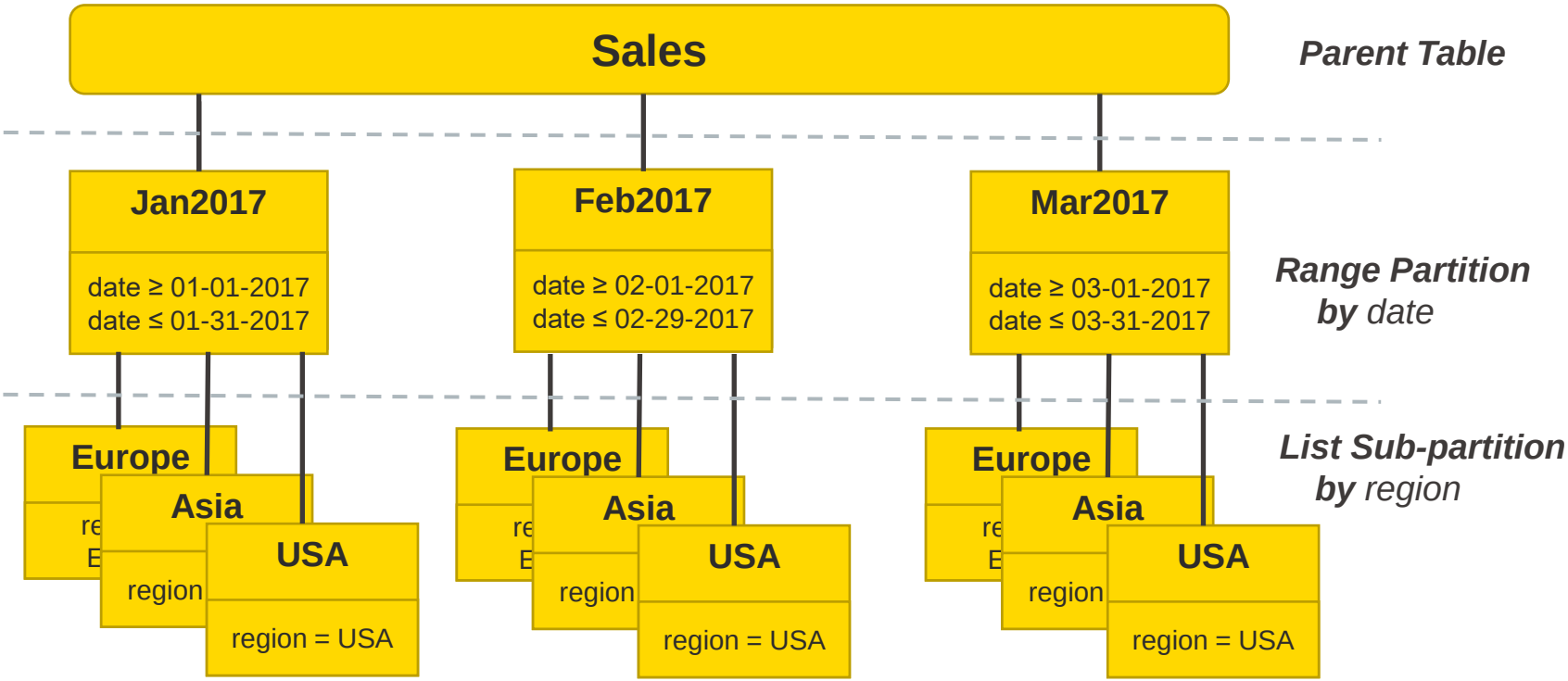
Recap: DB Loader Node

- Inserts data rows into an existing database table
  - Uses **fast** bulk loading for large amounts of data
  - Column names and order need to match exactly
  - Doesn't check the column type compatibility or the values itself
    - Can lead to a corrupt data table





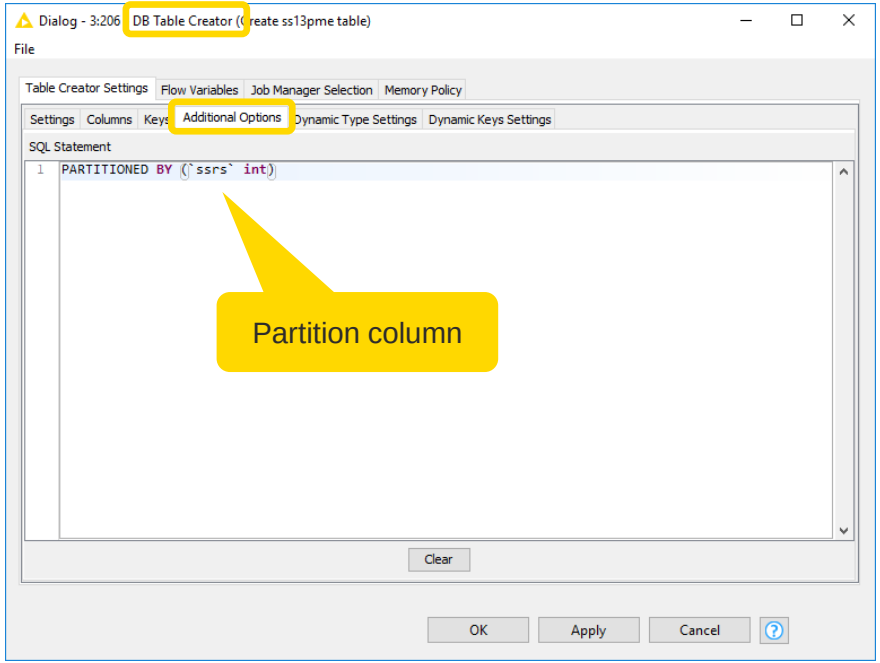
Hive Partitioning



Partitioning

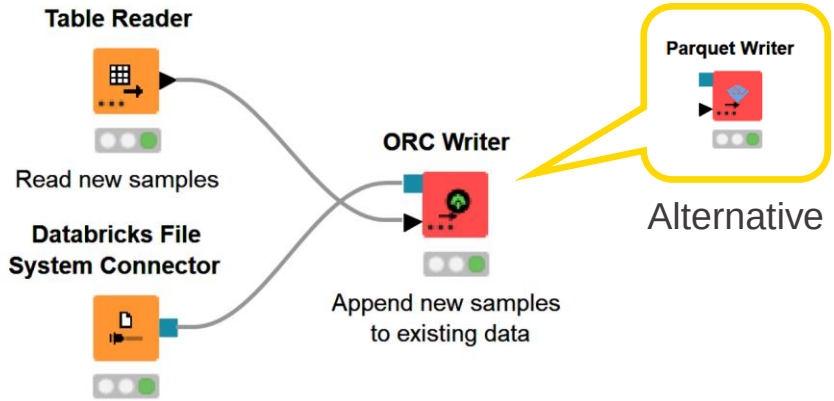
About partition columns:

- Optional (!) performance optimization
- Use columns that are often used in WHERE clauses
- Use only categorical columns with suitable value range, i.e. not too few distinct values (e.g. 2) and not too many distinct values (e.g. 10 million)
- Partition columns should not contain missing values



Columnar File Formats

- Improve performance when reading, writing, and processing data in HDFS
- Benefits:
  - **Efficient compression**: Stored as columns and compressed, which leads to smaller disk reads.
  - **Fast reads**: Data is split into multiple files. Files include a built-in index, min/max values, and other aggregates. In addition, predicate pushdown pushes filters into reads so that minimal rows are read.
  - **Proven in large-scale deployments**: Facebook uses the ORC file format for a 300+ PB deployment.
- Available in KNIME Analytics Platform: ORC and Parquet



Example: Columnar File Formats

ID	Gender	Age
1	female	45
2	male	20
⋮	⋮	⋮
3333	male	42

Metainformation
ID: min = 1 ; max = 3333
Gender: female; male
Age: min = 20; max = 45

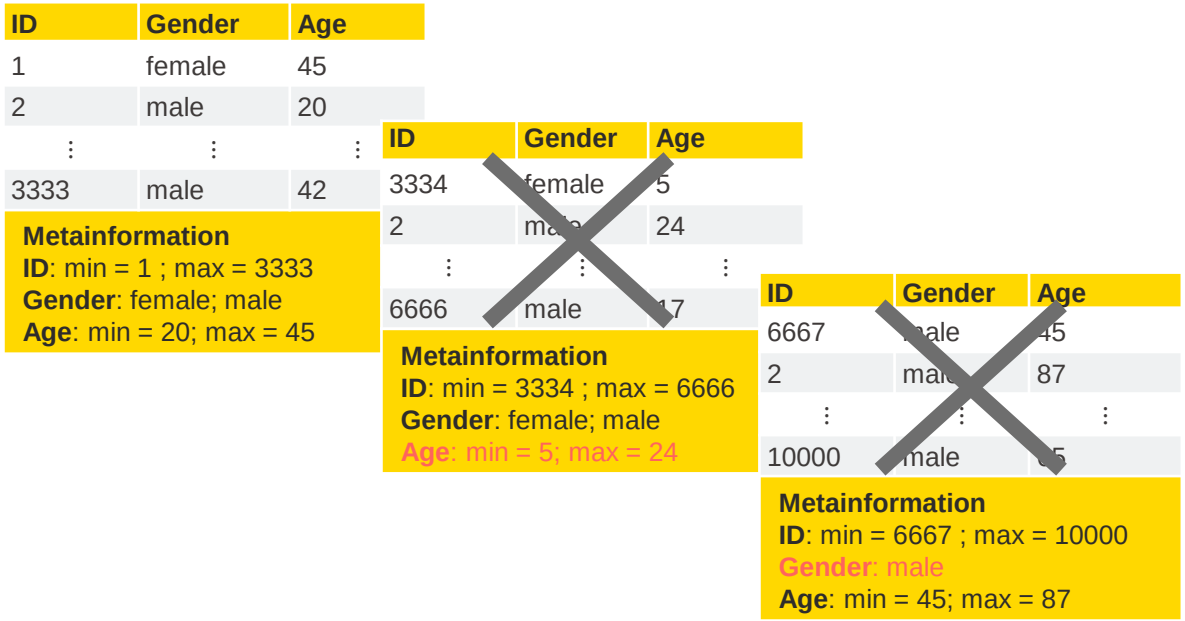
ID	Gender	Age
3334	female	45
2	male	20
⋮	⋮	⋮
6666	male	42

Metainformation
ID: min = 3334 ; max = 6666
Gender: female; male
Age: min = 5; max = 24

ID	Gender	Age
6667	male	45
2	male	87
⋮	⋮	⋮
10000	male	65

Metainformation
ID: min = 6667 ; max = 10000
Gender: male
Age: min = 45; max = 87

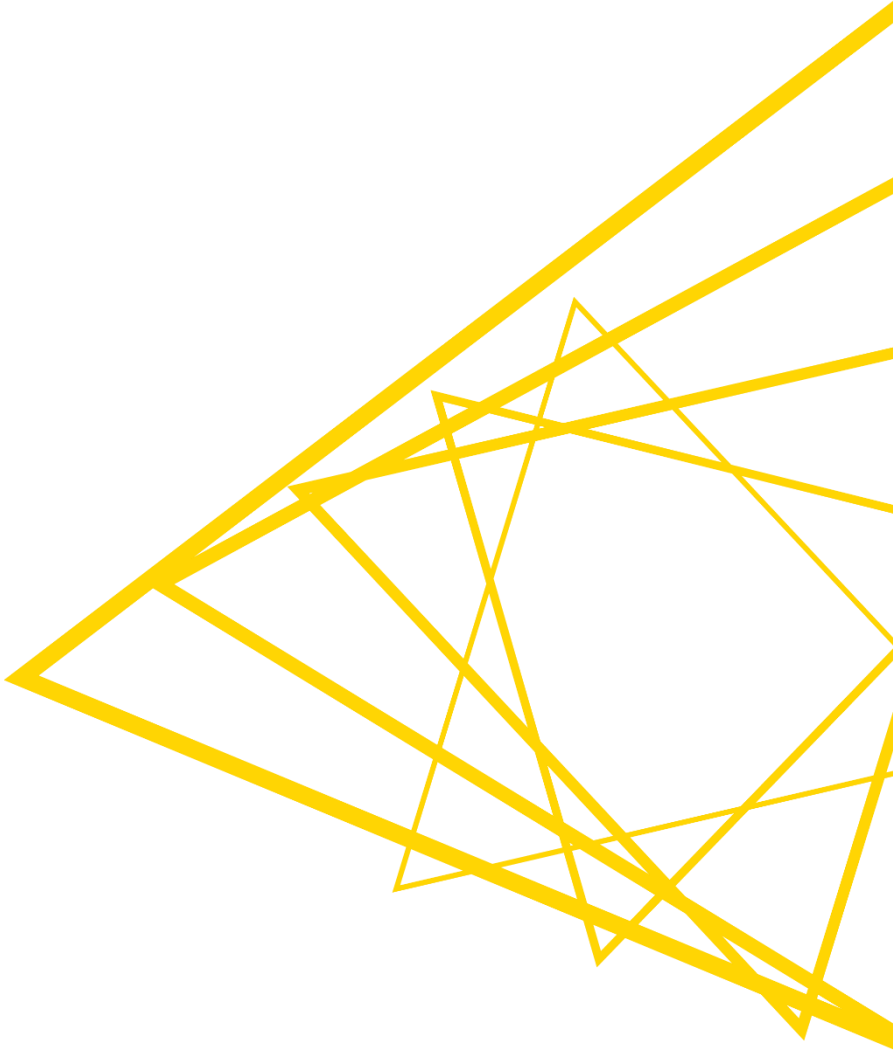
Example: Columnar File Formats



Select **ID** from table where

Age > 30 and Gender = female

KNIME Extension for Apache Spark



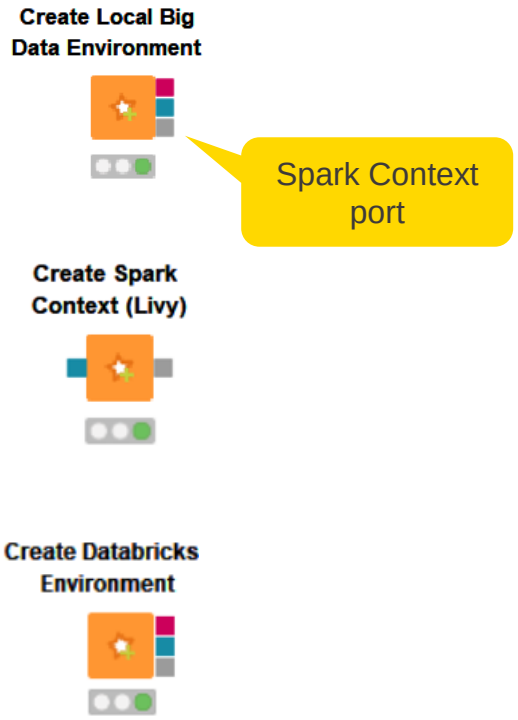
Apache Spark

- Runs on Hadoop
- Supported Spark Versions
  - 1.2, 1.3, 1.5, 1.6, 2.x, 3.1, 3.2
  - One KNIME extension for all Spark versions
- Nodes for IO, data transformation, statistics, mining + scripting nodes
- Scalable machine learning library (Spark MLlib and spark.ml)
- Algorithms for
  - Classification (decision tree, naïve Bayes, logistic regression, …)
  - Regression (linear regression, …)
  - Clustering (k-means)
  - Collaborative filtering (ALS)
  - Dimensionality reduction (SVD, PCA)
  - Item sets / Association rules

Spark Context: Creating

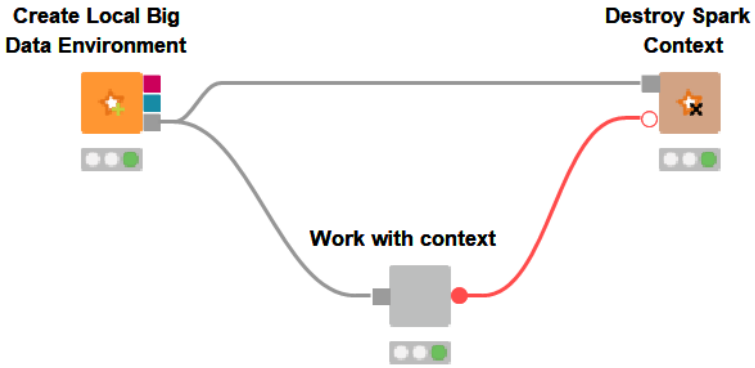
Three nodes to create a Spark context:

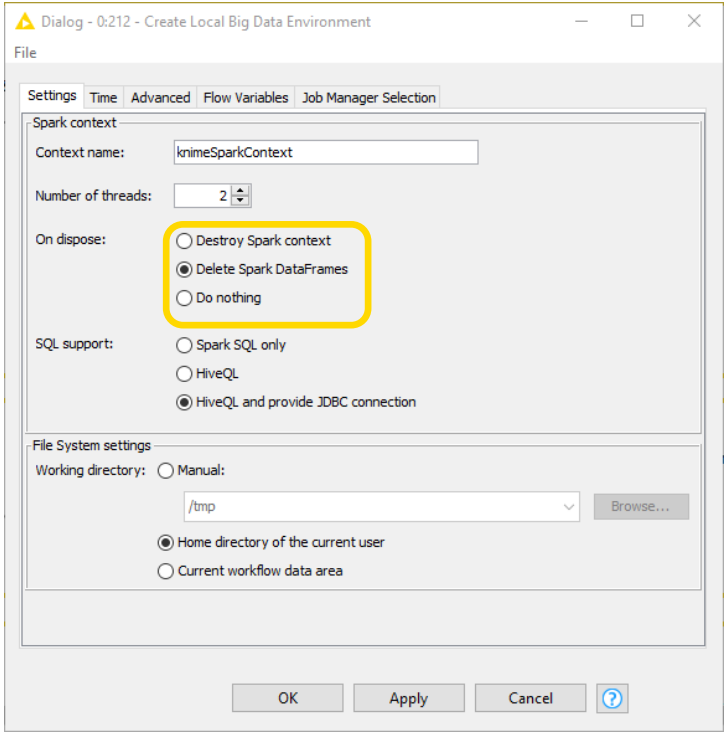
- Create Local Big Data Environment
  - Runs Spark locally on your machine (no cluster required)
  - Good for workflow prototyping
- Create Spark Context (Livy)
  - Requires a cluster that provides the Livy service
    - e.g. Amazon EMR, Azure HDInsight
  - Good for production use
- Create Databricks Environment
  - Runs Spark on a remote Databricks cluster
  - Good for large-scale production use



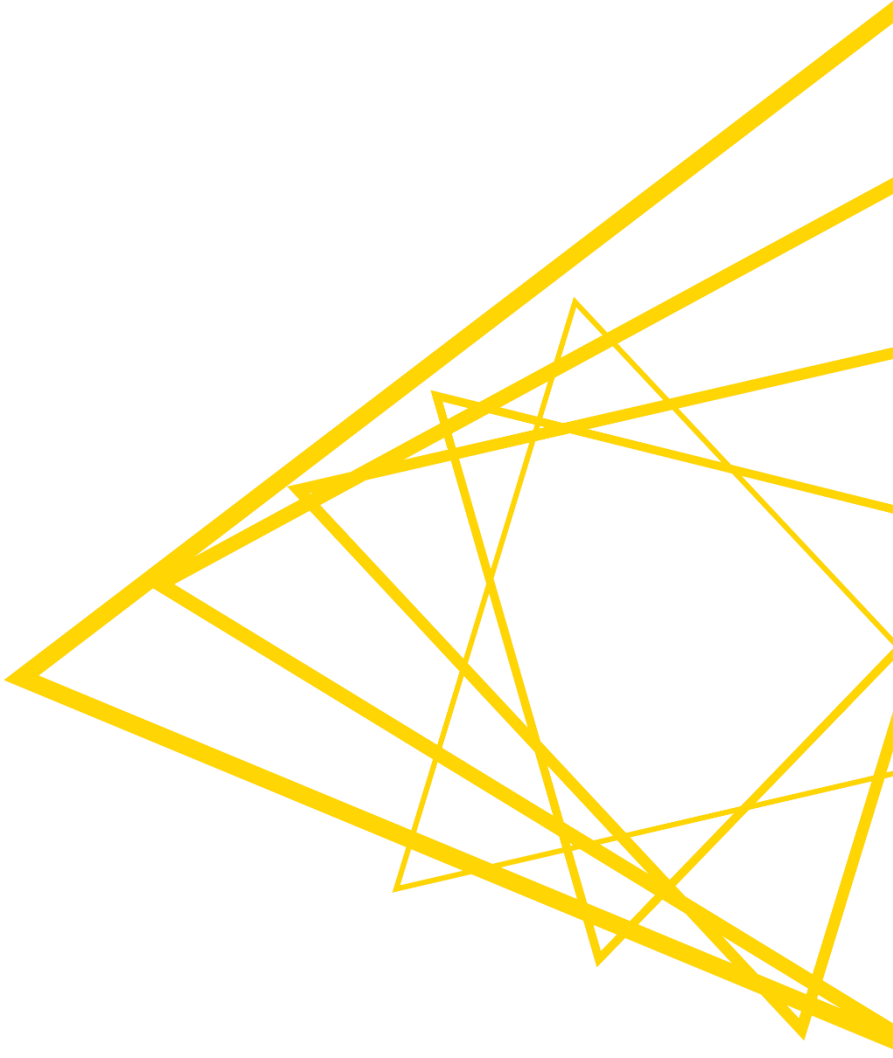
Spark Context: Using, Destroying

- Spark Context port is required by all Spark nodes
- Destroying a Spark Context
  - destroys all Spark DataFrames within the context
  - frees up the resources Spark Context allocated on the cluster

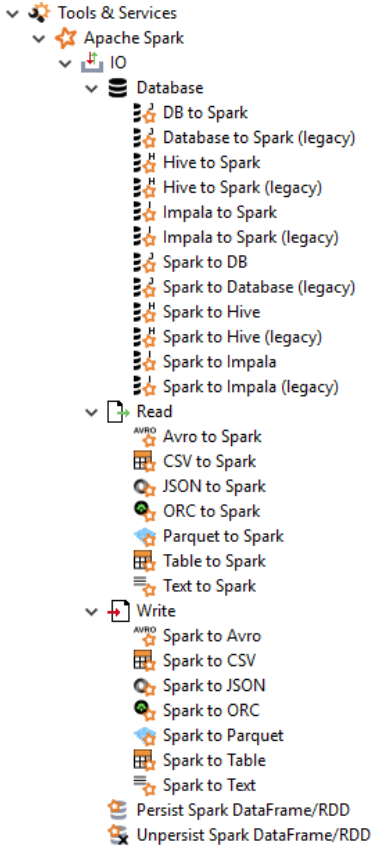




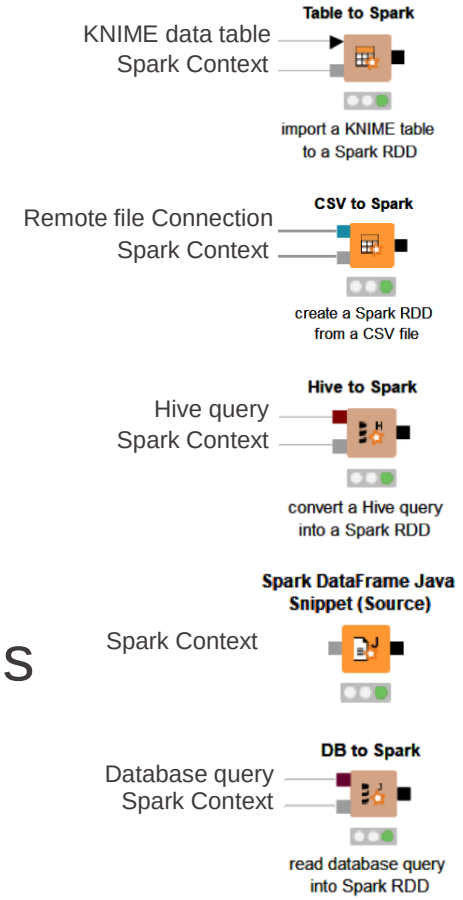
Import Data to Spark



Import Data to Spark

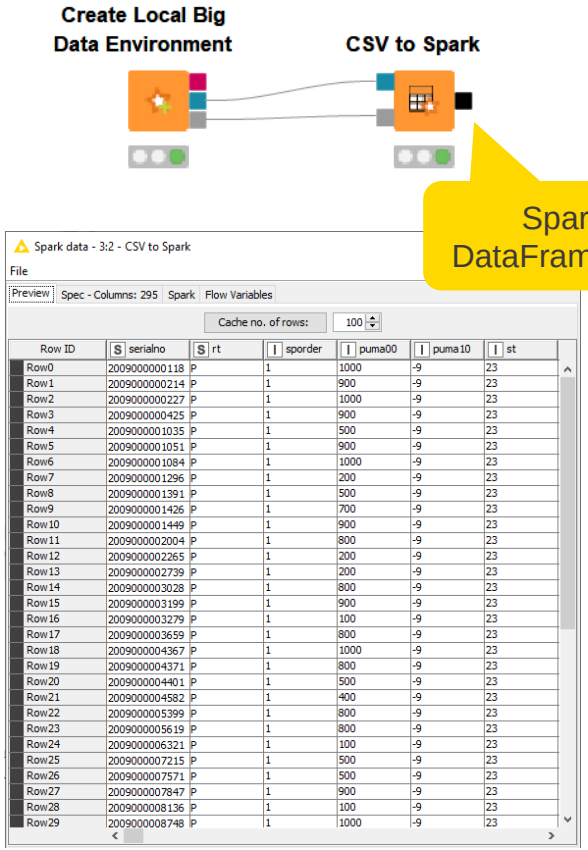


- From KNIME
- From CSV file in HDFS
- From Hive
- From other sources
- From Database



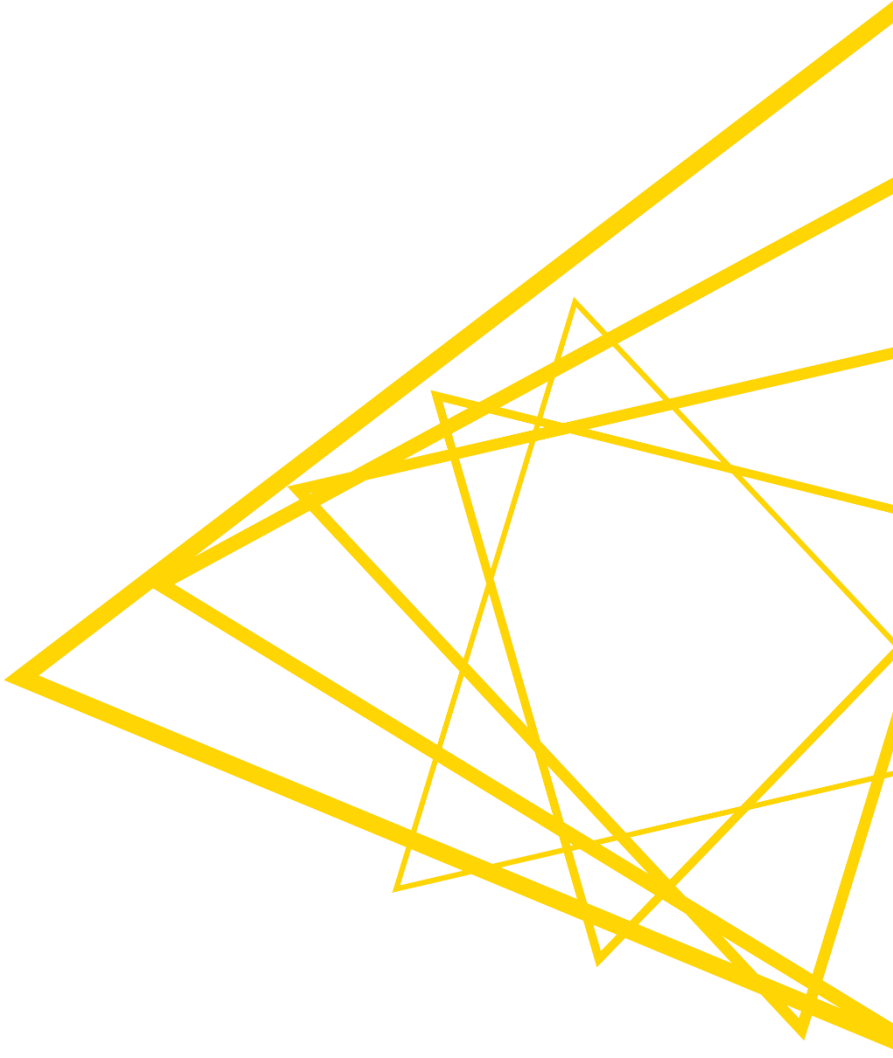
Spark DataFrame Ports

- Spark DataFrame port points to a DataFrame in Spark cluster
- Data stays within Spark
  - Output port provides data preview and column information
- Reminder: Lazy Evaluation
  - A green node status does not always mean that computation has been performed!

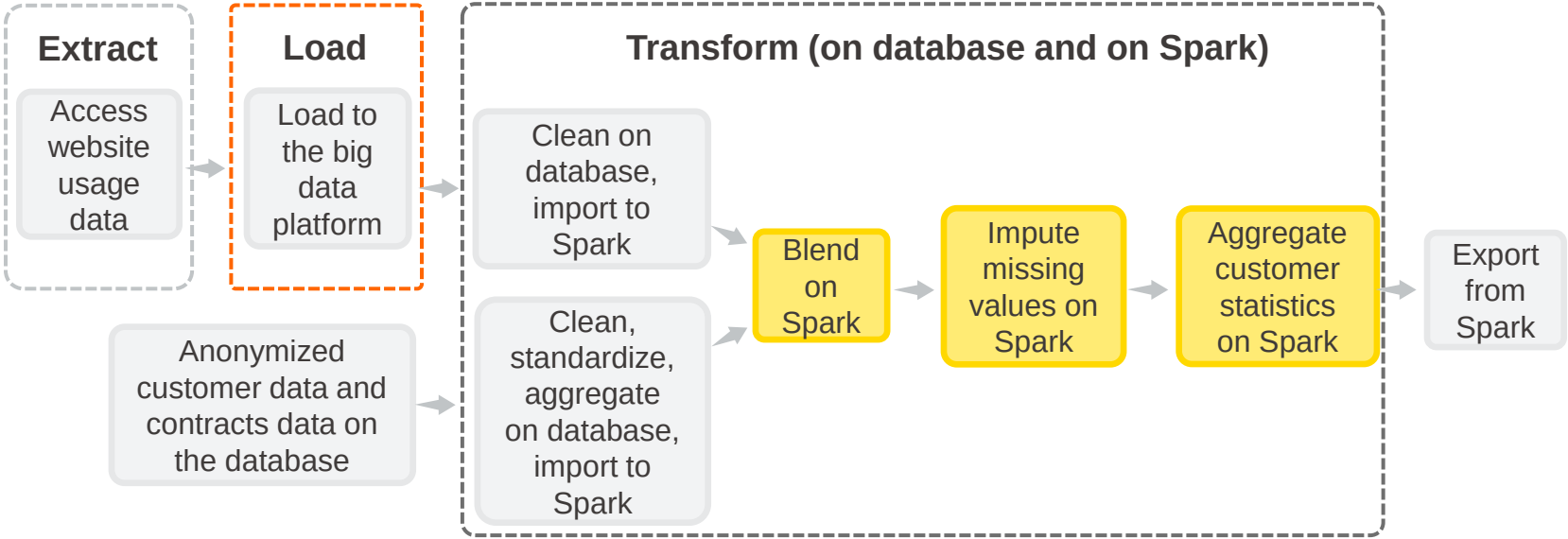


Row ID	serialno	rt	sporder	puma00	puma10	st
Row0	2009000000118	P	1	1000	-9	23
Row1	2009000000214	P	1	900	-9	23
Row2	2009000000227	P	1	1000	-9	23
Row3	2009000000425	P	1	900	-9	23
Row4	2009000001035	P	1	500	-9	23
Row5	2009000001051	P	1	900	-9	23
Row6	2009000001084	P	1	1000	-9	23
Row7	2009000001296	P	1	200	-9	23
Row8	2009000001391	P	1	500	-9	23
Row9	2009000001426	P	1	700	-9	23
Row10	2009000001449	P	1	900	-9	23
Row11	2009000002004	P	1	800	-9	23
Row12	2009000002265	P	1	200	-9	23
Row13	2009000002739	P	1	200	-9	23
Row14	2009000003028	P	1	800	-9	23
Row15	2009000003199	P	1	900	-9	23
Row16	2009000003279	P	1	100	-9	23
Row17	2009000003659	P	1	800	-9	23
Row18	2009000004367	P	1	1000	-9	23
Row19	2009000004371	P	1	800	-9	23
Row20	2009000004401	P	1	500	-9	23
Row21	2009000004582	P	1	400	-9	23
Row22	2009000005399	P	1	800	-9	23
Row23	2009000005619	P	1	800	-9	23
Row24	2009000006321	P	1	100	-9	23
Row25	2009000007215	P	1	500	-9	23
Row26	2009000007571	P	1	500	-9	23
Row27	2009000007847	P	1	900	-9	23
Row28	2009000008136	P	1	100	-9	23
Row29	2009000008748	P	1	1000	-9	23

Pre-Processing with Spark

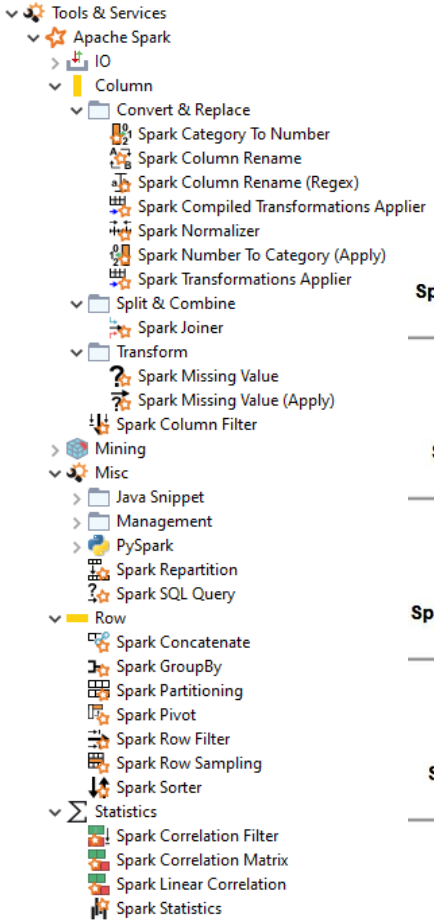


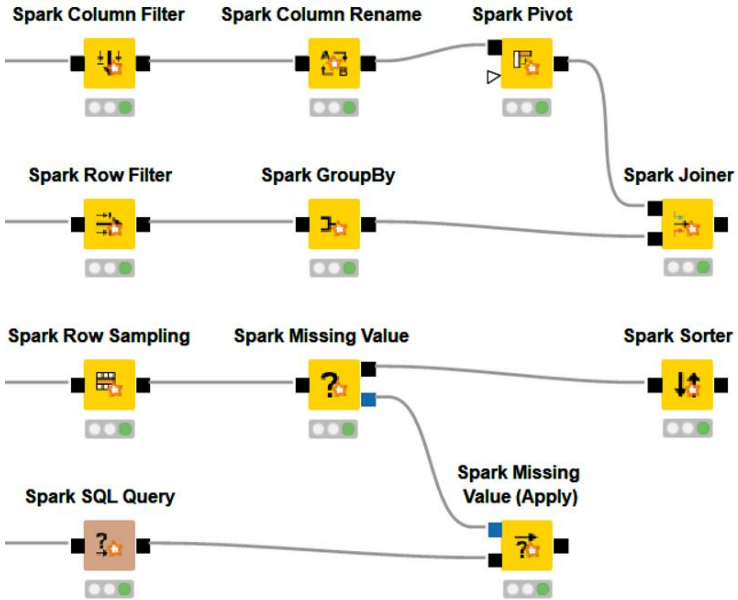
Pre-Processing with Spark



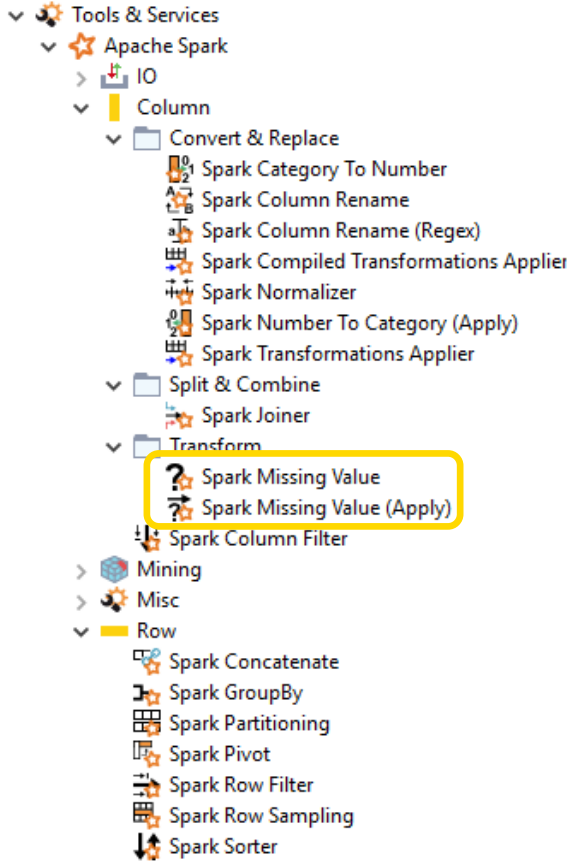
Spark Query Nodes

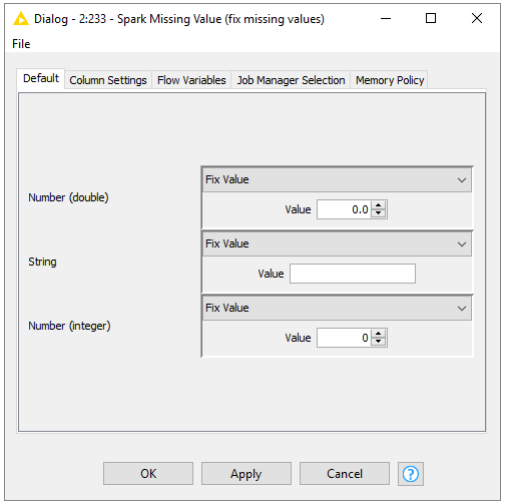
- Various manipulations
  - Filter rows and columns
  - Join tables
  - Extract samples
  - Sort
  - Aggregate
  - Write your own query
- Configuration is similar to KNIME Manipulation nodes (in most cases)
- No coding

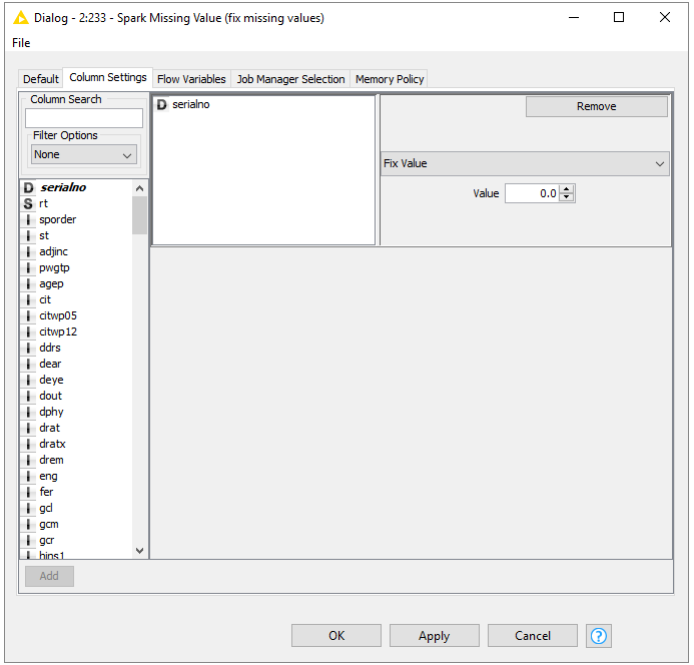




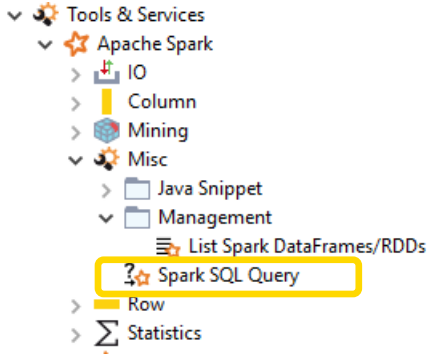
Spark Missing Value Node

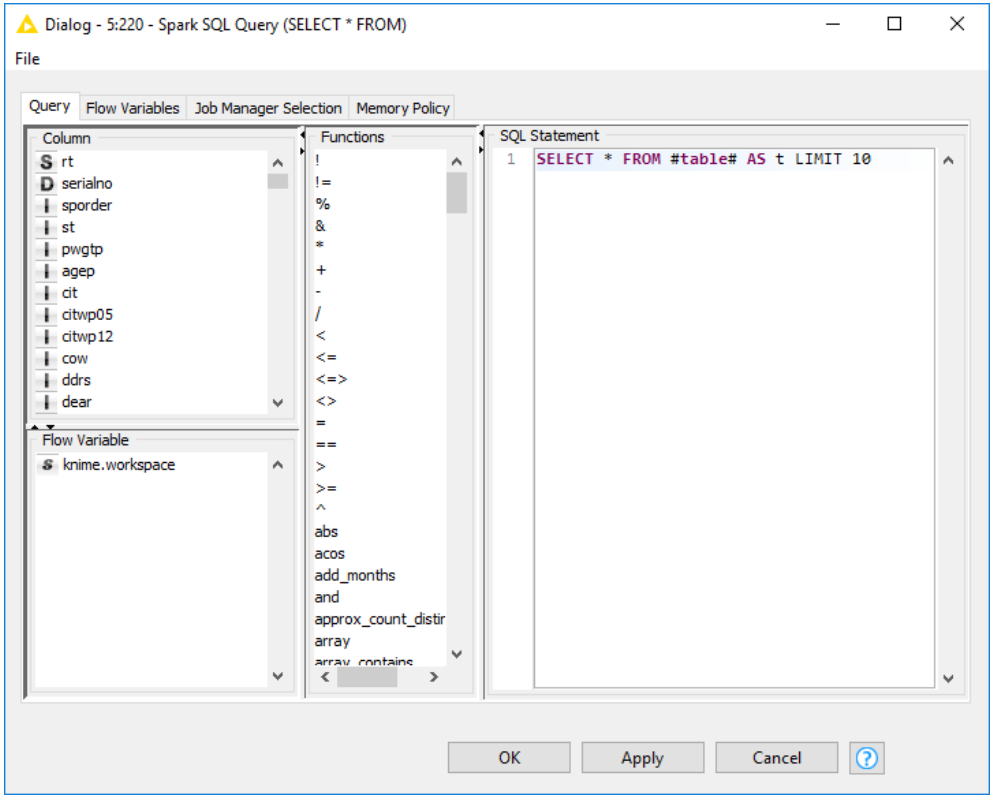






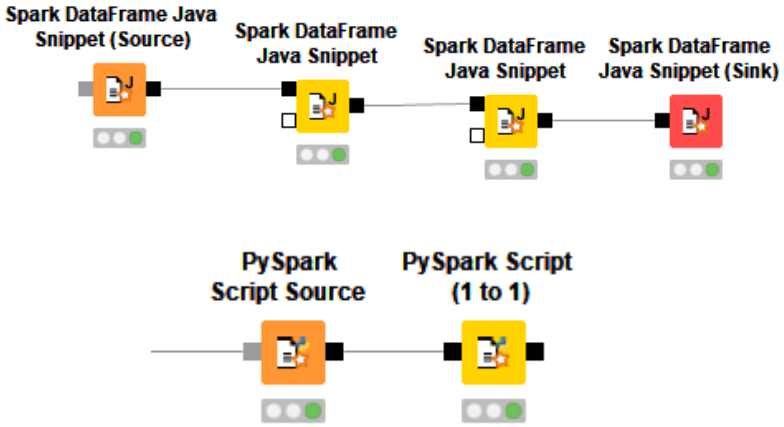
Spark SQL Query Node

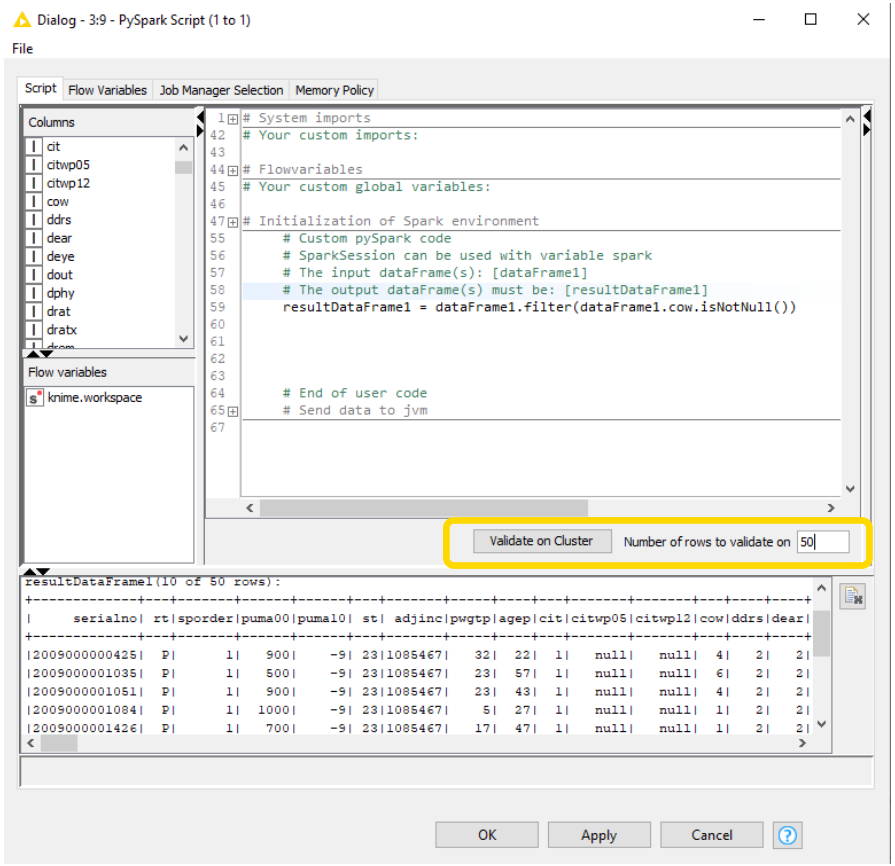




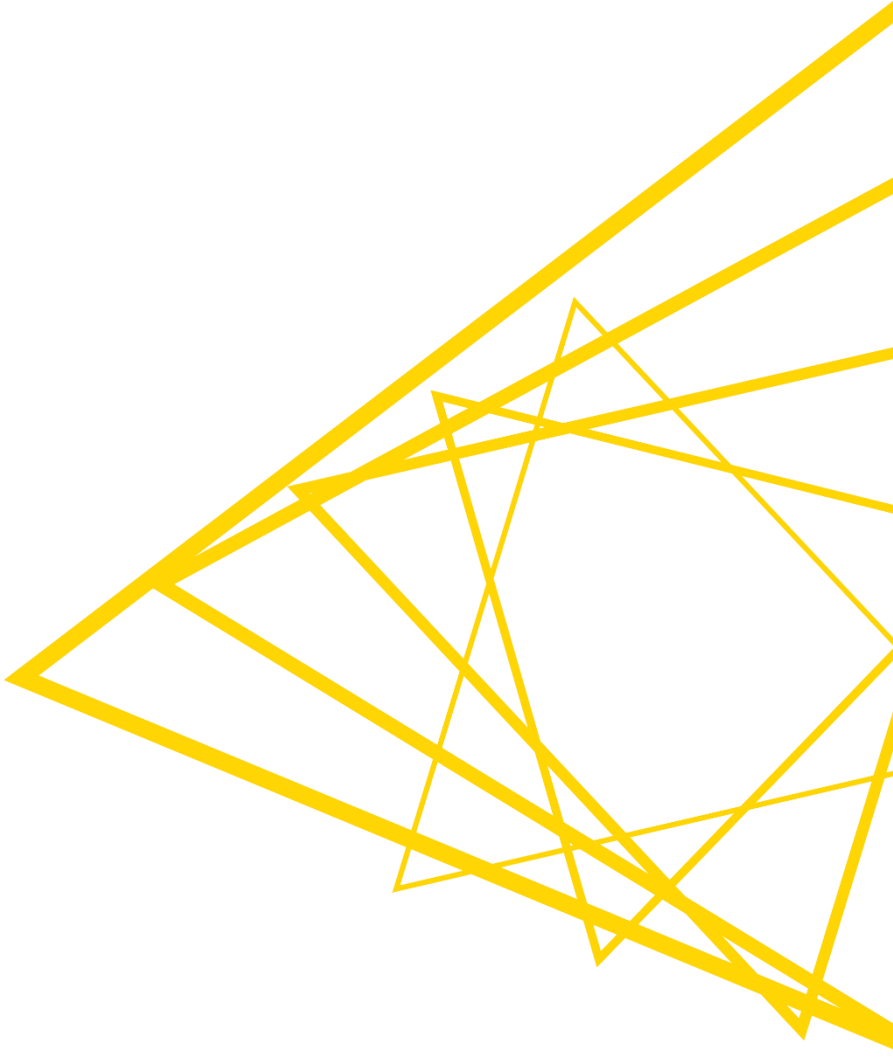
Modularize and Execute Your Own Spark Code

- Java Snippets
- PySpark Script
  - Validate on cluster





Best Practices: Spark Performance Optimization

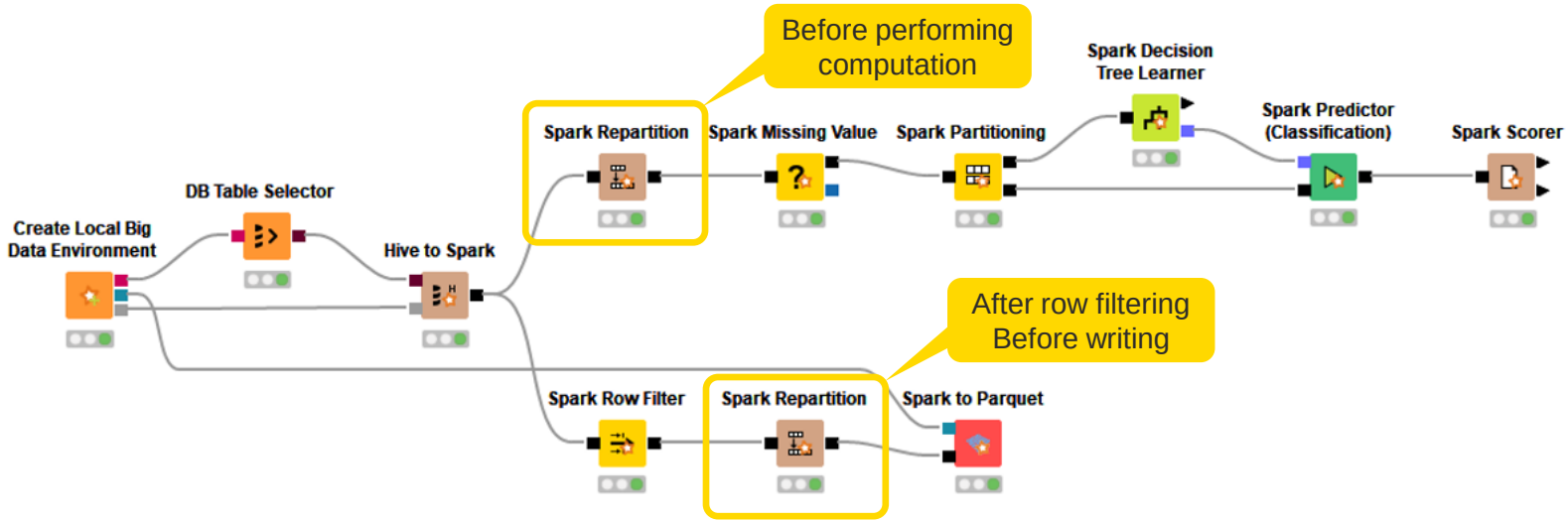


Spark Repartition

- First partitioned randomly
  - If original partition doesn’t work, repartition might be necessary
- Partitioning into same size groups increases the speed of parallel execution
  - No delayed completion of a stage due to an uneven distribution
- Optimal number of partitions
  - No idle/out of memory/slow executors due to too much data per partition
  - No unnecessary overhead due to too little data per partition
  - Via trial & error
  - At least a low multiple of the number of available executor cores in the Spark cluster
- Partitioning based on the values in a selected column
  - The column must be chosen carefully
  - A column with evenly distributed groups

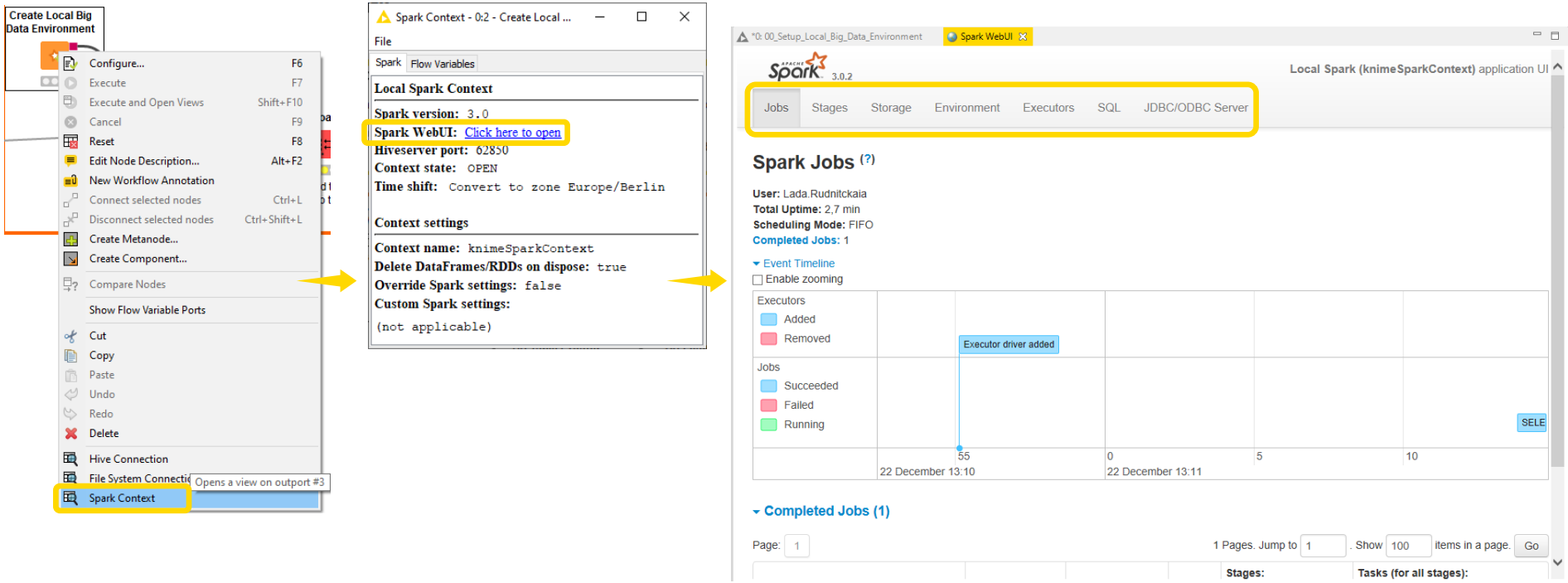
Spark Repartition Node

- Returns a Spark DataFrame with increased or decreased partition count
- When to use?
  - Before performing computation on a DataFrame (e.g. preprocessing or learning a model)
  - Before writing a DataFrame to storage to ensure fast writing and reading of the DataFrame
  - After operations that could cause uneven distribution, e.g., after row filtering
- Avoid expensive shuffling the data during repartition

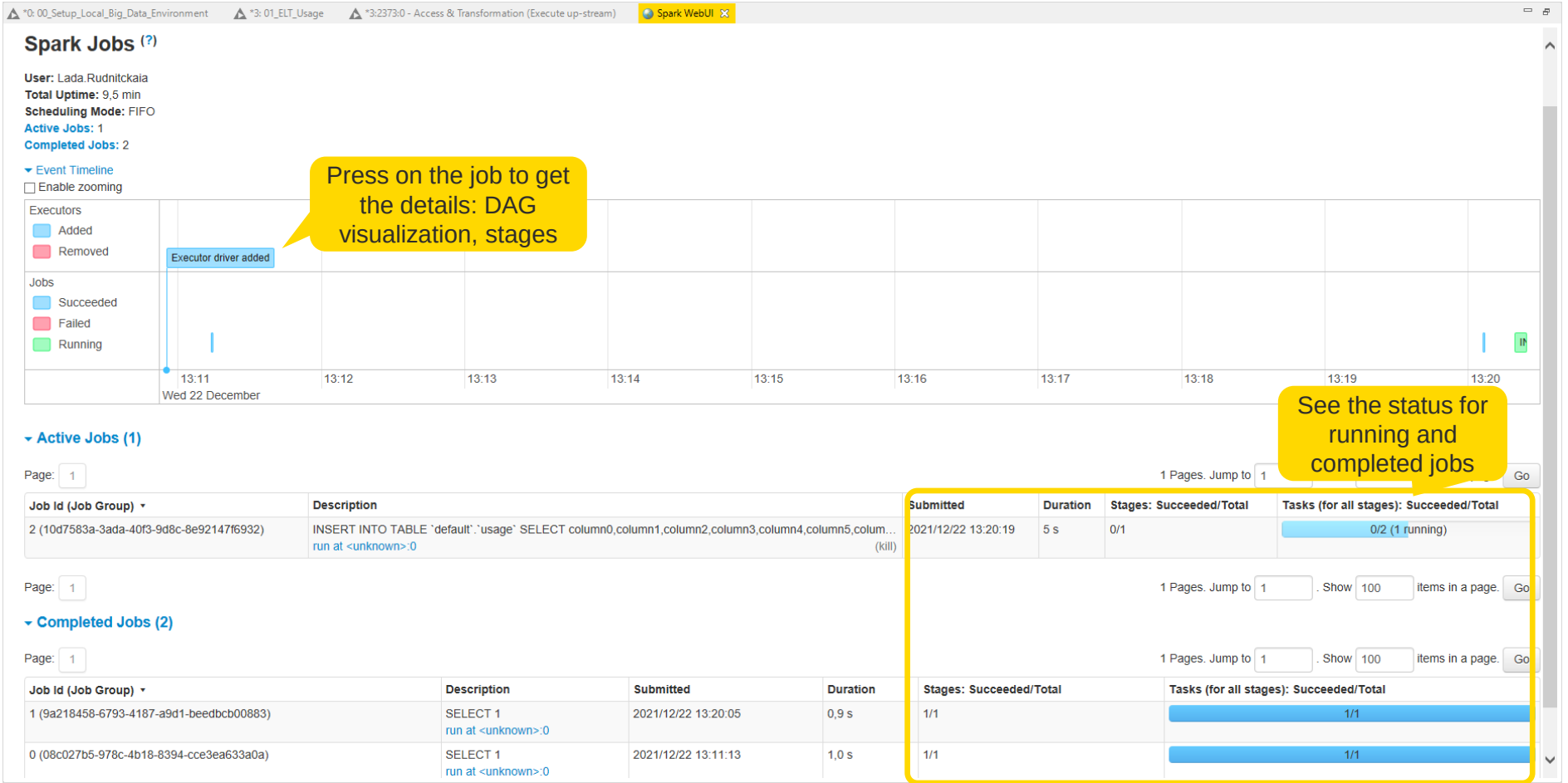


Spark WebUI

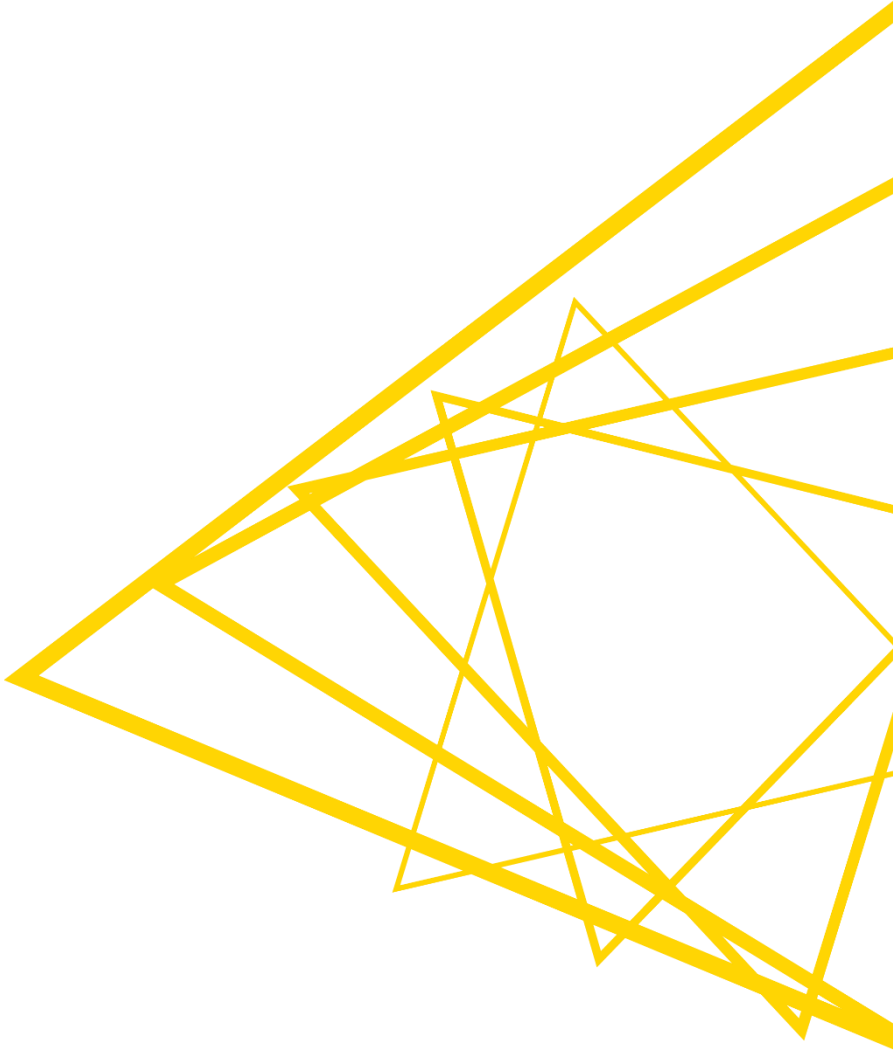
- Web interface of a running Spark application to monitor and inspect Spark job executions in a web browser
- Open as a tab in KNIME Analytics Platform



Spark Web UI – Jobs

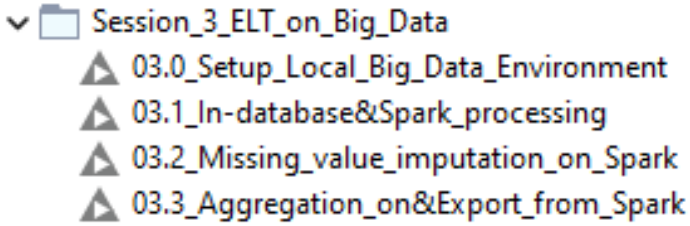


Demo



Exercises – Session 3

- Before starting the exercise **(skip if you performed these steps for session 1)**
  - Install local instance of PostgreSQL
  - Download the training workflows from the KNIME Community Hub
  - Install necessary extensions (open 00.1_Extensions_setup)
  - Execute workflow 00.2_Setup_PostgreSQL_Database
    - Use the credentials for your local instance of PostgreSQL
- Execute workflow 03.0_Setup_Local_Big_Data_Environment **(except this step)**

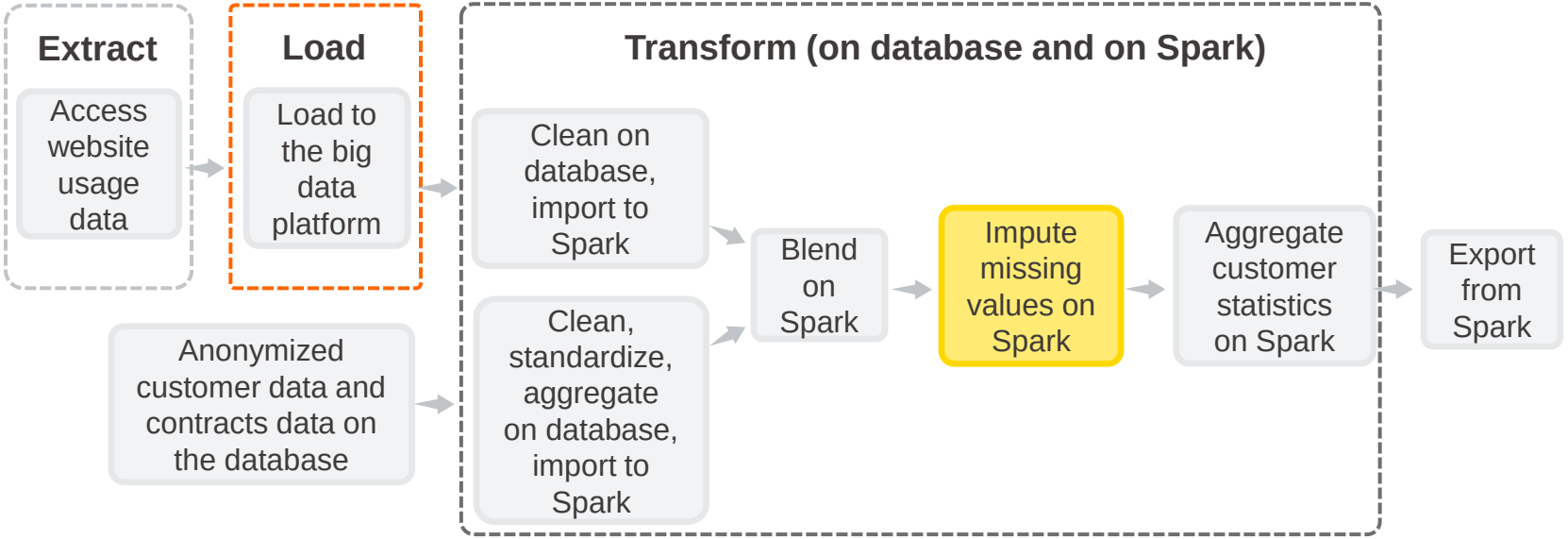


Exercise – 03.1_In-database&Spark_processing

- This exercise is the first step to build application “ELT on Usage data”
  - 1 Access the data, transform on database, import into Spark
    - Use the credentials for your local instance of PostgreSQL
  - Find detailed instructions in the workflow

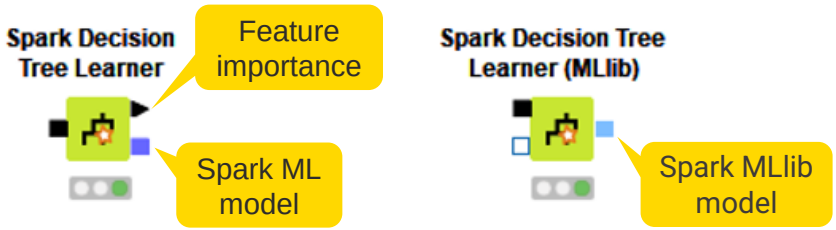
Machine Learning with Spark

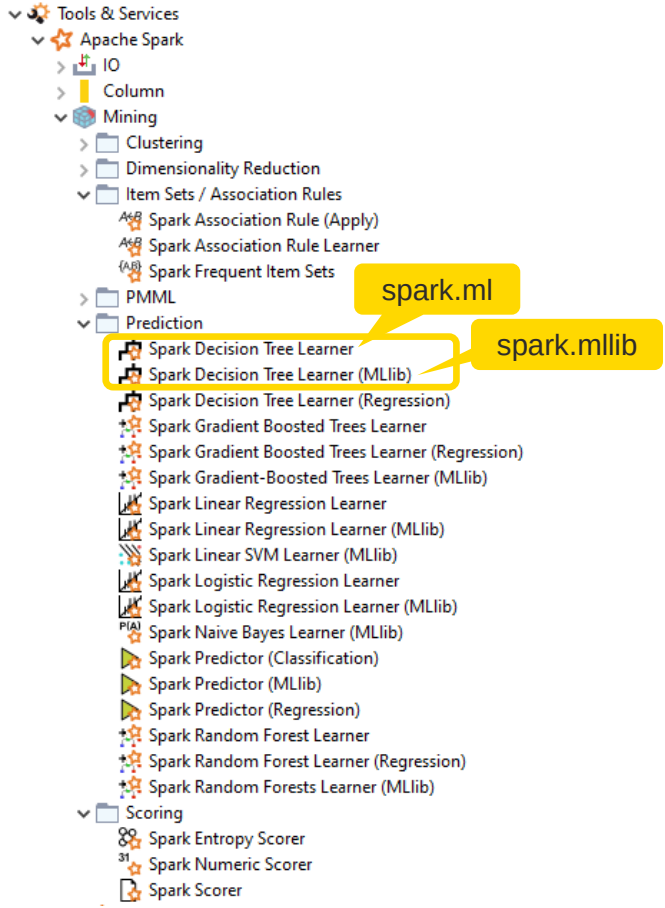
Machine Learning with Spark



Machine Learning Integrations

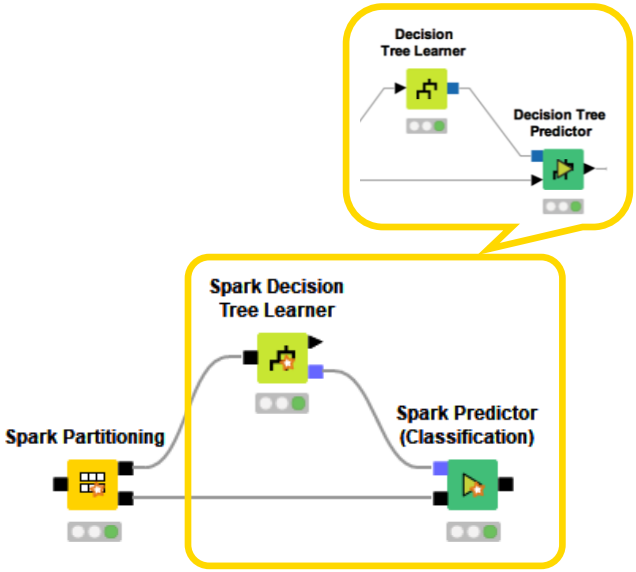
- KNIME Extension for Apache Spark includes nodes to train various machine learning models in Spark
- Implementation based on both spark.mllib and spark.ml packages
  - spark.ml is a primary package – use it always when possible
- spark.ml based nodes have improved functionality
  - accept categorical features, provide training statistics, e.g., feature importance, provide conditional class probabilities

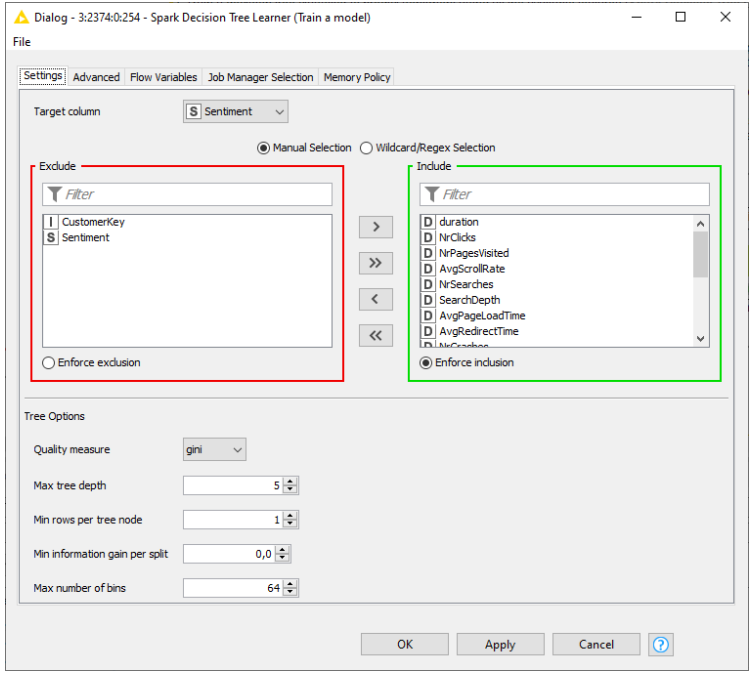




Spark ML Integration: Familiar Usage Model

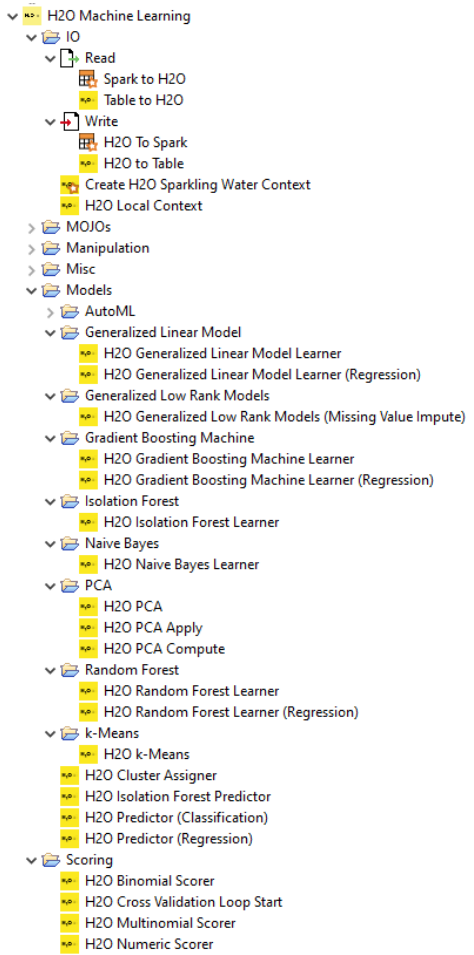
- Usage model and dialogs like existing nodes
- No coding required
- Various algorithms for classification, regression and clustering supported





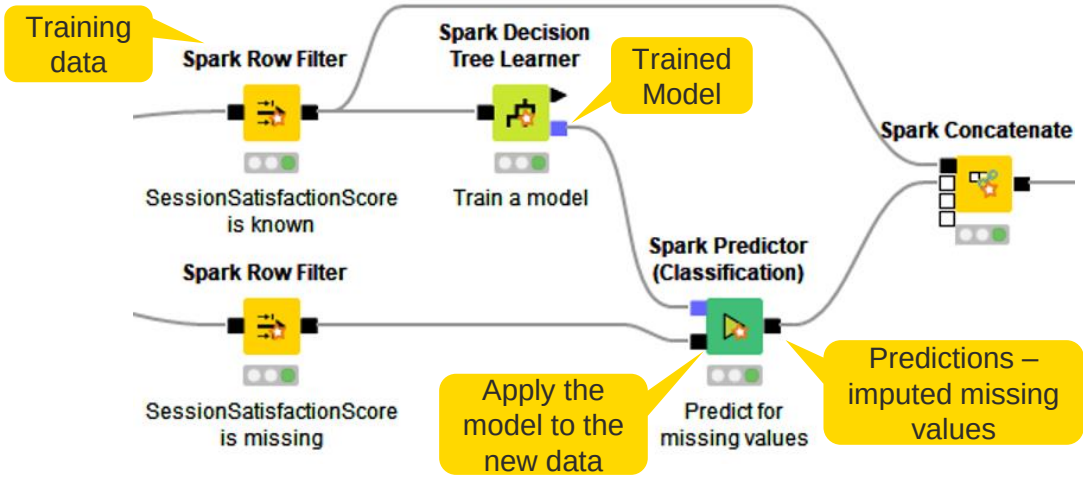
H2O on Spark

- H2O: Open source, focus on scalability and performance
- Sparkling Water = H2O on Spark
- KNIME H2O integrations
  - KNIME H2O Machine Learning Integration
  - KNIME H2O Sparkling Water Integration
- Supports many different models
  - Generalized Linear Model
  - Gradient Boosting Machine
  - Random Forest
  - k-Means, PCA, Naive Bayes, etc. and more to come!
- Includes support for MOJO model objects for deployment

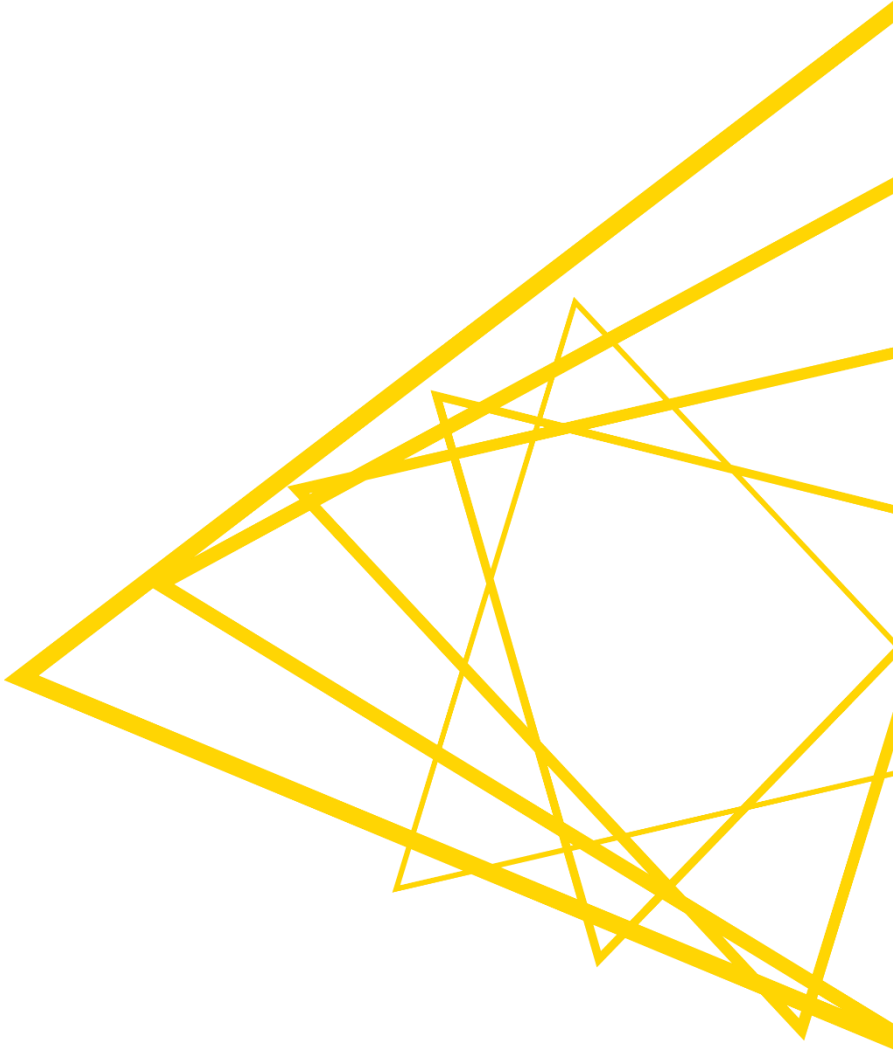


Missing Values Imputation

- Separate data rows with SessionSatisfactionScore present and missing
- Train a machine learning classifier model to predict SessionSatisfactionScore (only on data rows with SessionSatisfactionScore)
- Apply the trained model to predict SessionSatisfactionScore where it is missing
- Update original data set with new predicted SessionSatisfactionScore values – **impute missing values**



Best Practices: Spark Performance Optimization

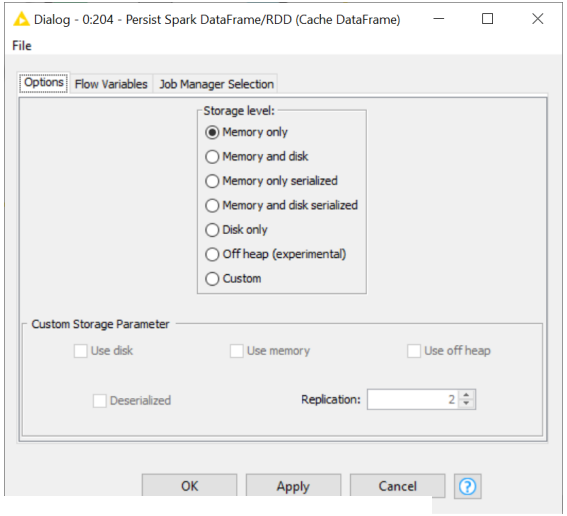


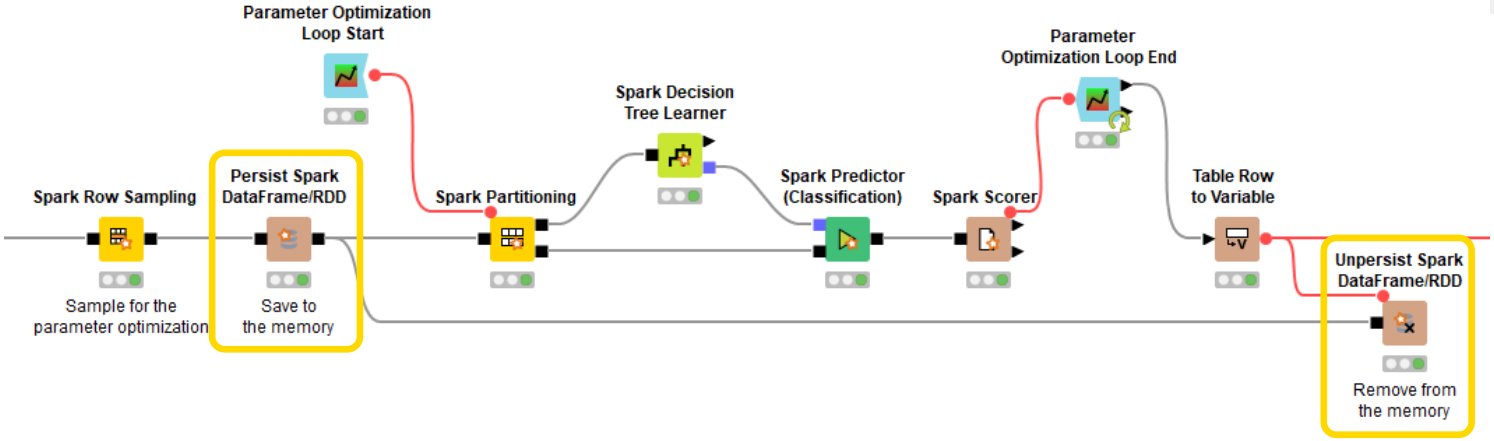
Spark DataFrame Persistence

- Spark DataFrames
  - are created and exist in memory on executors
  - Once a DataFrame is transformed and is no longer needed, it will be removed permanently
- Cache or persist the DataFrame
  - The DataFrame will be kept in memory on all of the nodes of the cluster where it is computed after the 1st action is called on it
  - Useful when more than one action is required, e.g., within a loop body, to avoid an entire reevaluation
  - The cashed DataFrame can be seen in the Storage tab in the Spark application UI

Persist & Unpersist Spark DataFrame/RDD Nodes

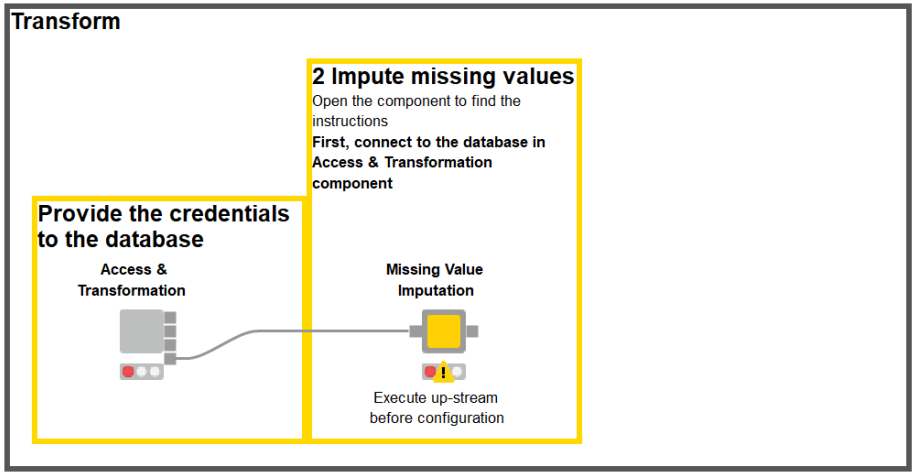
- Persist (cache) the Spark DataFrame using the specified storage level
- Clean the storage when the operation is over (e.g., loop)



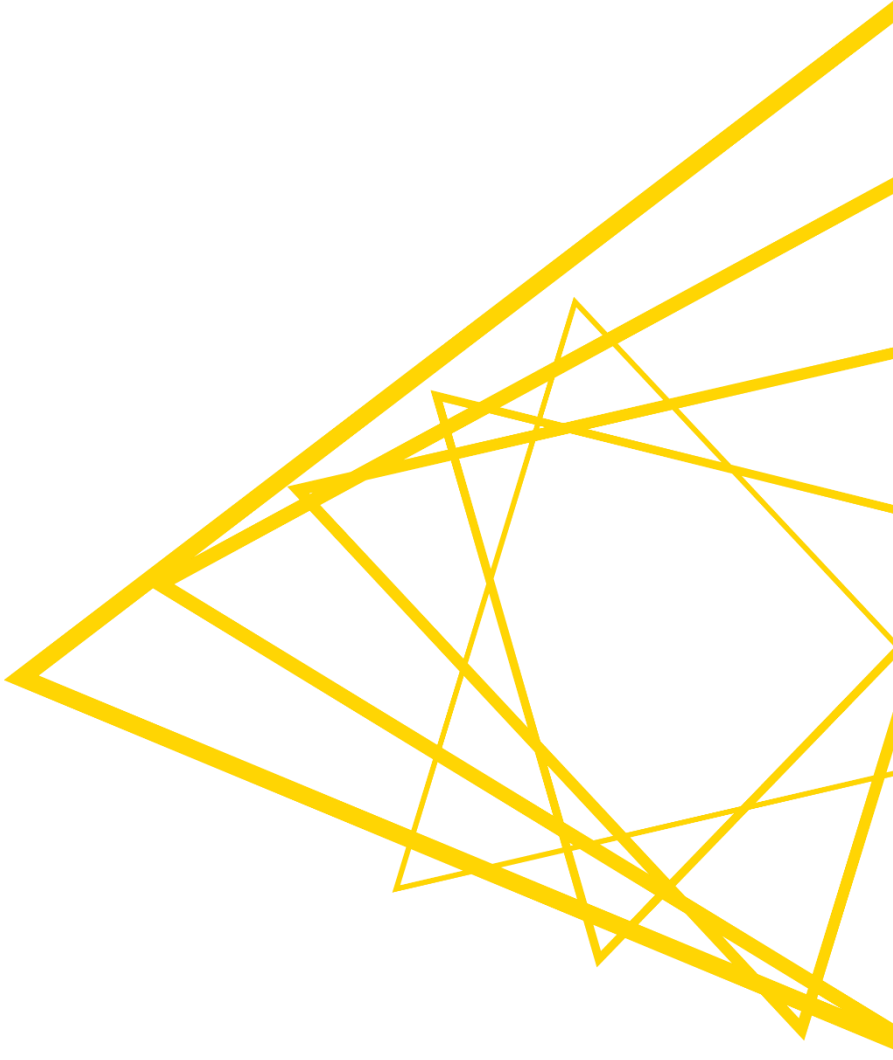


Exercise – 03.2_Missing_value_imputation_on_Spark

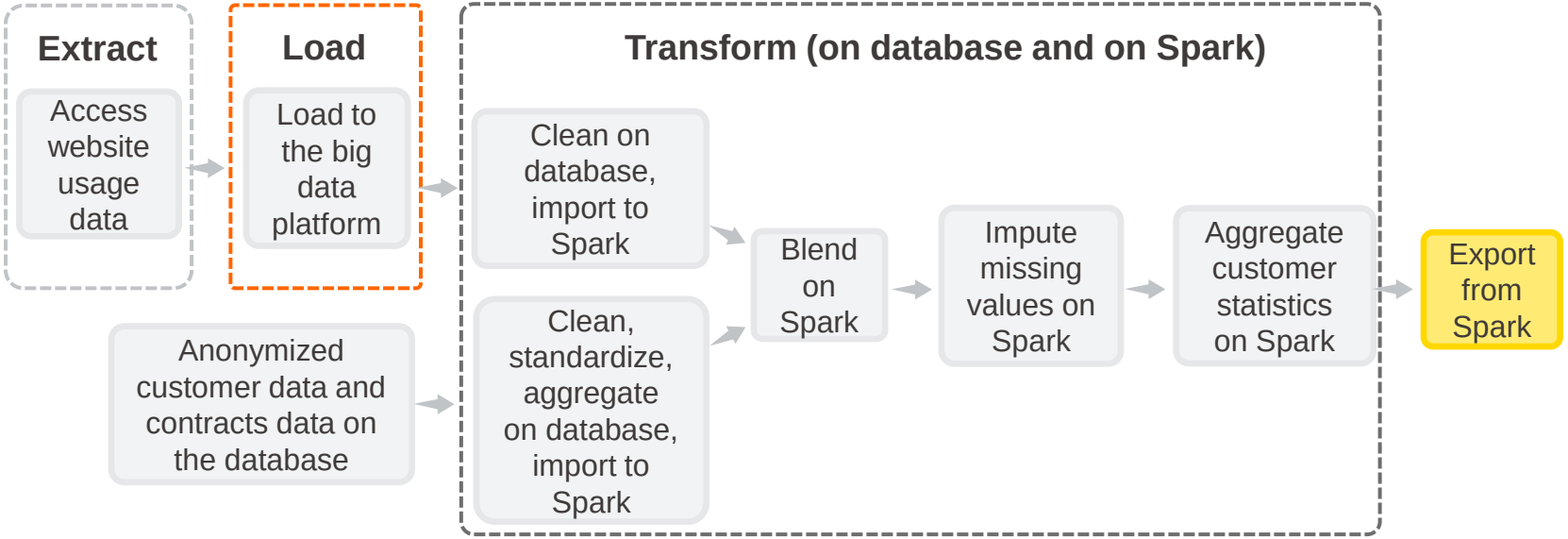
- This exercise is the second step to build application “ELT on Usage data”
  - The solution to the previous exercises is already in the workflow
  - 2 Impute missing values
  - Find detailed instructions in the workflow



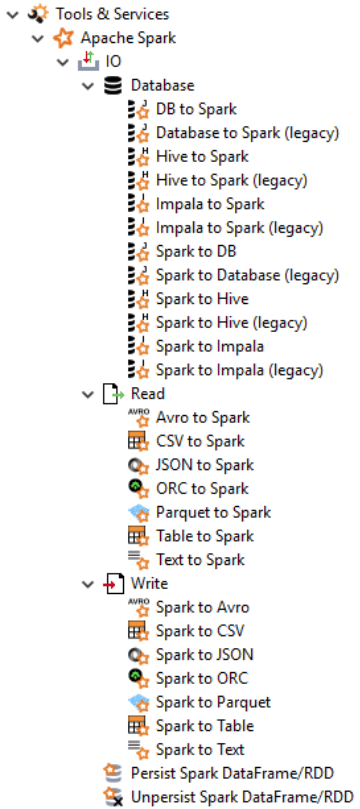
Export Data from Spark



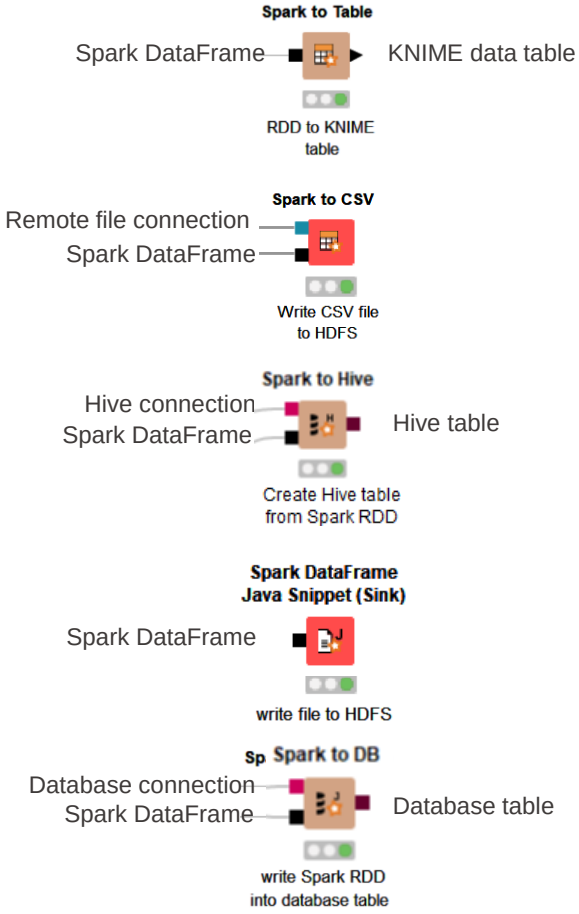
Export Data from Spark



Export Data from Spark

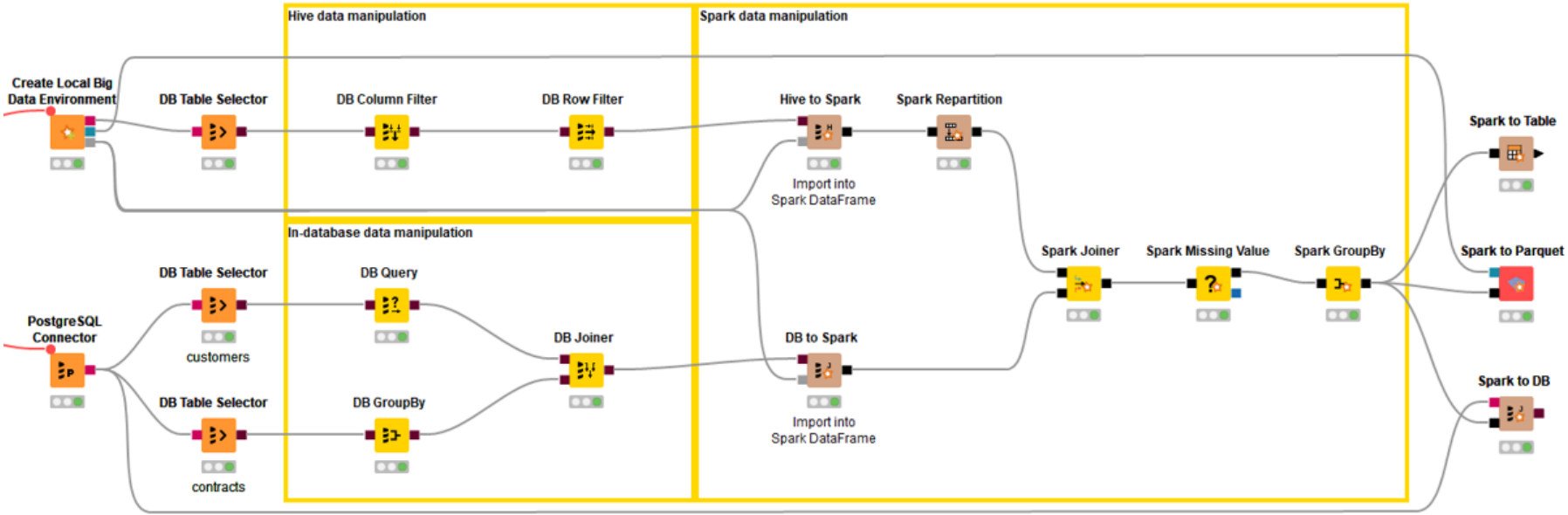


- To KNIME
- To CSV file in HDFS
- To Hive
- To Other Storages
- To Database

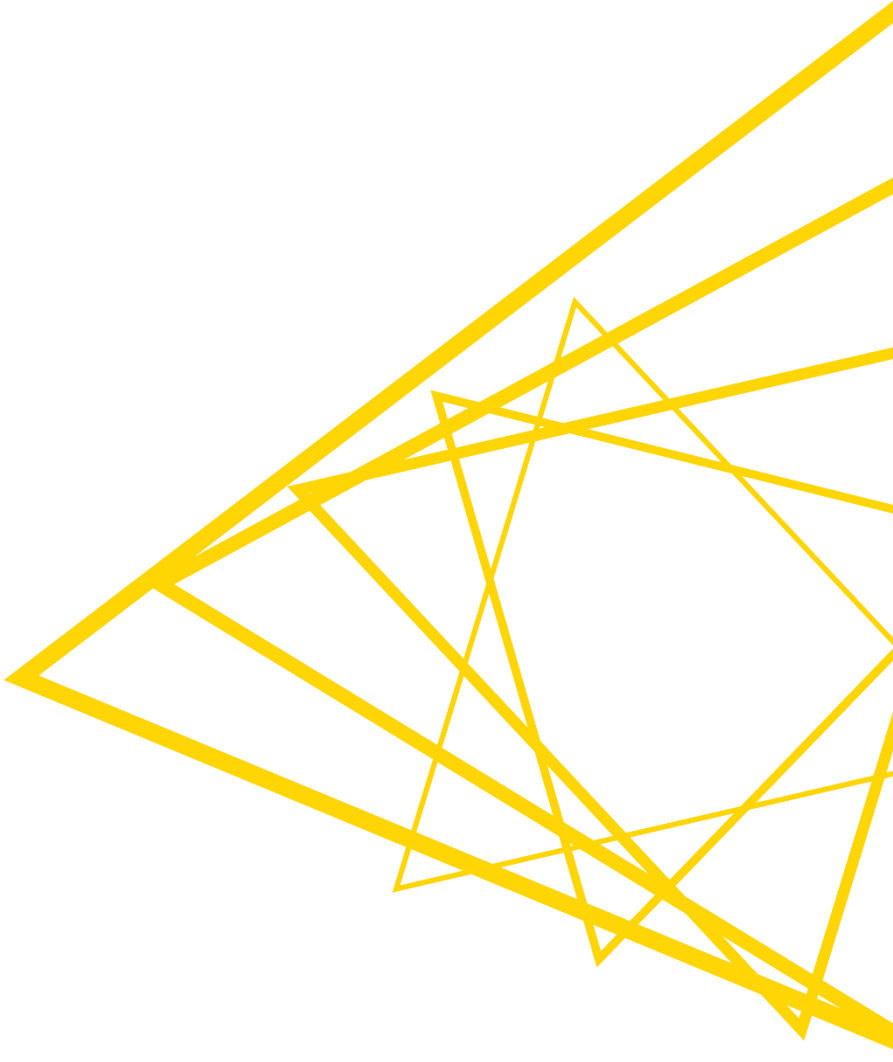


Mix & Match

- Thanks to the transferring nodes (Hive to Spark, Spark to Hive, Table to Spark, Spark to Table, DB to Spark, Spark to DB) you can mix and match processing in KNIME, on database, and on Spark

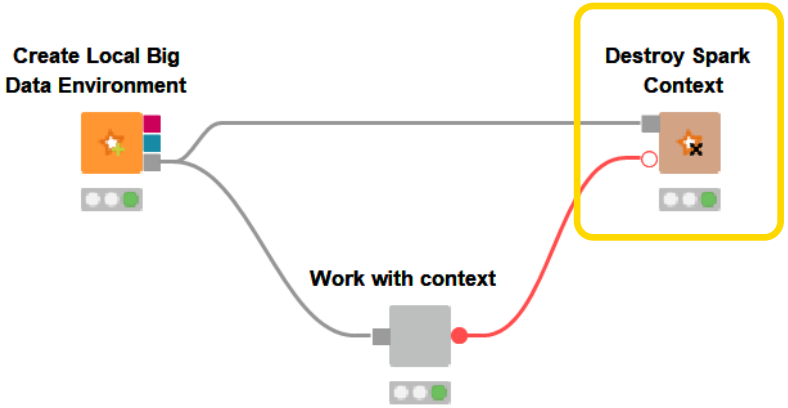


Best Practices: Spark Performance Optimization

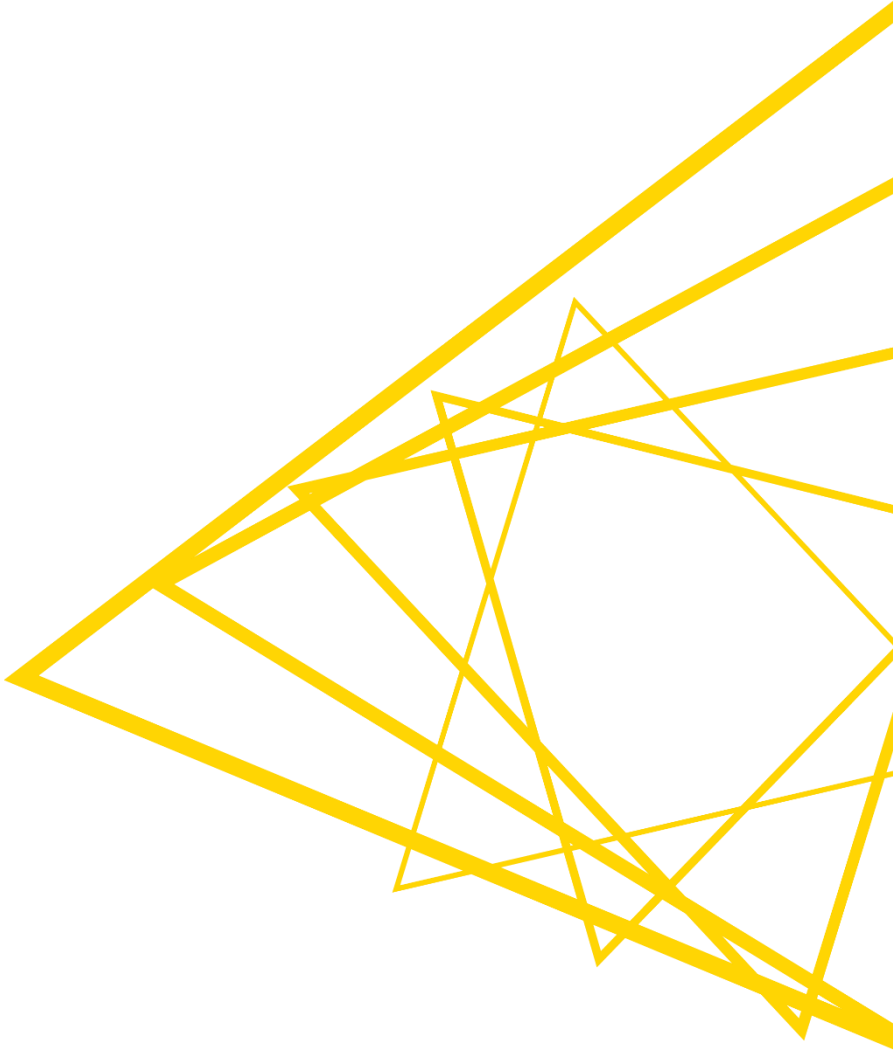


Recap: Destroying Spark Context

- Free up the resources Spark Context allocated on the cluster

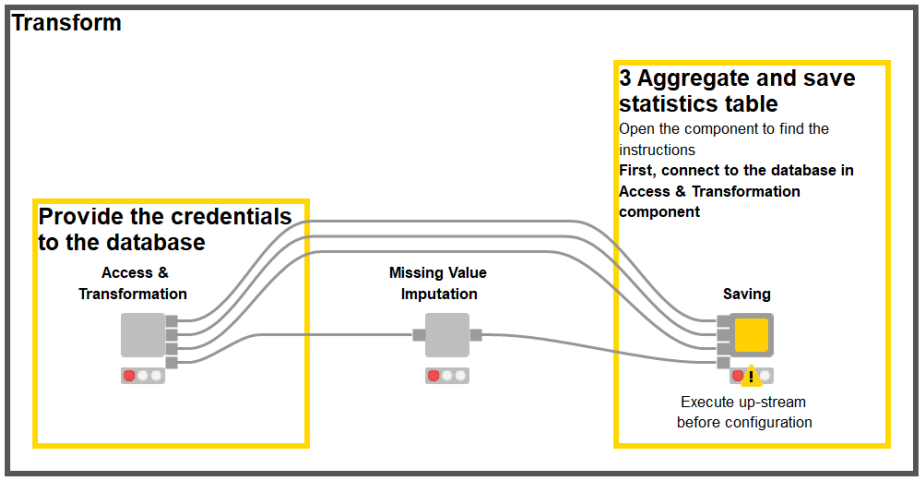


Demo



Exercise – 03.3_Aggregation_on&Export_from_Spark

- This exercise is the third step to build application “ELT on Usage data”
  - The solution to the previous exercises is already in the workflow
  - 3 Aggregate and save statistics table
  - Find detailed instructions in the workflow



Session 3: Summary

Now you should be able to:

- Integrate Hadoop applications, such as Hive, HDFS, and Spark into KNIME Analytics Platform
- Process relatively large data on a database server and on Spark
- Train and apply a machine learning model on Spark
- Impute missing values on Spark
- Build the application that aggregates website usage data, personal data, and contract data into a customer statistics table

Session 4: Cloud and Big Data connectivity, Orchestration

Session 4: Learning Outcomes

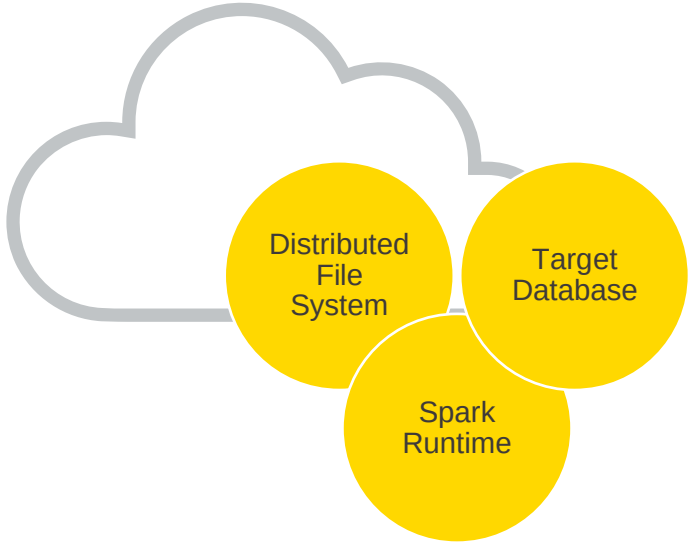
At the end of this session, you will be able to:

- Integrate cloud Hadoop Applications in KNIME
- Orchestrate modular workflows from a caller workflow
- Build an application that triggers and orchestrates the applications from previous sessions

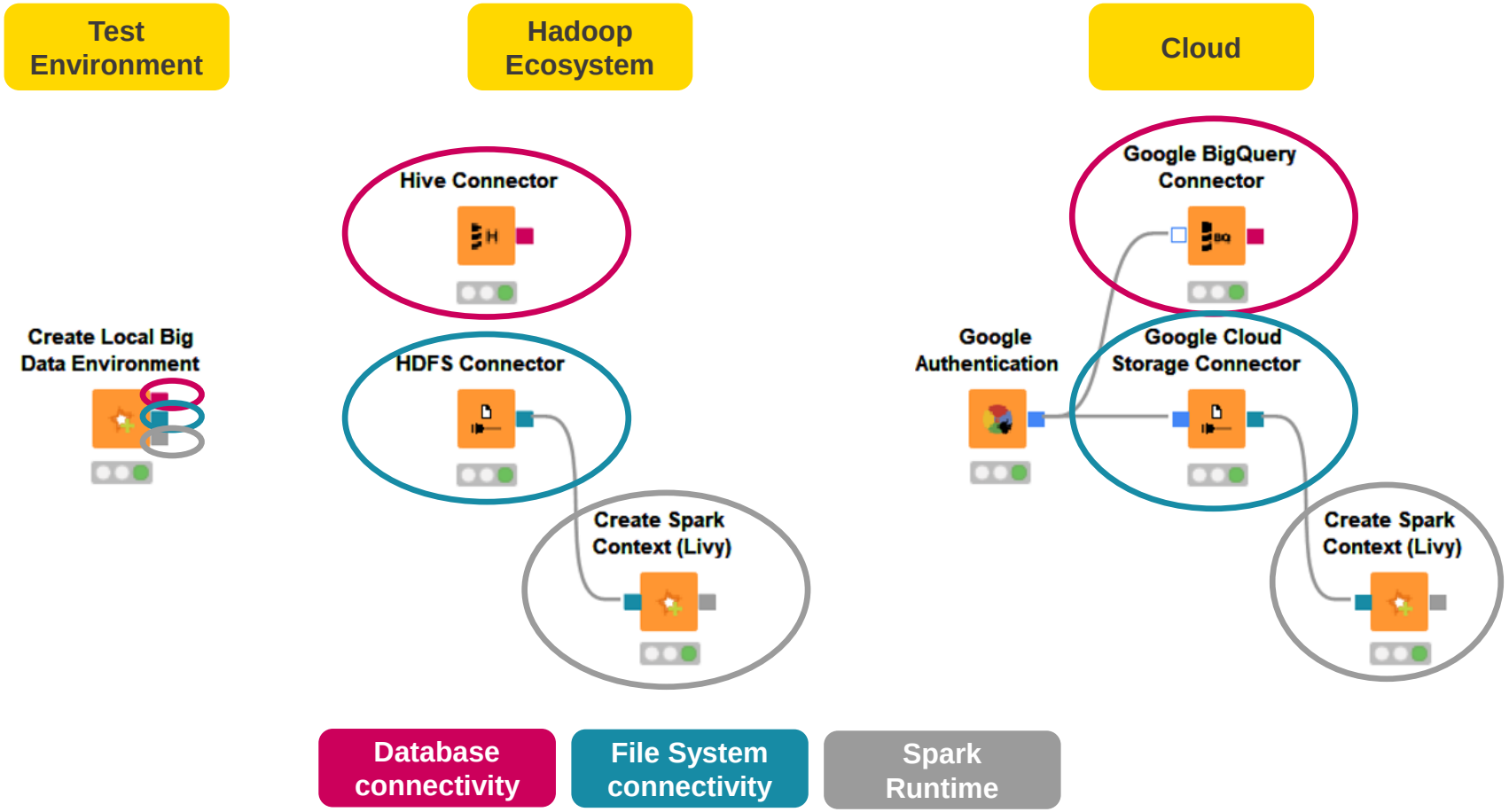
Cloud & Big Data Connectivity

Running Hadoop Applications in a Cloud

- Clusters managed by the cloud provider run Hadoop and Hadoop ecosystem applications
- Resources are offered by a cloud provider
- Everything is on one cloud platform
  - Distributed file system / Data lake / Data storage
  - Database for querying
  - Spark runtime for in-memory processing

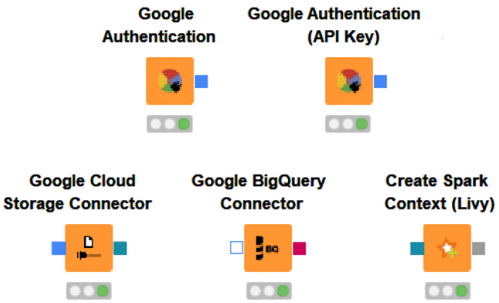


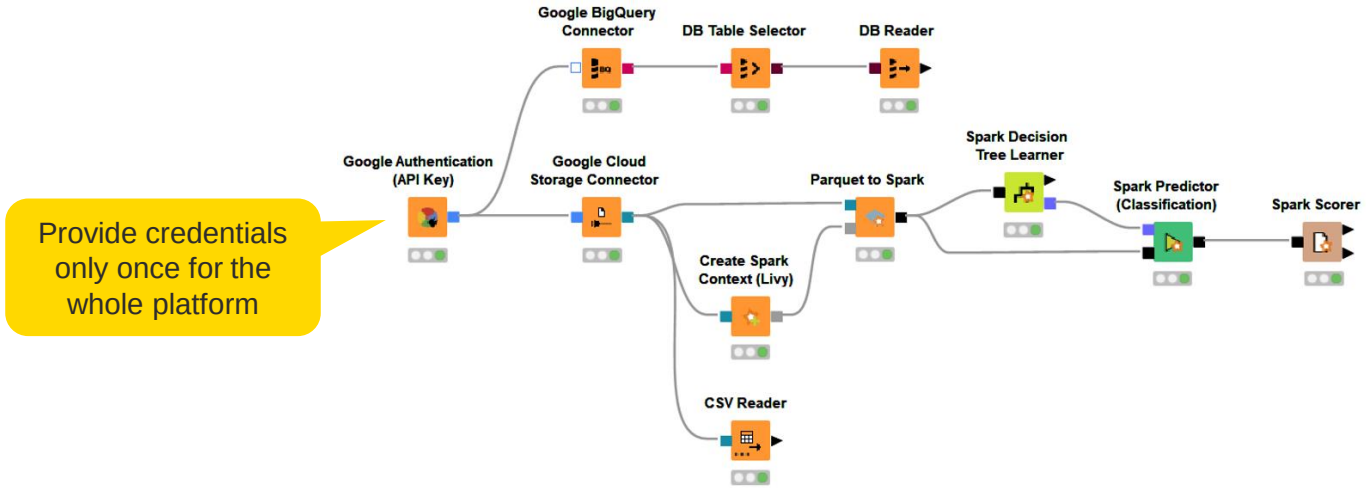
Test Environment vs. Hadoop Ecosystem vs. Cloud



Google Cloud Platform

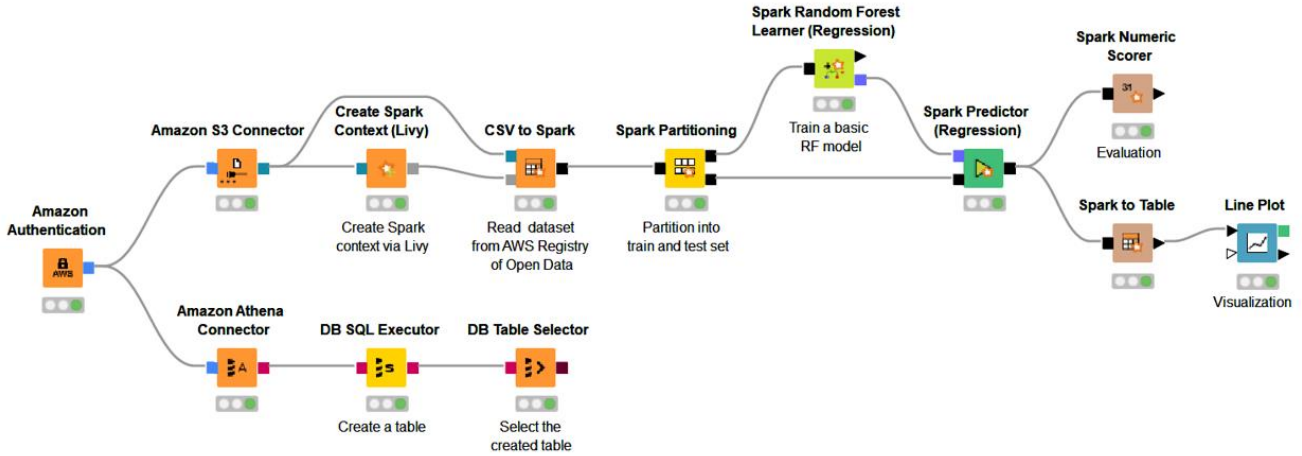
- Connectivity to
  - Data storage: Google Cloud Storage
  - Database: Google Big Query, Hive on Google Dataproc
  - Spark Runtime: Google Cloud Dataproc
- KNIME Google Cloud Integration User Guide
  - Dataproc cluster setup
  - Connect to Dataproc cluster with Livy





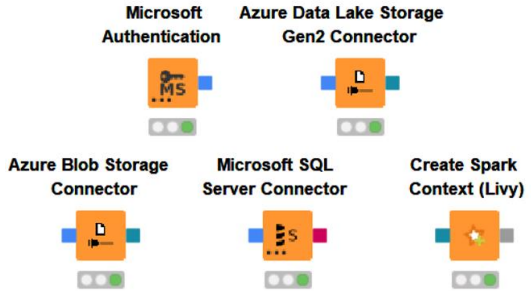
Amazon Web Services (AWS)

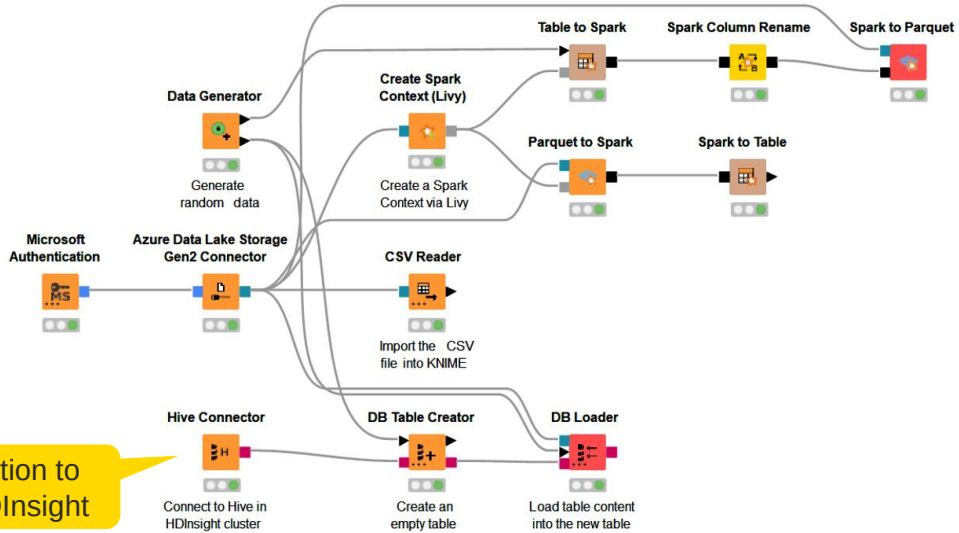
- Connectivity to
  - Data storage: Amazon S3
  - Database: Amazon Athena, Hive on EMR
  - Spark Runtime: Amazon EMR (Elastic MapReduce)
- KNIME Amazon Web Services Integration User Guide
  - Create an Amazon EMR cluster
  - Connect to EMR cluster with Livy



Microsoft Azure

- Connectivity to
  - Data storage: Azure Data Lake Storage Gen2, Azure Blob Storage
  - Database: Hive on Azure HDInsight, Azure SQL Database
  - Spark Runtime: Azure HDInsight
- KNIME Azure Integration User Guide
  - Azure HDInsight cluster setup
  - Connect to HDInsight cluste

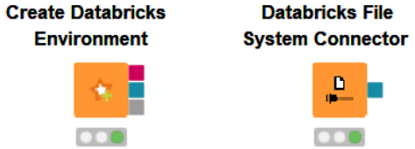




Establish a connection to Apache Hive on HDInsight

Databricks

- Connectivity to
  - Data storage: Databricks File System (DBFS)
  - Database: Databricks Database, Databricks Delta
  - Spark Runtime: Databricks Cluster
- KNIME Databricks Integration User Guide
  - Create a Databricks cluster
  - Connect to Databricks

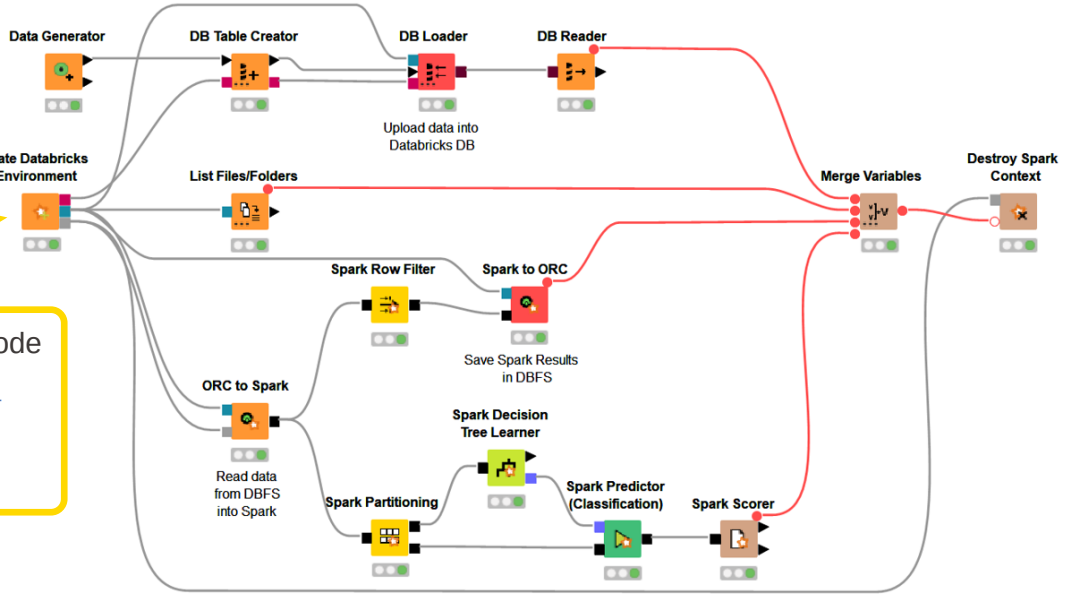


One node creates the whole environment: file system, database, and Spark context

Connect directly to DBFS without starting a cluster

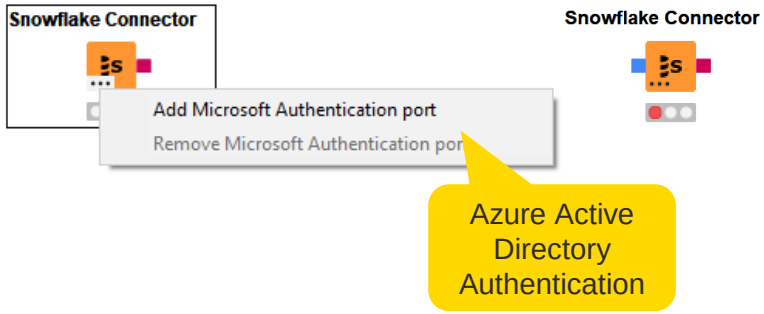
Alternative node

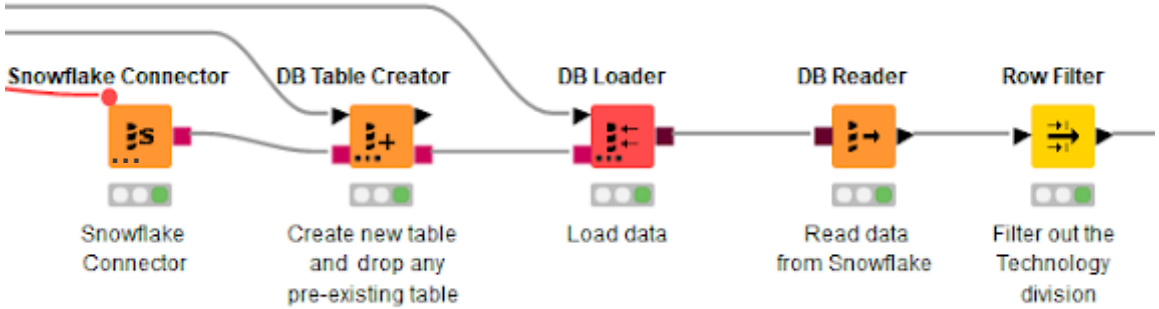
Databricks File System Connector



Snowflake

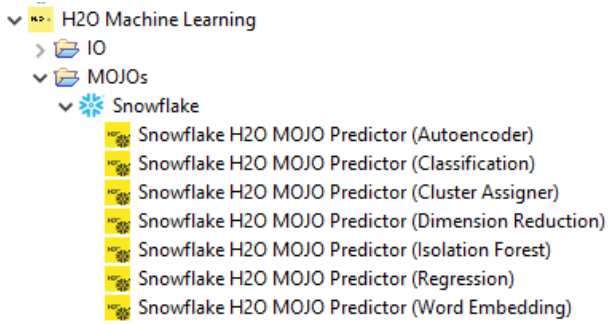
- Snowflake Cloud Data Platform
  - In-cloud data warehouse
  - Can be deployed on Azure, AWS, or GCP
- KNIME Snowflake Extension Guide

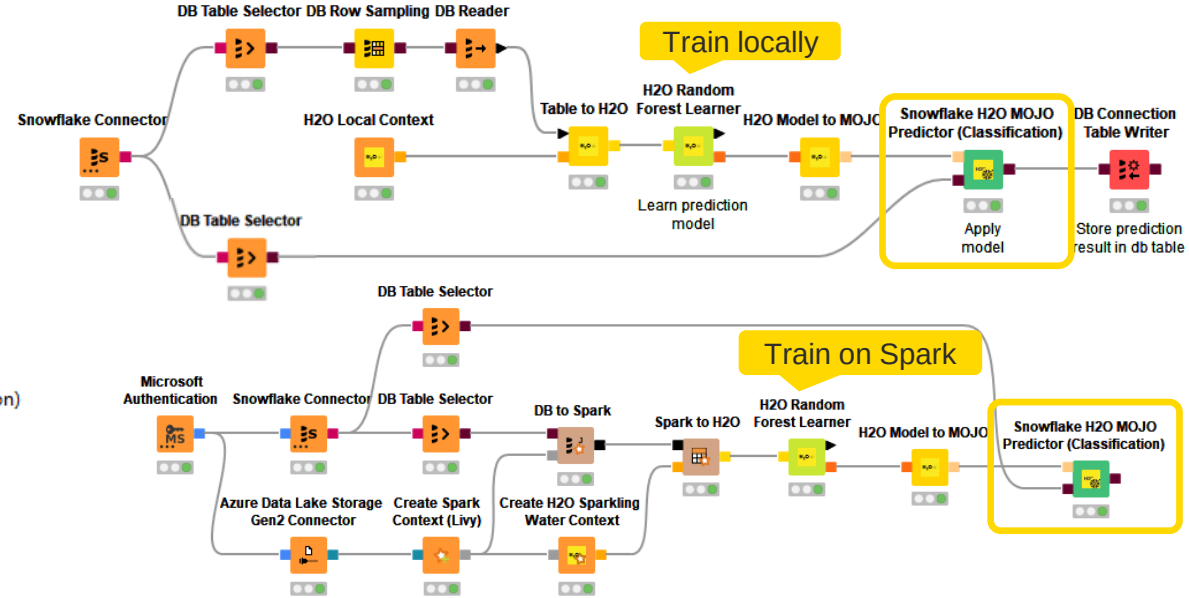




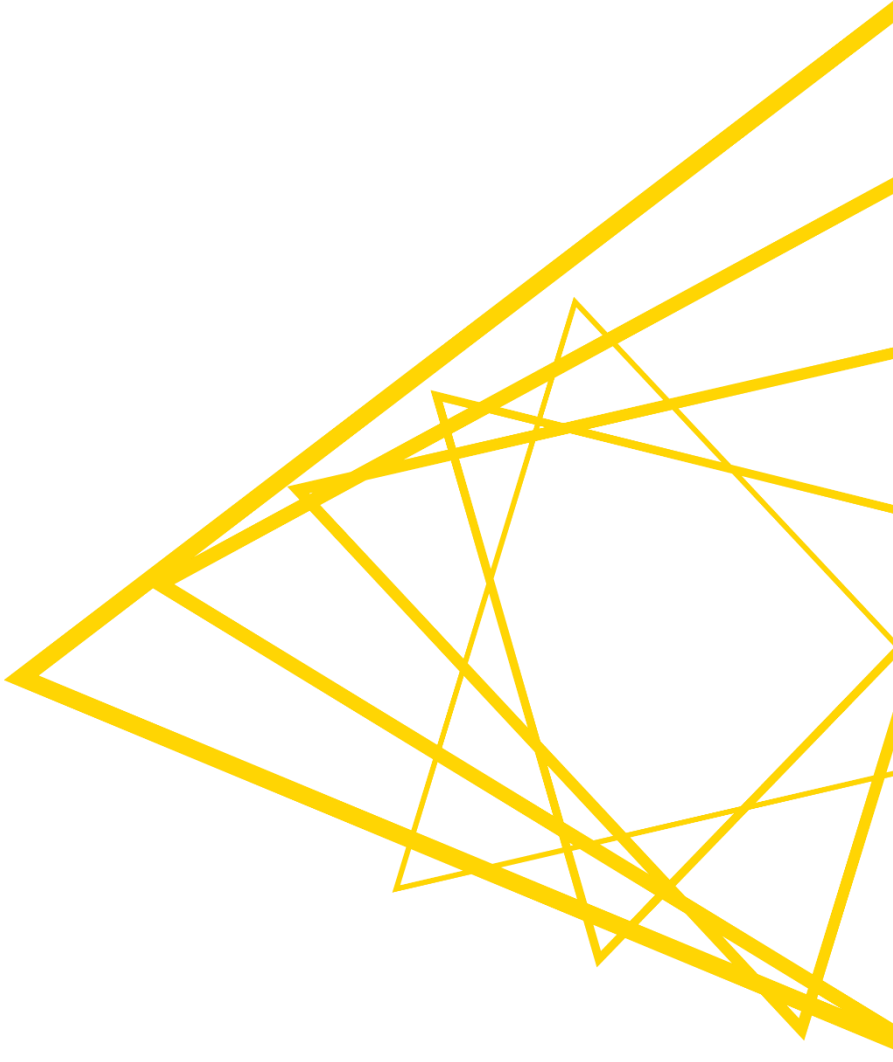
Snowflake H2O Machine Learning

- Data prediction within Snowflake without moving the data out of Snowflake
  - Prediction result is stored in a Snowflake table
- Supported H2O MOJO models learned via
  - KNIME H2O Machine Learning Integration
  - KNIME H2O Sparkling Water Integration (within a Spark runtime)
- Different predictors available in KNIME H2O Snowflake Integration



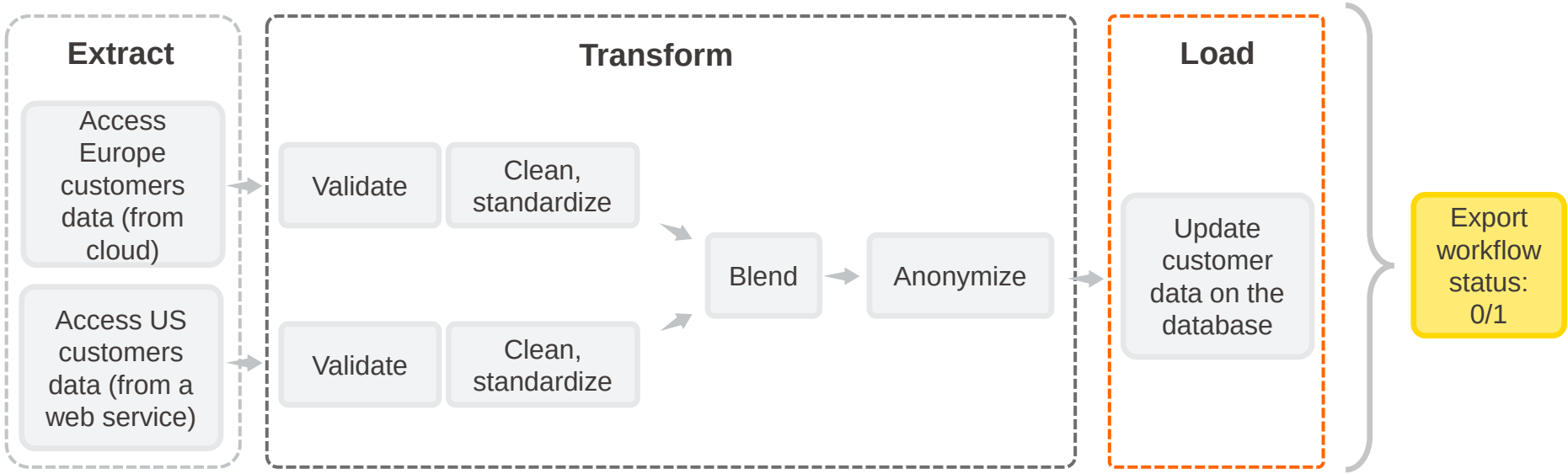


Demo

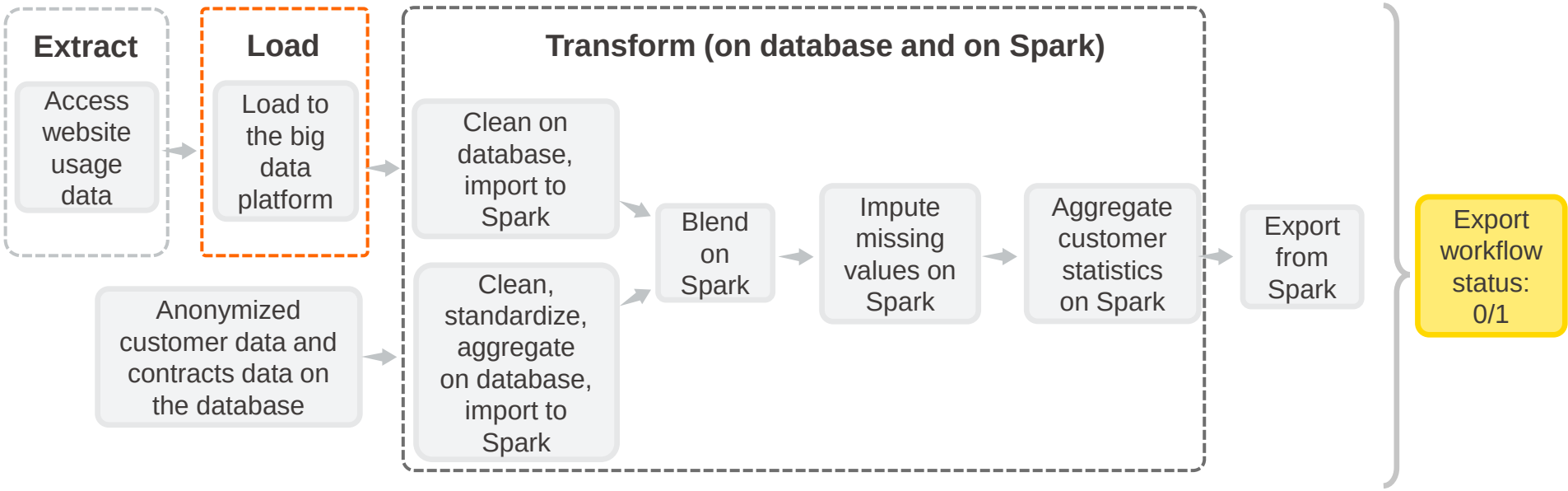


Today‘s example: Orchestration

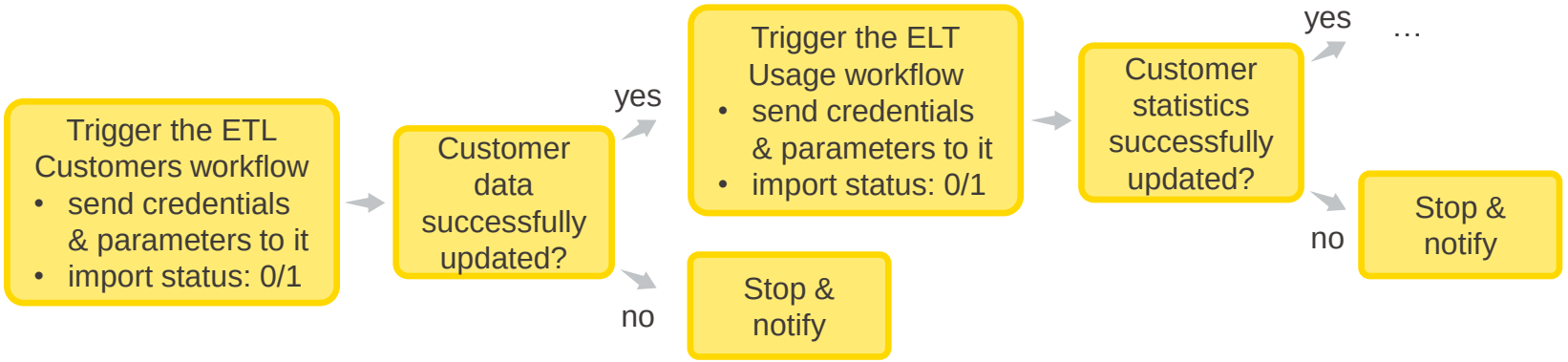
Recap: ETL on Customers Data



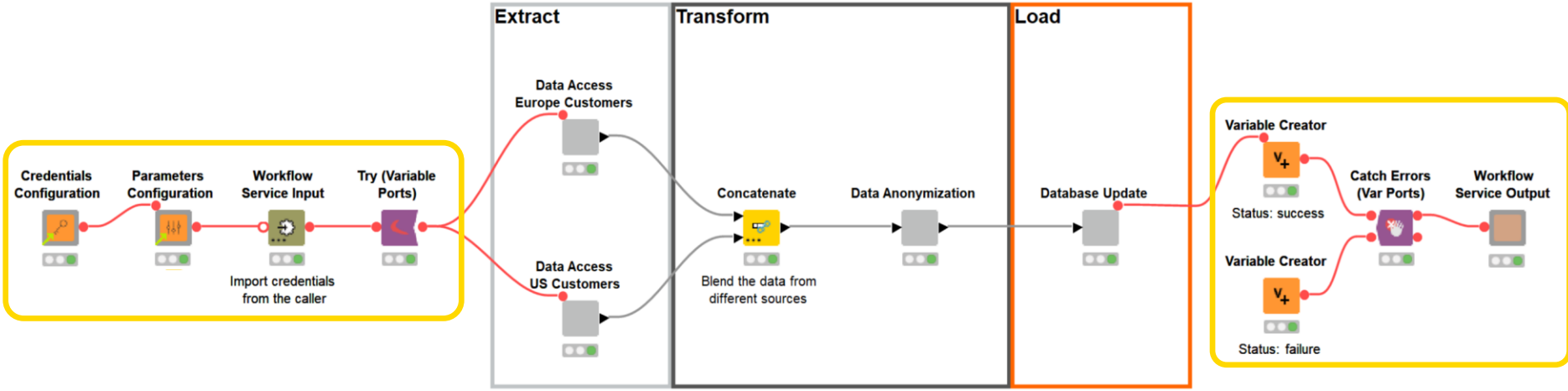
Recap: ELT on Usage Data



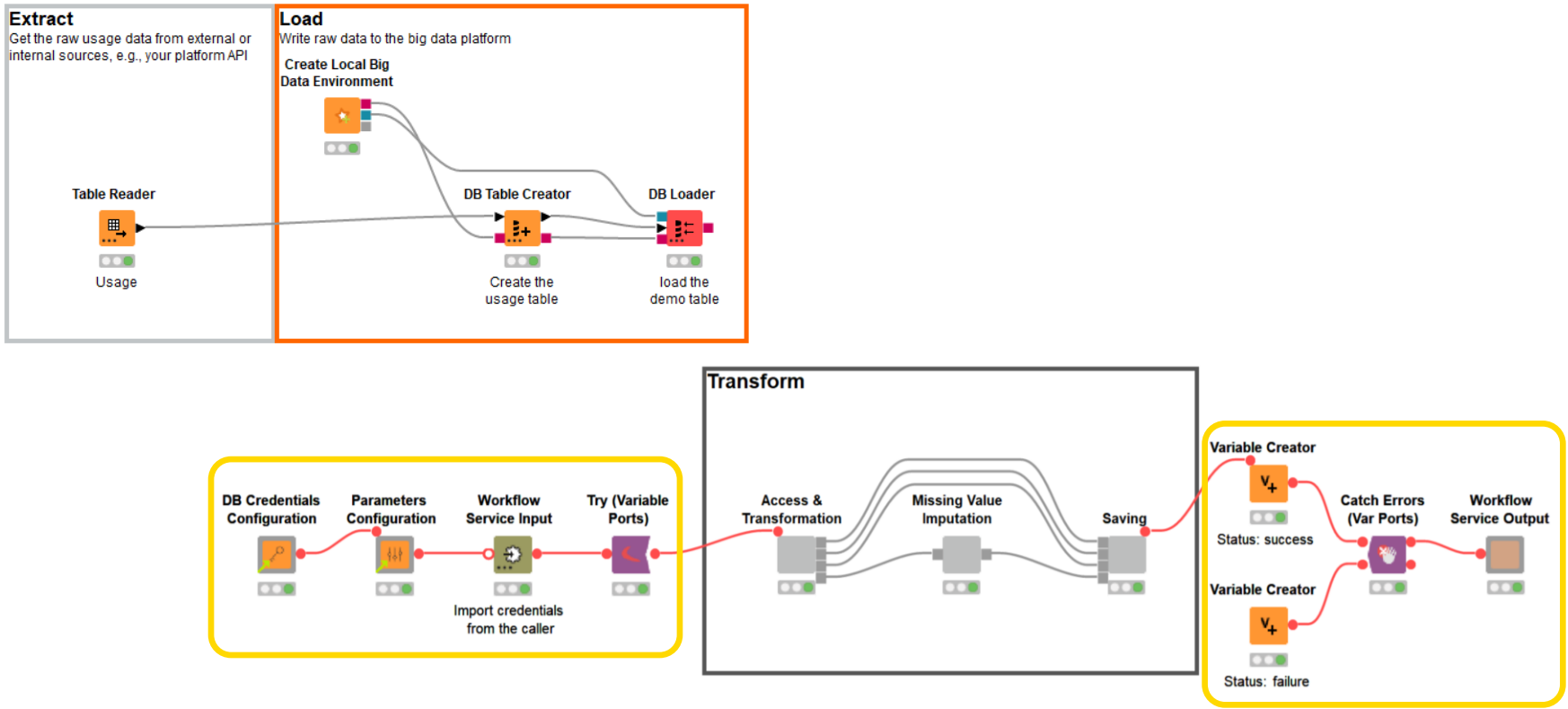
Session 4: Orchestration



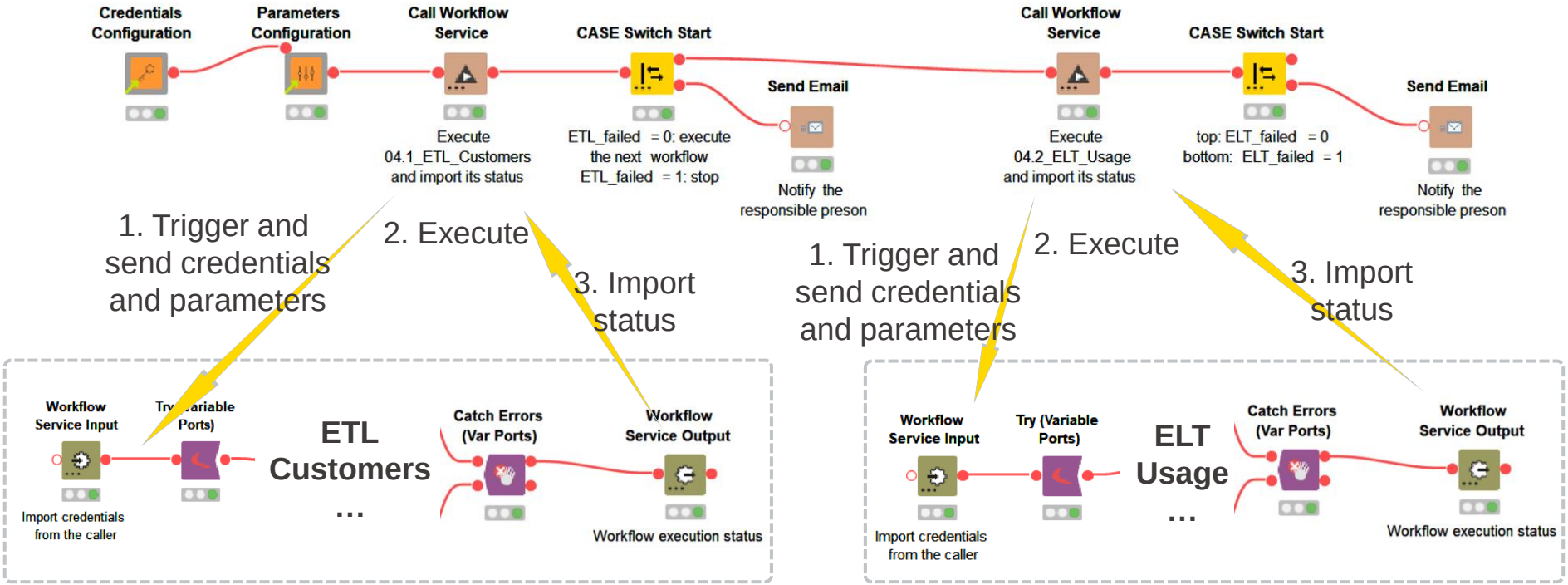
Recap: ETL on Customers Data



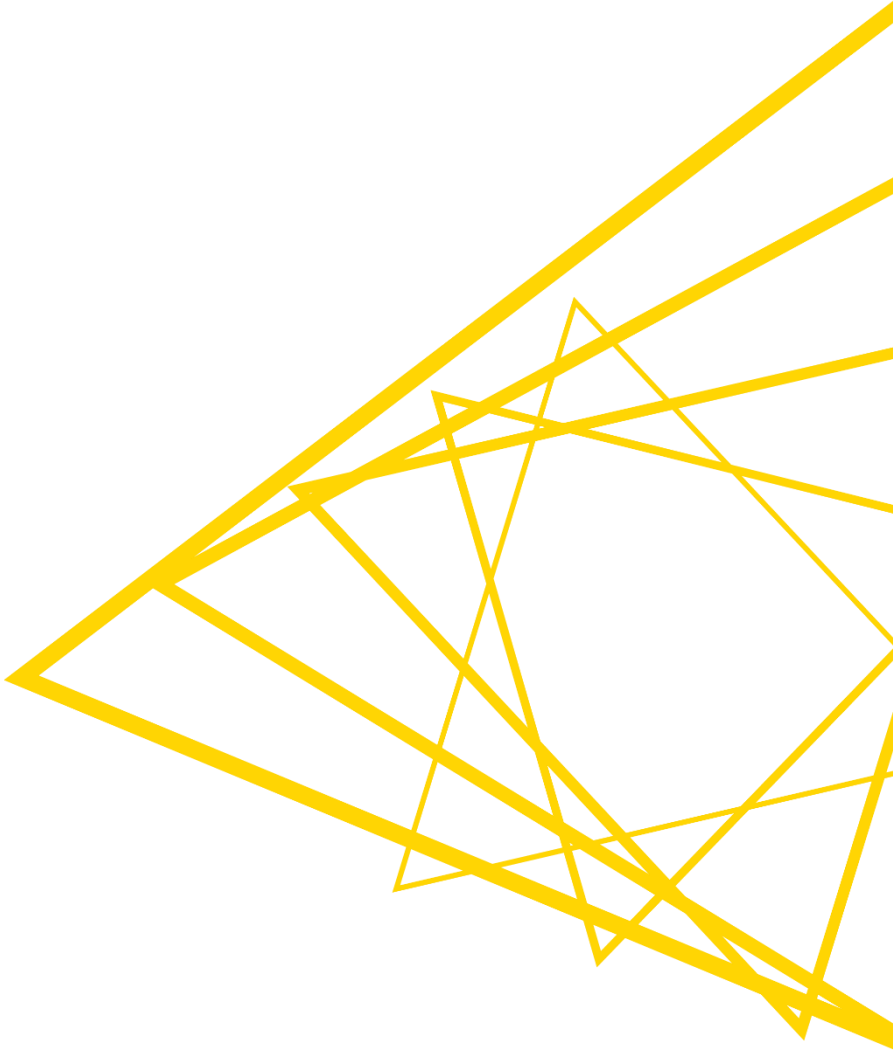
Recap: ELT on Usage Data



Today's Example: Orchestration

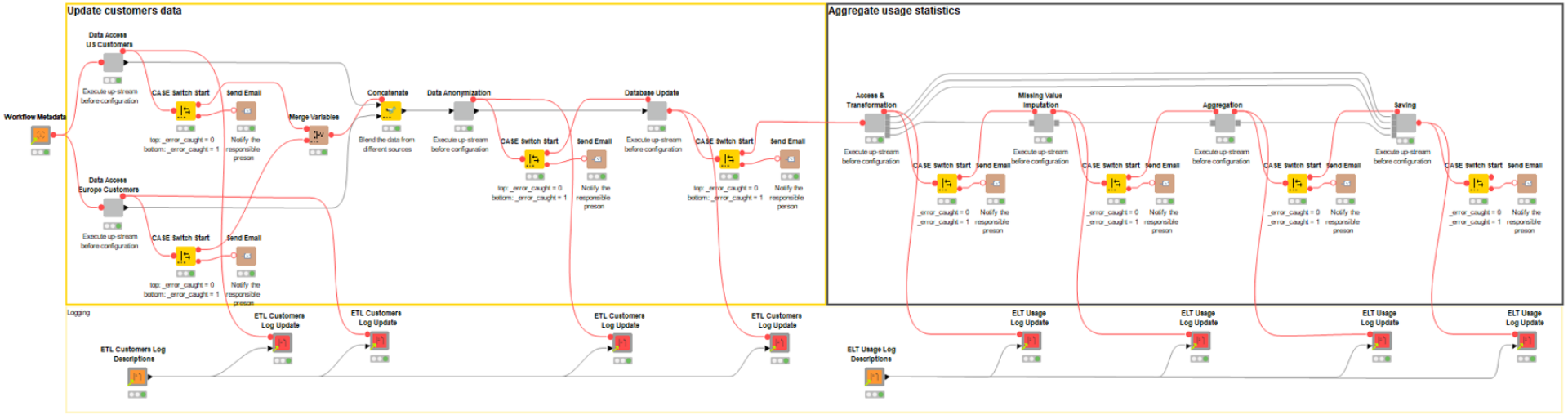


Orchestration



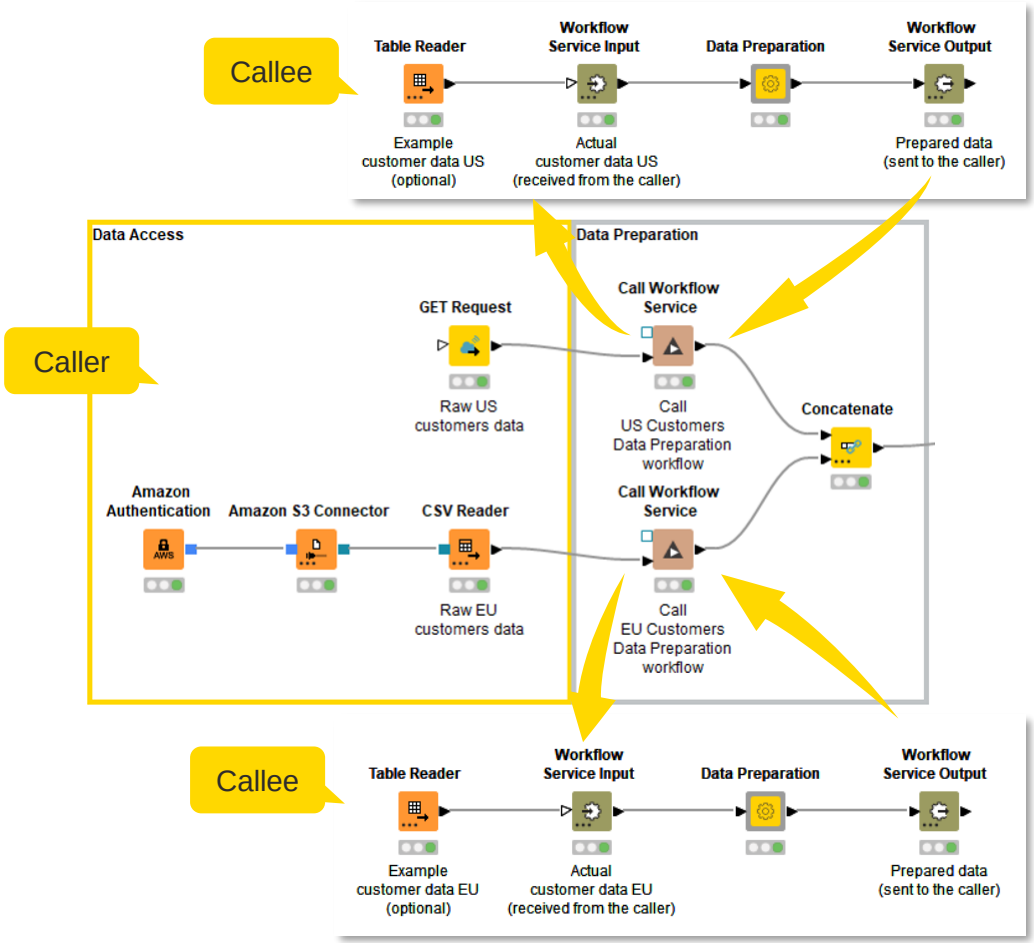
Motivation behind Orchestration

- When the workflow grows…
  - Many different processes and purposes in one workflow
  - Difficult to maintain
  - Difficult to test
  - Loading slows down



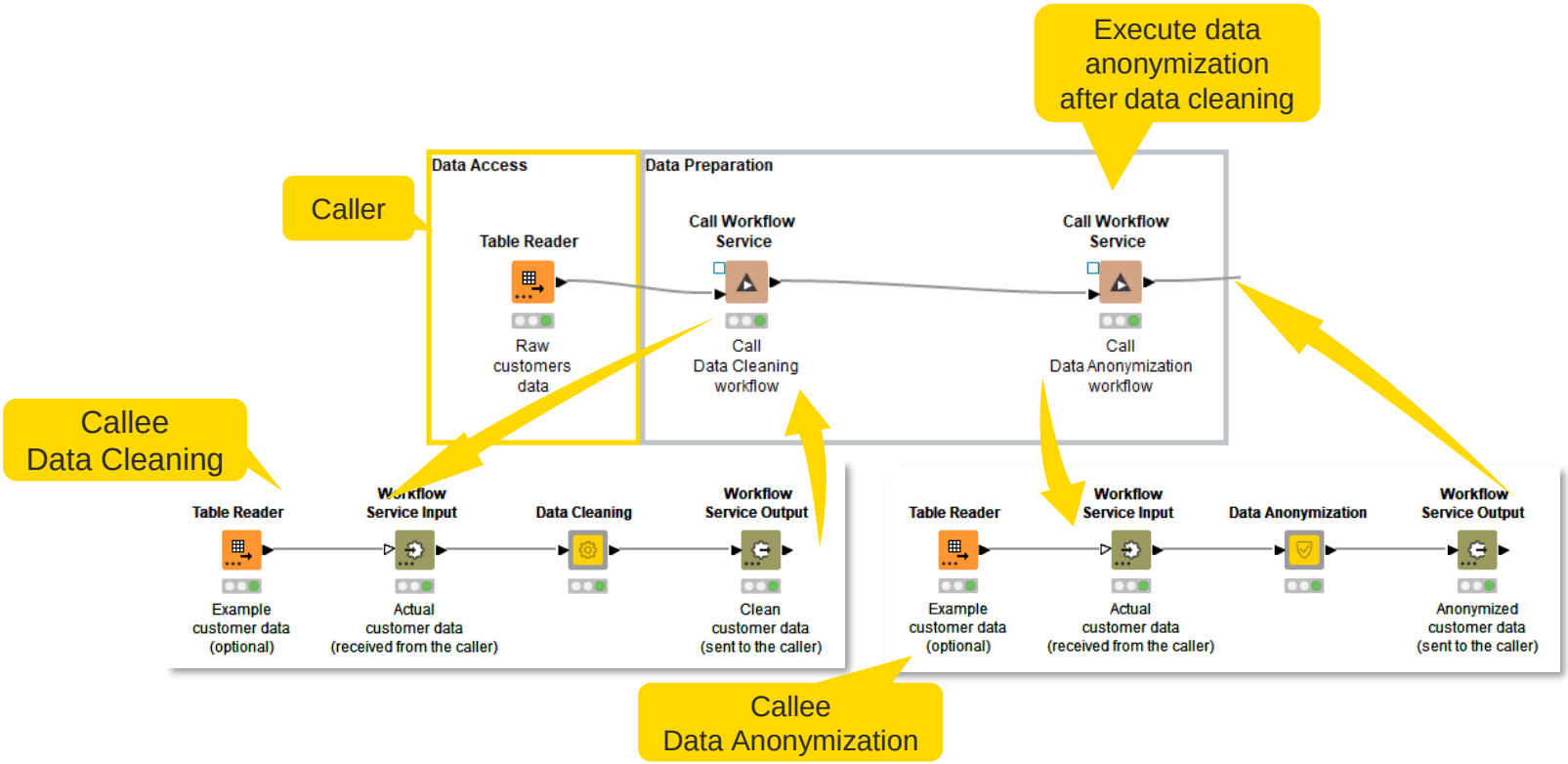
Orchestration

- Create modular workflows
  - Split workflows that perform different processes (*callee* workflows) and orchestration workflow (*caller*)
  - Callee can get data from and expose data to a caller
  - Call and orchestrate in the caller workflow
- Execute in parallel or setup workflow dependencies or cascades



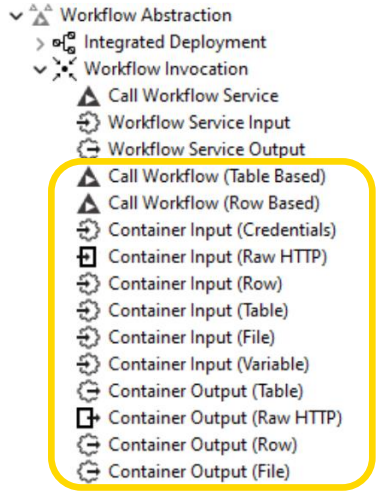
Sequential Workflow Execution

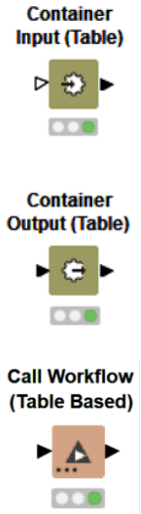
- Setup workflow dependencies using Workflow Services nodes



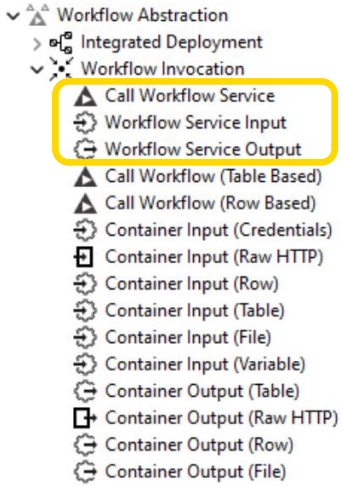
Workflow Invocation vs. Workflow Services

- Workflow Invocation (external clients)
  - Expose different data types to the REST interface
  - Use to define KNIME Business Hub REST APIs for **external clients**
  - Standardized (but limiting) JSON-based APIs



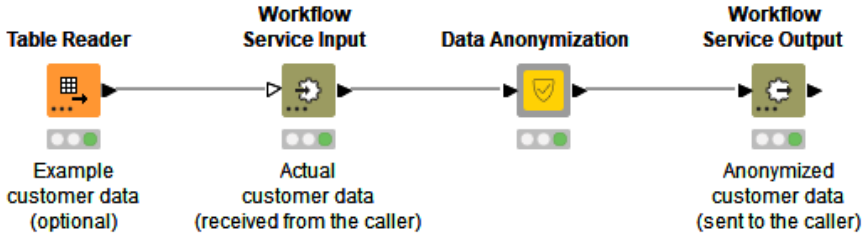


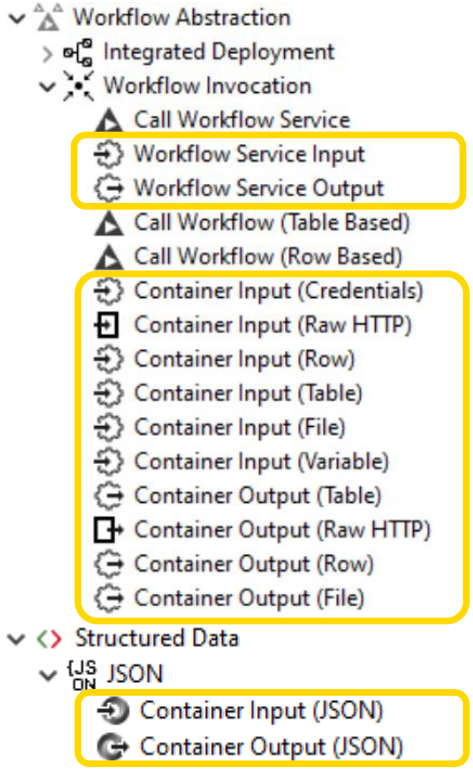
- Workflow Services
  - For **KNIME use** only - easier and **faster** to call KNIME workflows from other workflows
  - KNIME native API endpoints – no serialization into/from JSON-objects
  - Share text, models, and many more data types
  - Dynamic ports



Callee Workflow

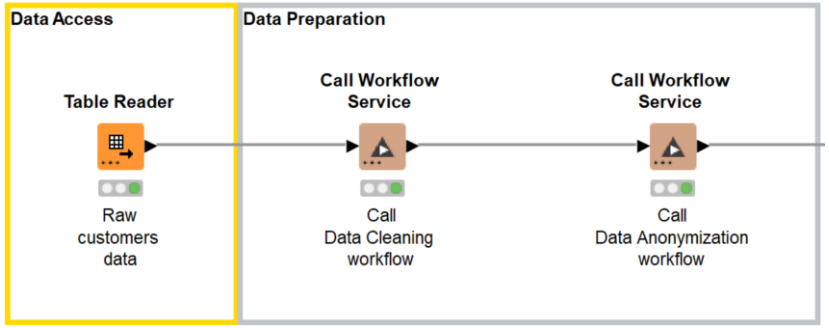
- Is called by the caller workflow
- Performs a particular task
- Contains
  - Workflow Service Input, Workflow Service Output (KNIME)
  - OR Container Input, or Container Output nodes (external)

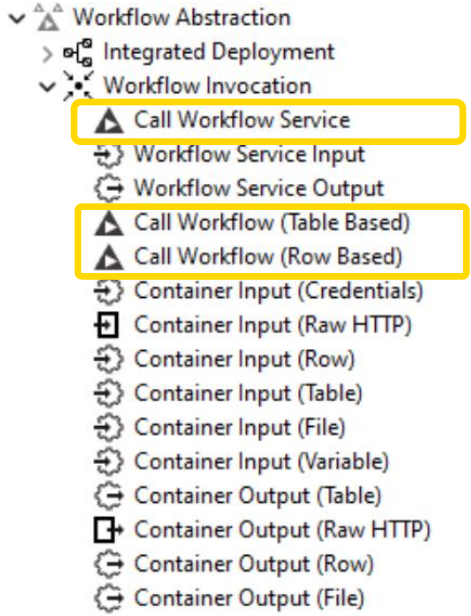




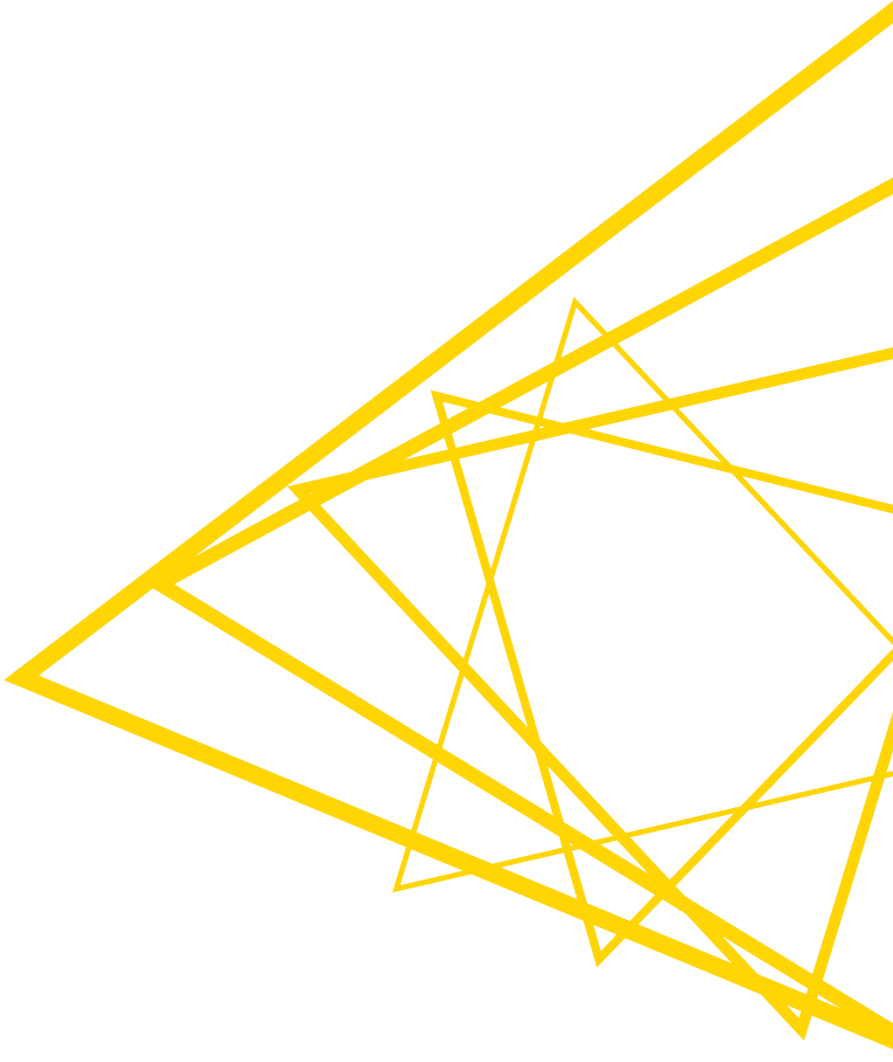
Caller Workflow

- Calls, sends data to, and gets the data from the callee workflows
- Orchestrates the dedicated callee workflows
- Contains
  - Call Workflow Service (KNIME)
  - Call Local Workflow, Call Remote Workflow, and Call Workflow nodes (external)



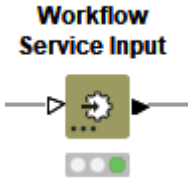


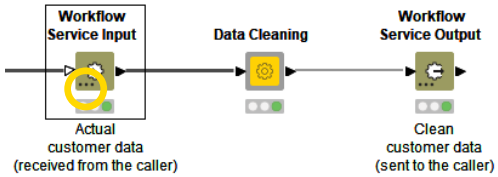
Workflow Services

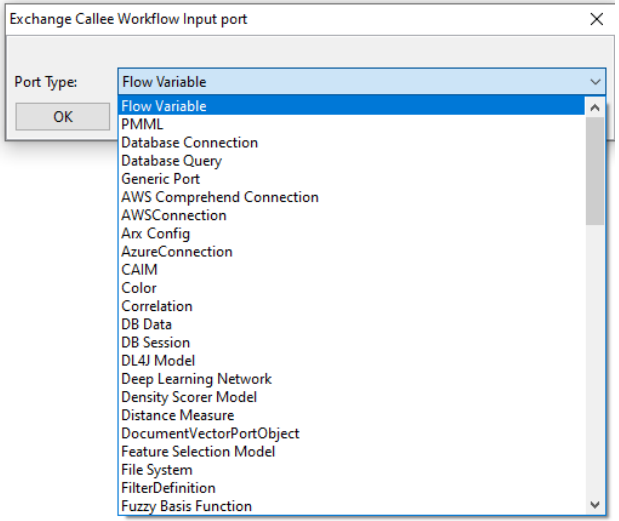


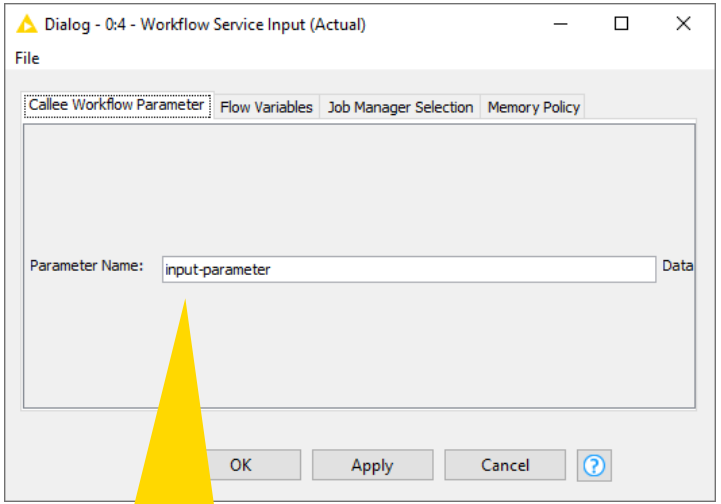
Workflow Service Input Node

- Receives an object from a caller
- Various port types are available





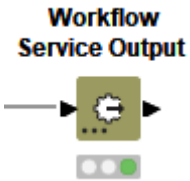


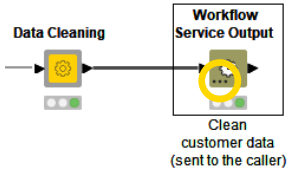


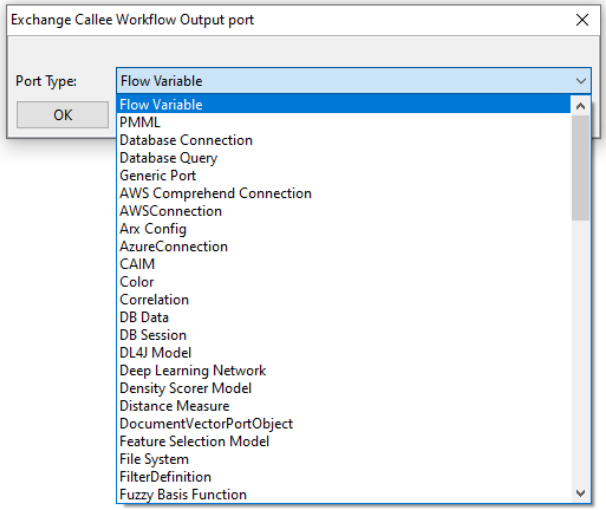
Give this parameter a meaningful name to recognize it from the Caller workflow

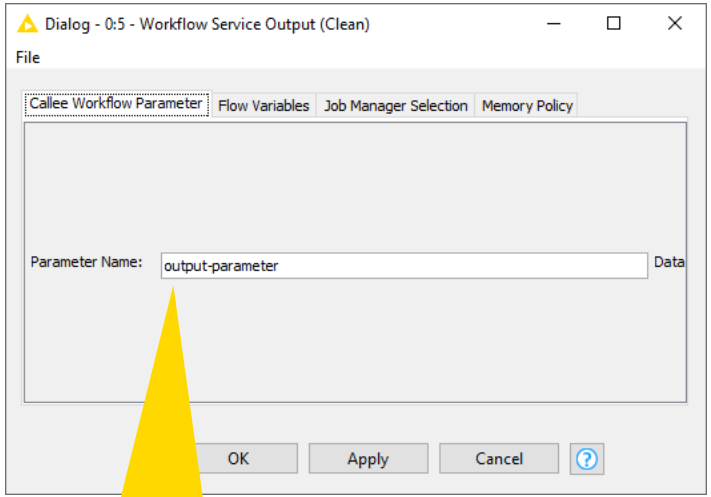
Workflow Service Output Node

- Sends an object to a caller
- Various port types are available





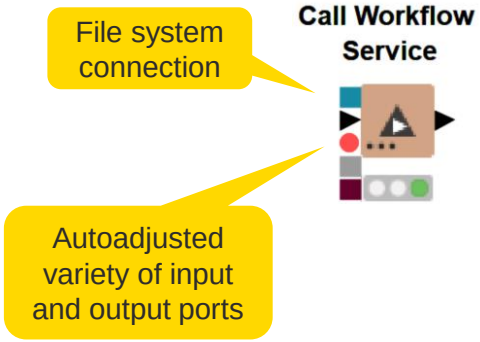


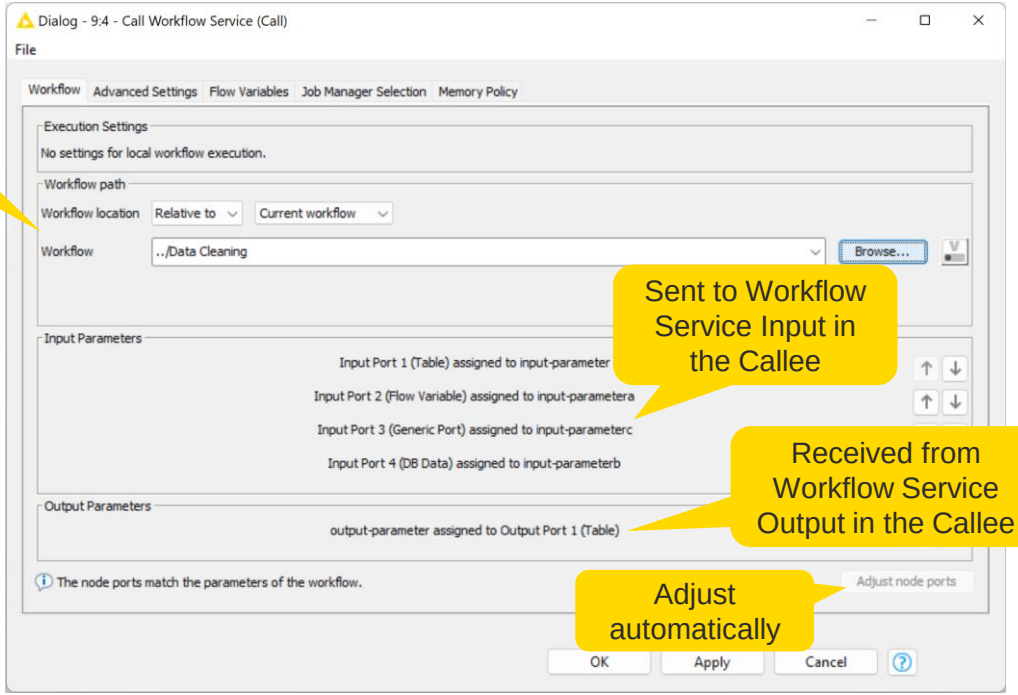


Give this parameter a meaningful name to recognize it from the Caller workflow

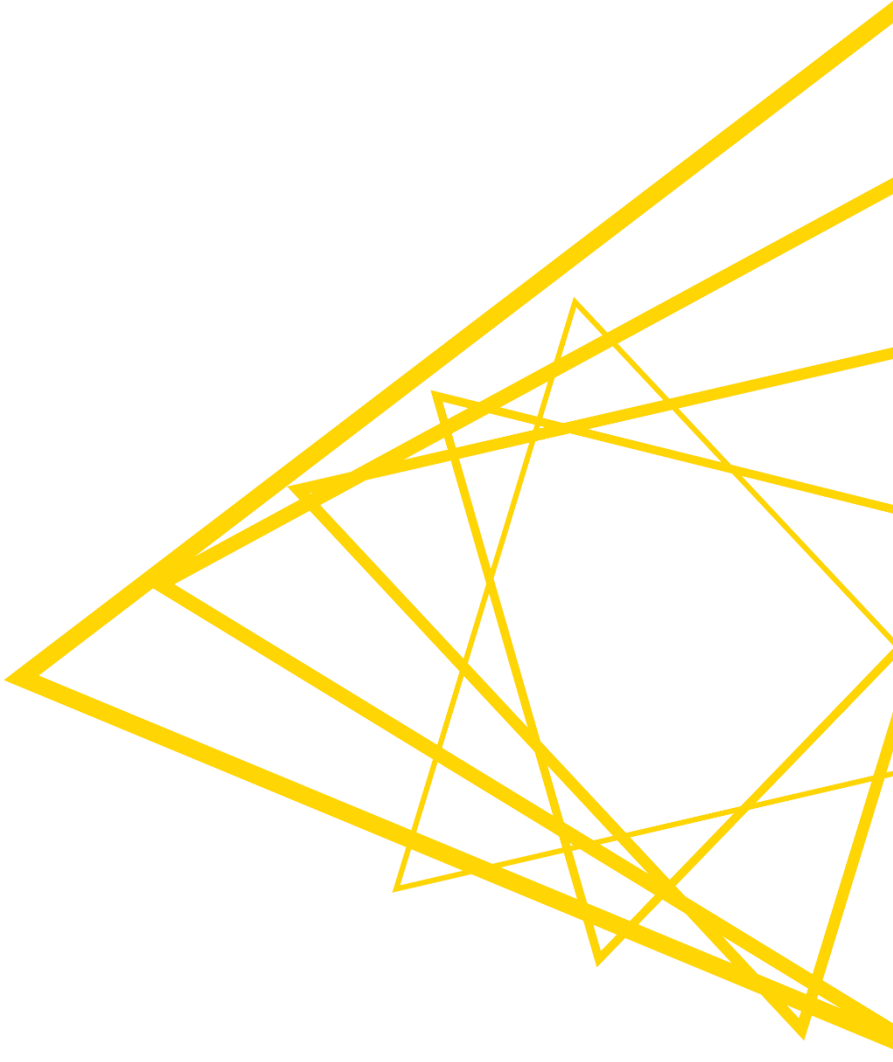
Call Workflow Service Node

- Calls other local and remote workflows
  - Sends input to, executes, & receives results from callee workflows
- Ports are adjusted automatically in accordance with the callee workflow selected in the configuration dialog
  - Various port types, multiple ports

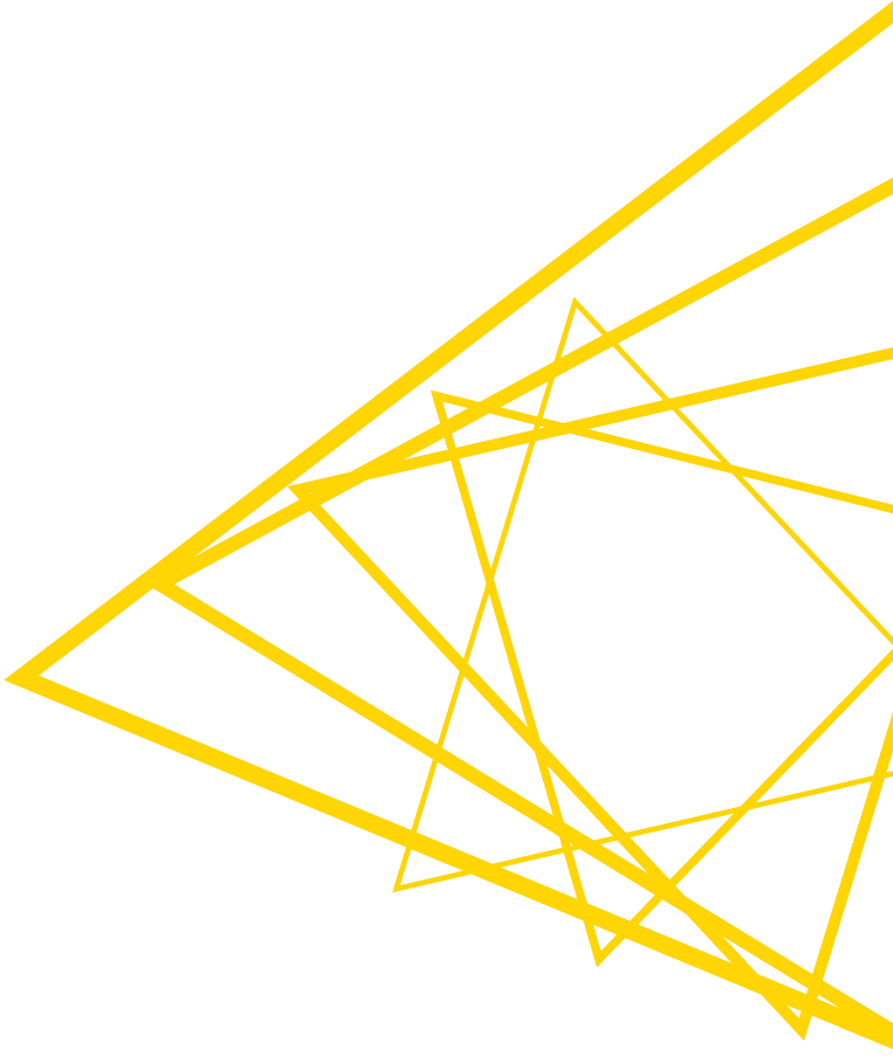




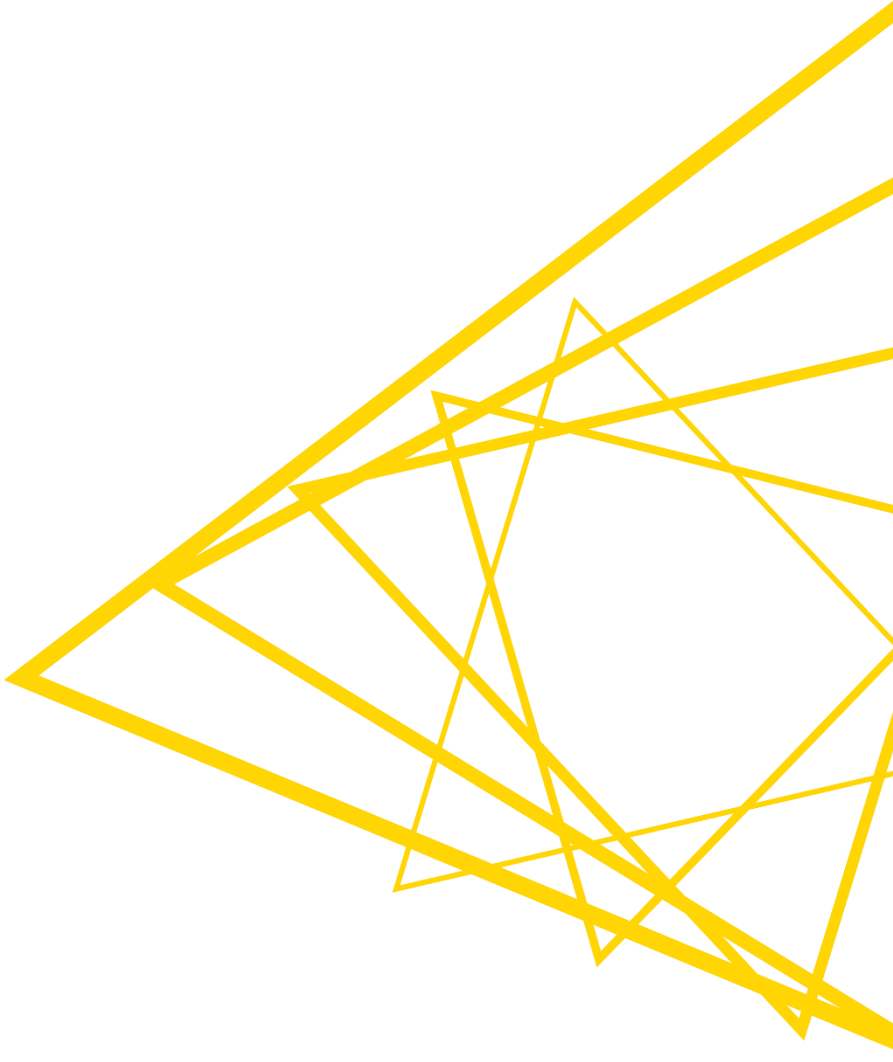
Demo



Best Practices: Workflow Orchestration



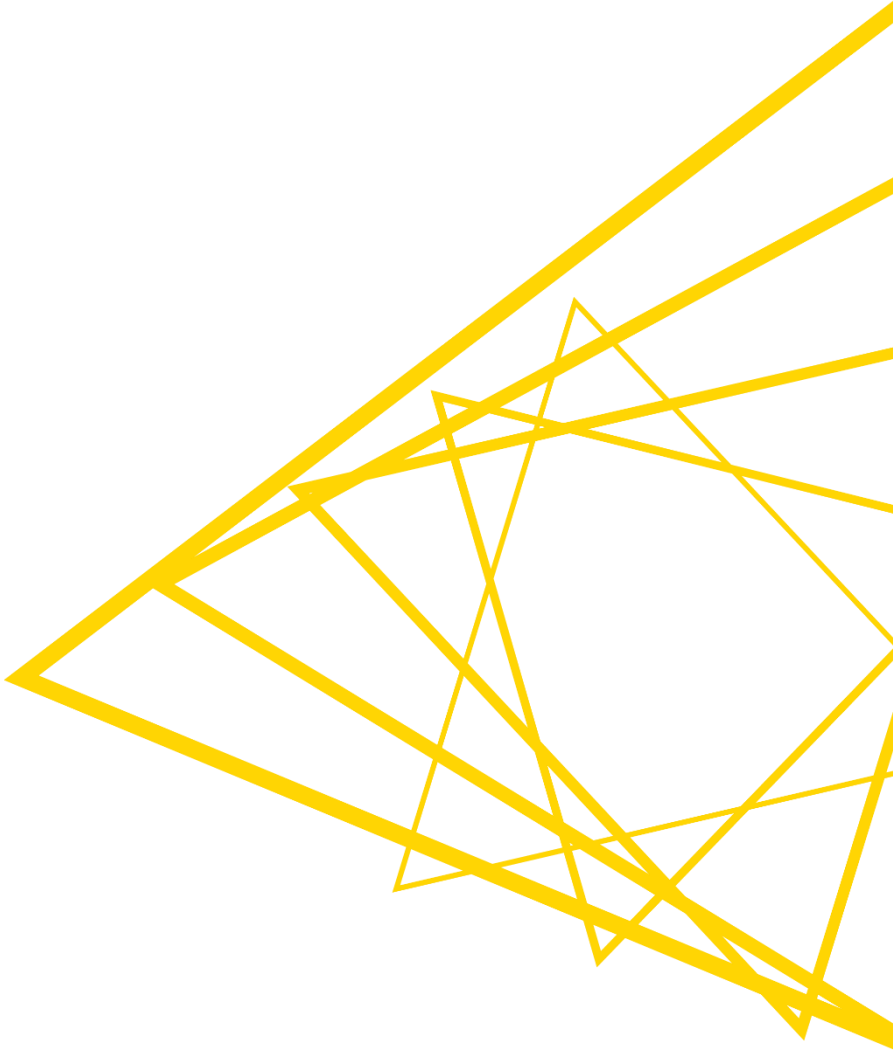
Efficiency



Efficiency via Orchestration

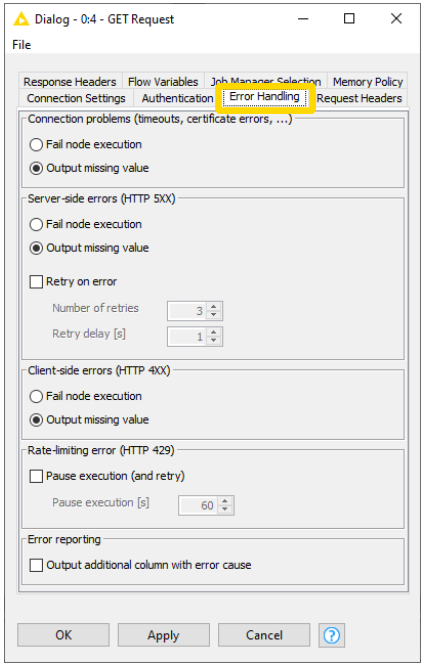
- Multiple, smaller workflows are sometimes faster than one big workflow
  - Makes it easier to maintain the workflows: to handle, i.e., grasp an overview, make changes
  - Each workflow performs a particular task
  - Makes it easier to test the workflows

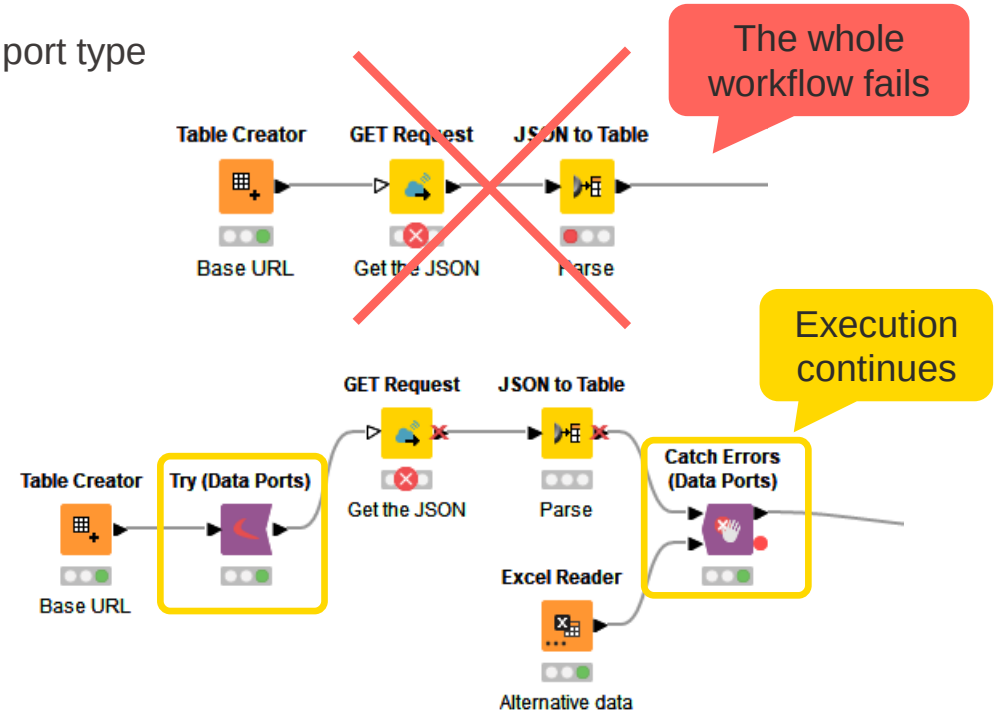
Error Handling



Handling Errors

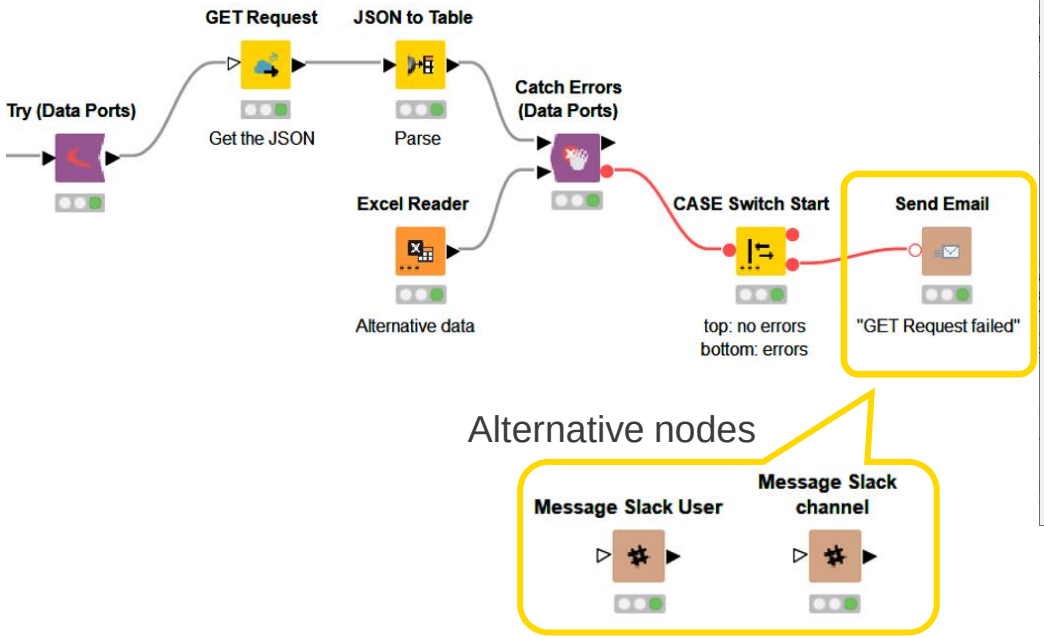
- Specify REST node behavior in case of error directly in the node
- Allow for a graceful exit with the Try & Catch nodes
  - Various ports available
  - Generic port can be used with any KNIME port type

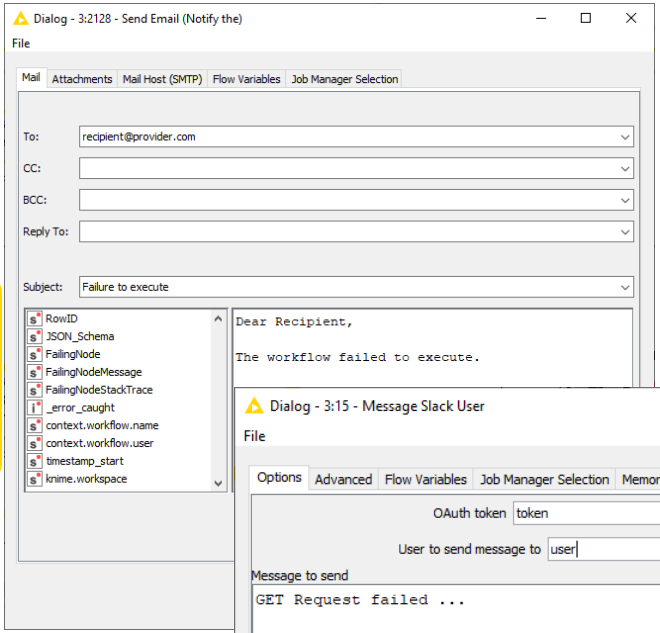


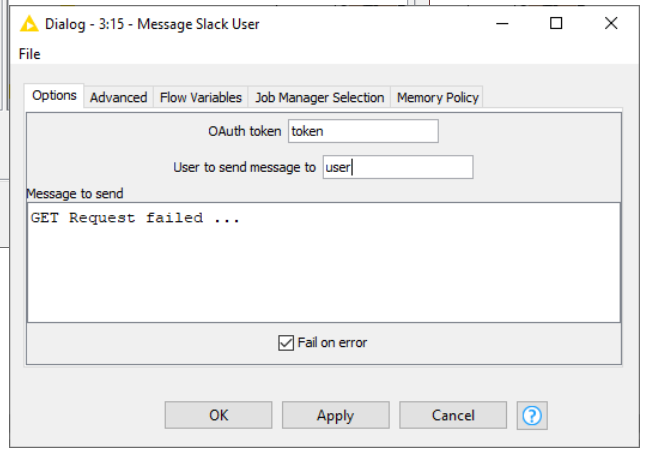


Reporting Failures

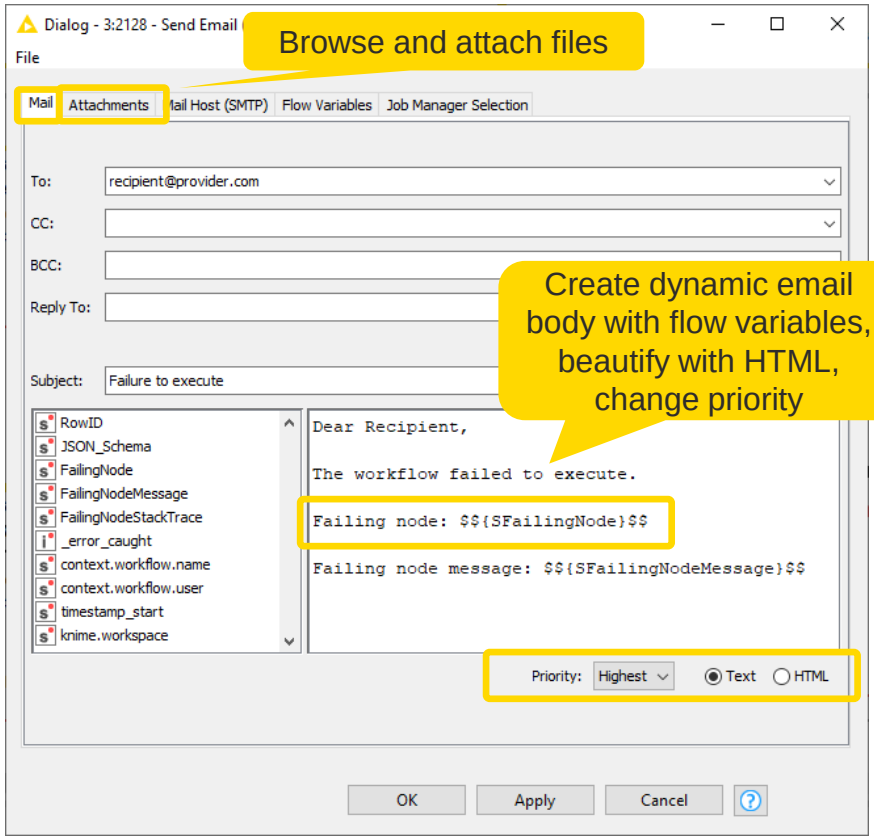
- Inform responsible people about a failure

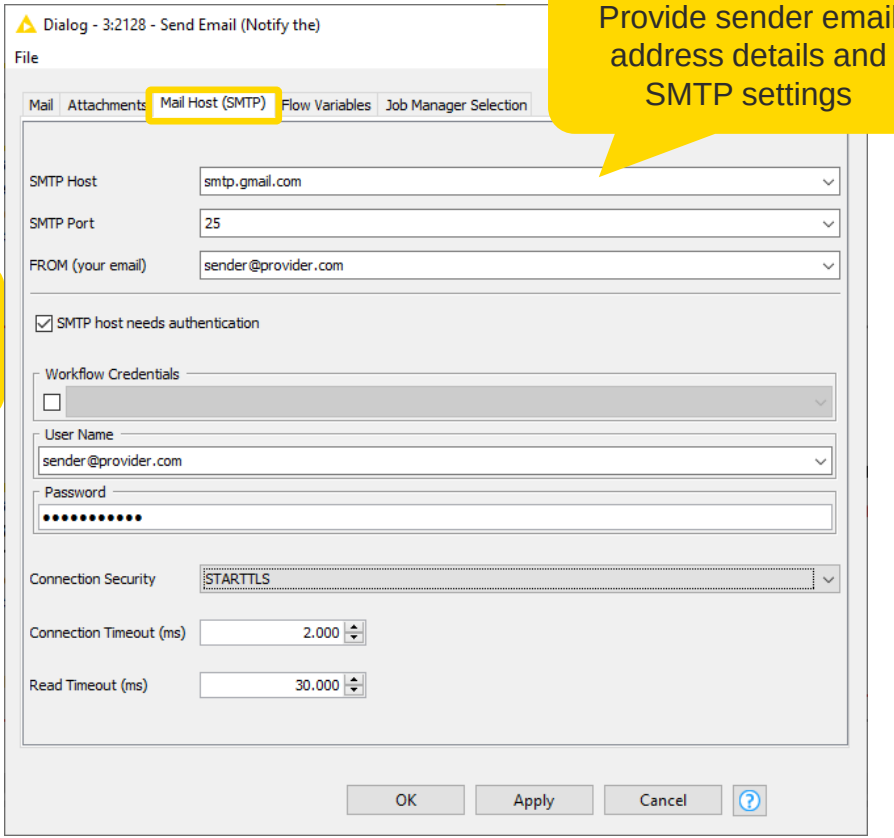






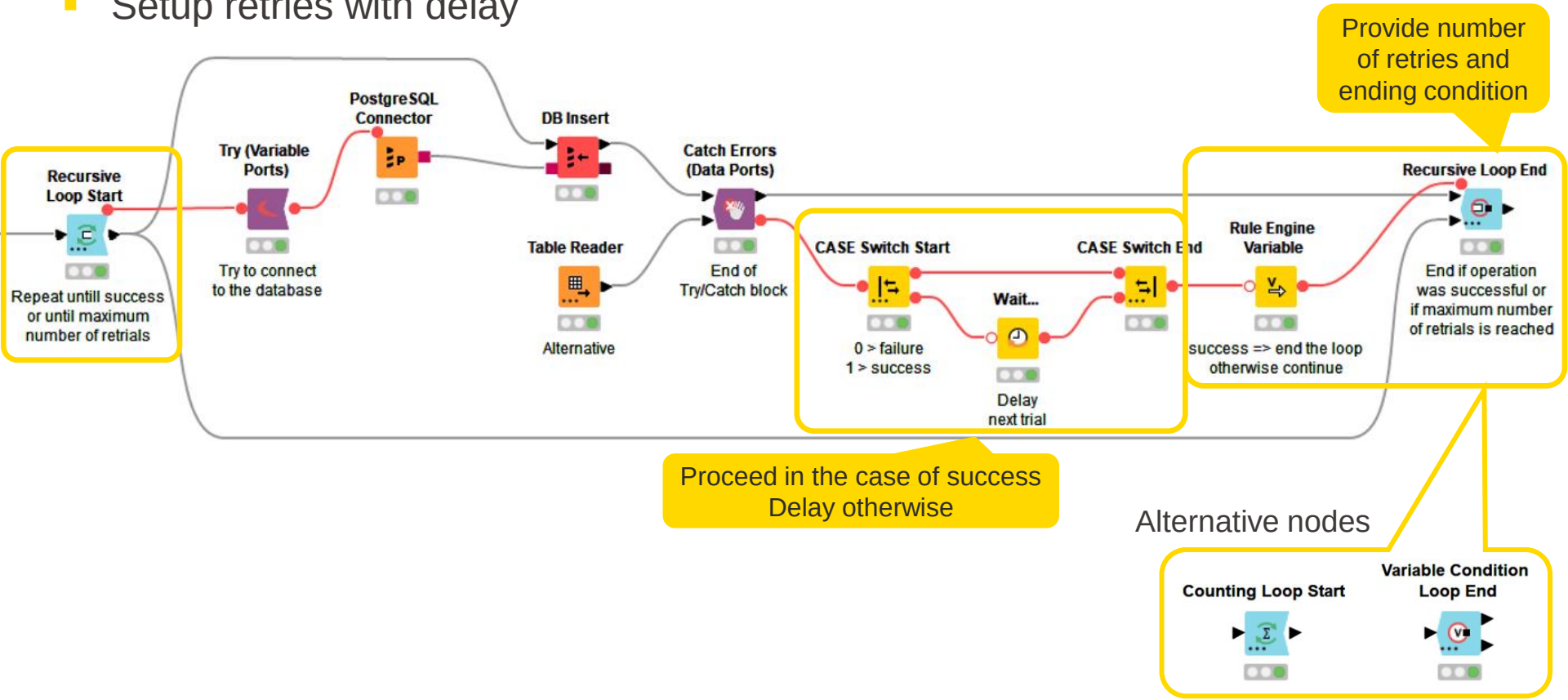
Send Email Node





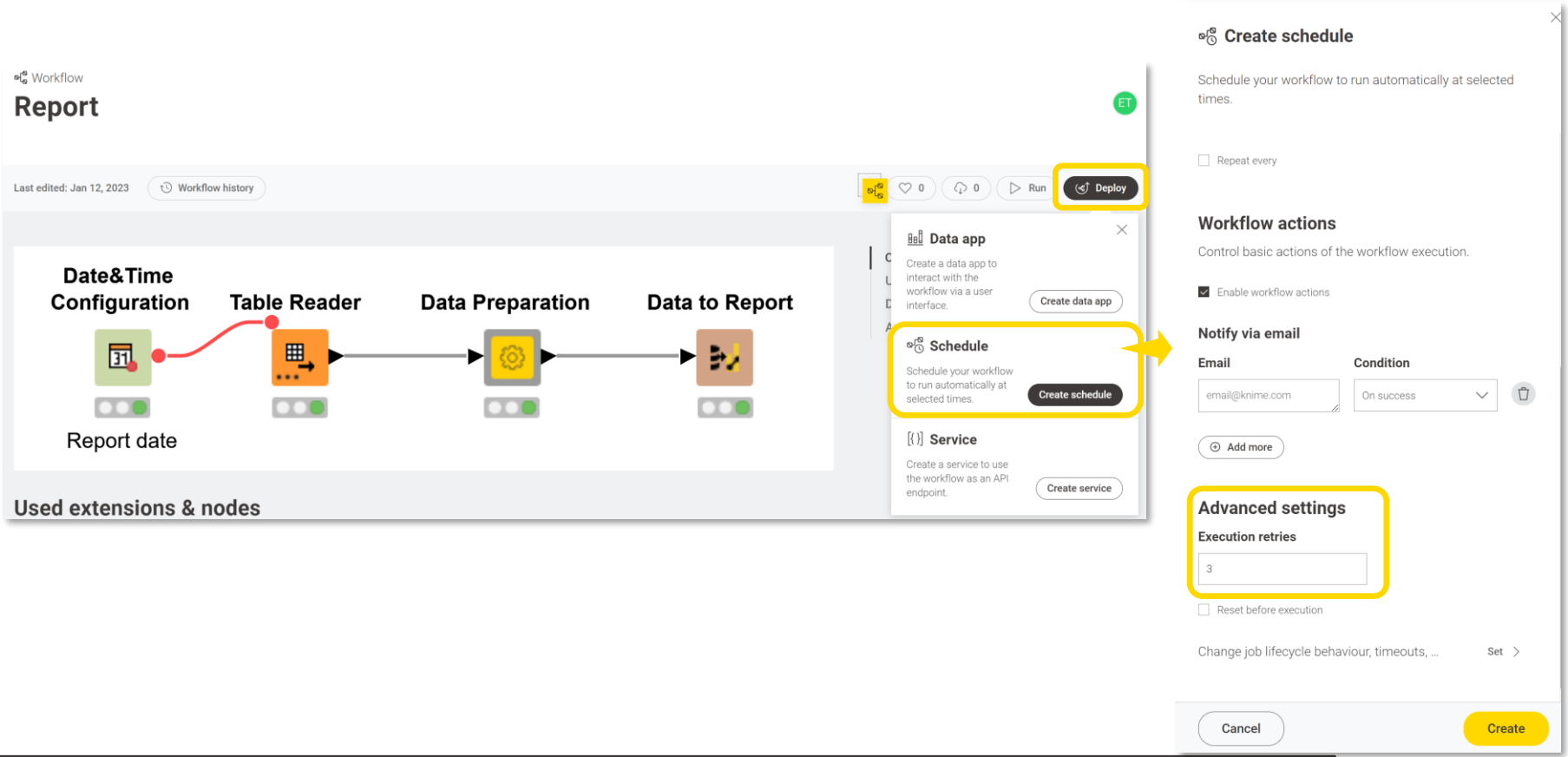
Handling Errors with Retries and Delay

- Setup retries with delay



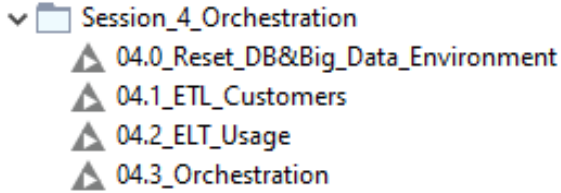
Handling Errors with Retries on the KNIME Business Hub

- Setup several retries when scheduling on the KNIME Business Hub



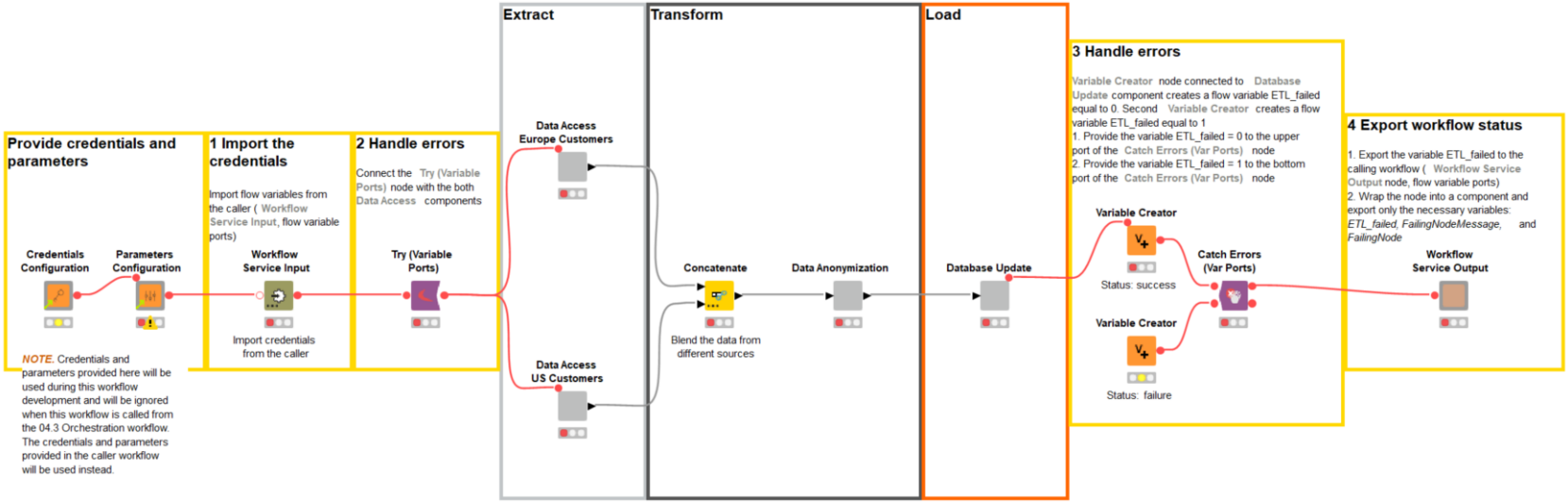
Exercises – Session 4

- Before starting the exercise **(skip if you performed these steps for session 1)**
  - Install local instance of PostgreSQL
  - Download the training workflows from the KNIME Community Hub
  - Install necessary extensions (open 00.1_Extensions_setup)
  - Execute workflow 00.2_Setup_PostgreSQL_Database
    - Use the credentials for your local instance of PostgreSQL
- Execute workflow 04.0_Reset_DB&Big_Data_Environment **(except this step)**



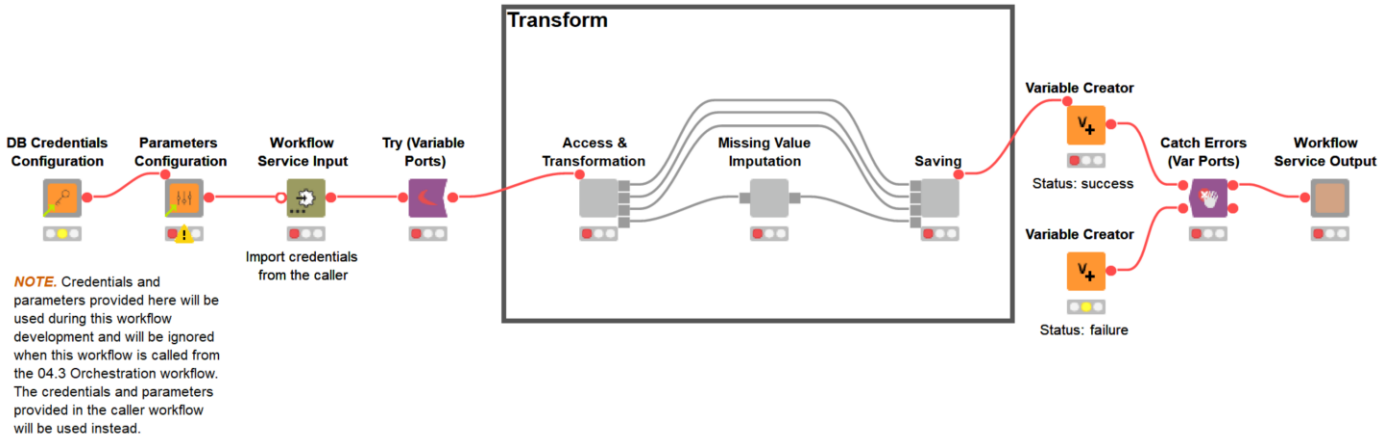
Exercise – 04.1_ETL_Customers

- This exercise is the final step to build application “ETL on Customers Data”
  - The solution to the previous exercises is already in the workflow
  - 1 Import credentials
  - 2-3 Handle errors
  - 4 Export final workflow status
  - Find detailed instructions in the workflow



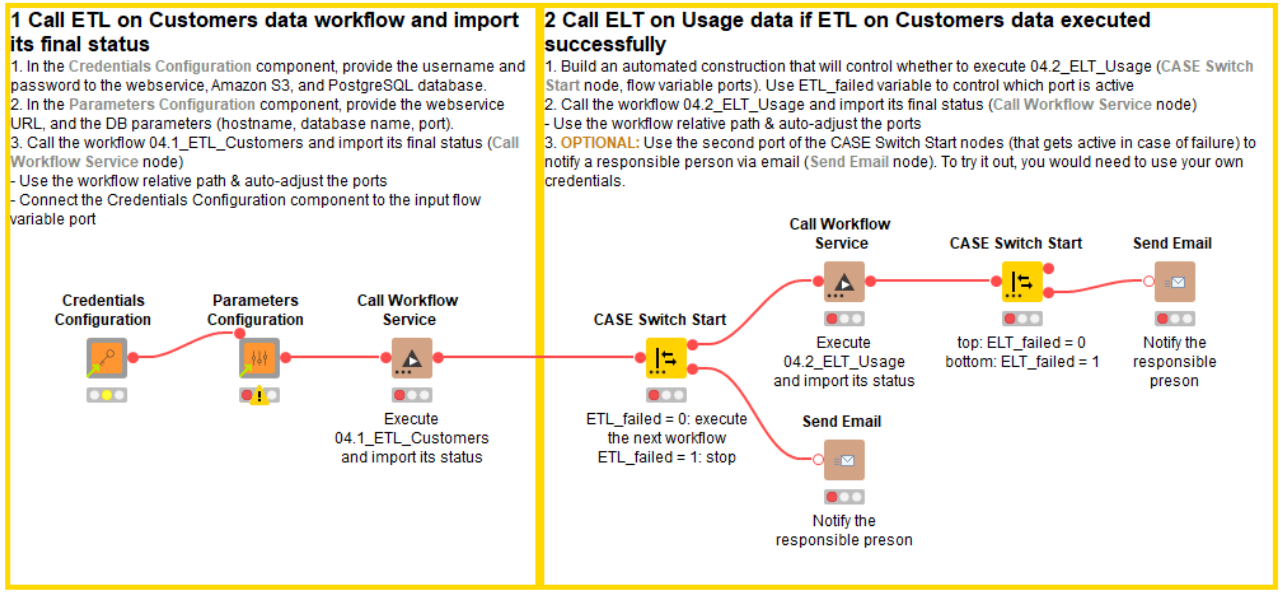
Exercise – 04.2_ELT_Usage

- This workflow is the final step to build application “ELT on Usage data”
  - **The solution is analogous to the exercise 04.1_ETL_Customers and is already provided**



Exercise – 04.3_Orchestration

- This exercise connects and orchestrates the applications “ETL on Customers Data” and “ELT on Usage data”
  - 1 Call ETL on Customers data and import its final status
  - 2 Call ELT on Usage data if ETL on Customers data executed successfully
  - Find detailed instructions in the workflow

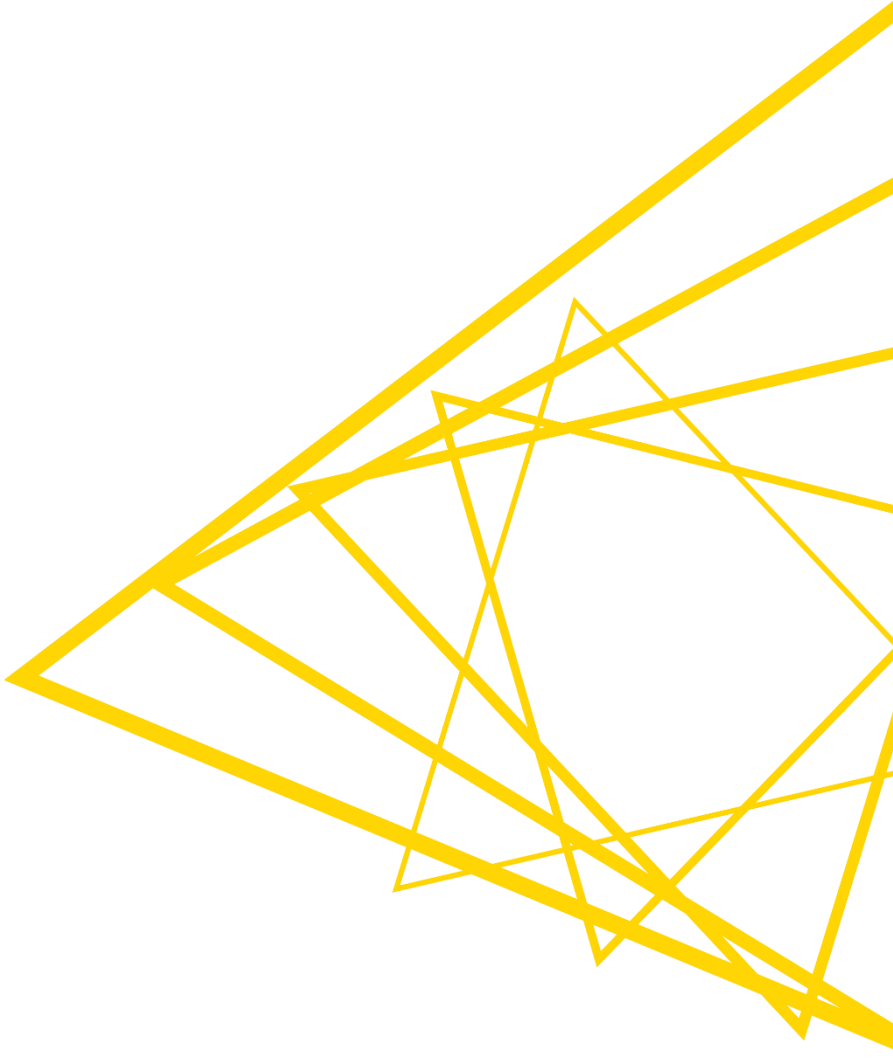


Session 4: Summary

Now you should be able to:

- Integrate cloud Hadoop Applications in KNIME
- Orchestrate modular workflows from a caller workflow
- Build an application that triggers and orchestrates the applications from previous sessions

Conclusions



Summary of the Course

- You learnt how to…
  - Access various data types and connect to various data sources
  - Build ETL and ELT data pipelines
  - Work with big data in KNIME Analytics Platform
  - Orchestrate multiple workflows and build workflow dependencies
  - Apply **best practices** for data engineers
  - Build and orchestrate ETL and ELT applications

Best Practices for Data Engineers

- Efficiency 
- Scalability 
- Reusability 
- Error handling
- Security 
- Repeatability 

More on best practices:

- KNIME Best Practices Guide
- Best Practices to Build KNIME Workflows - Webinar

Stay Up-To-Date and Connected

Blog: knime.com/blog

Forum:
forum.knime.com

KNIME Community Hub:
hub.knime.com

KNIME Self-paced Courses:
https://www.knime.com/knime-self-paced-courses

Follow us on social media:

Open for Innovation
KNIME

The End

education@knime.com
Twitter: @KNIME