



Open for Innovation

KNIME

[L4-BD] Introduction to Big Data with KNIME Analytics Platform

education@knime.com

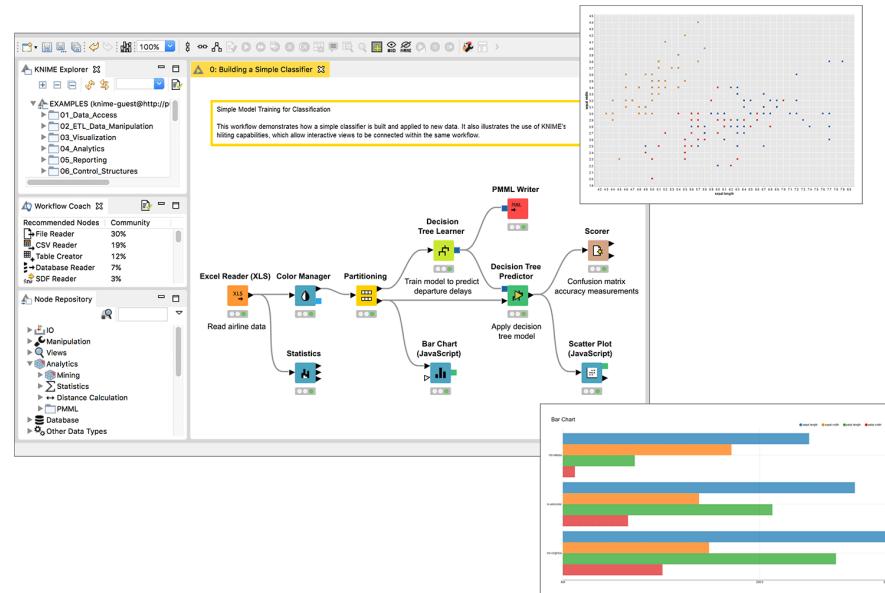


Overview KNIME Analytics Platform



What is KNIME Analytics Platform?

- A tool for data analysis, manipulation, visualization, and reporting
- Based on the graphical programming paradigm
- Provides a diverse array of extensions:
 - Text Mining
 - Network Mining
 - Cheminformatics
 - Many integrations, such as Java, R, Python, Weka, Keras, Plotly, H2O, etc.

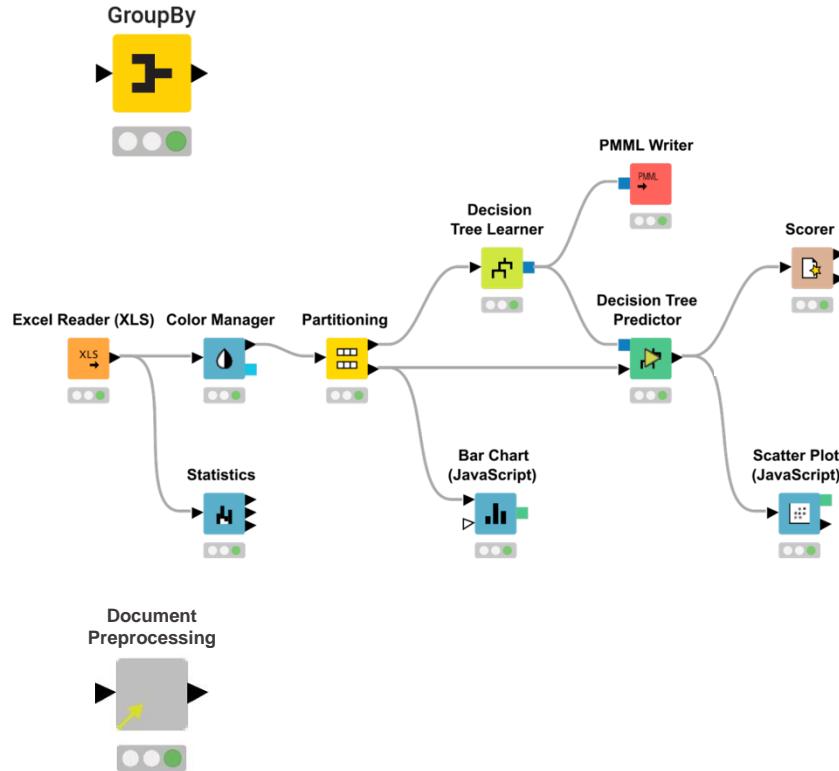


Visual KNIME Workflows

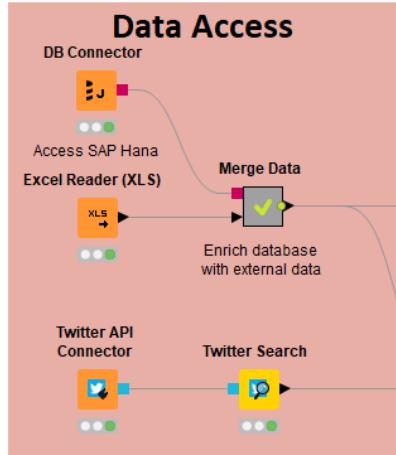
Nodes
perform tasks on data

Workflows
combine nodes to model data flow

Components
encapsulate complexity & expertise

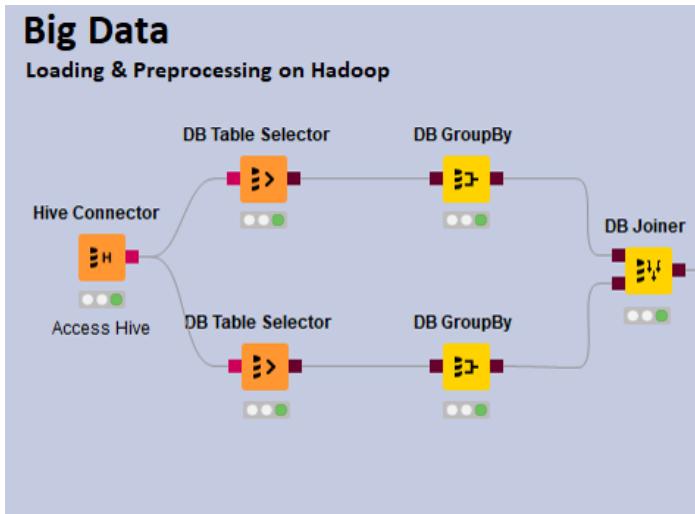


Data Access



- **Databases**
 - MySQL, PostgreSQL, Oracle
 - Theobald
 - any JDBC (DB2, MS SQL Server)
 - Amazon DynamoDB
- **Files**
 - CSV, txt, Excel, Word, PDF
 - SAS, SPSS
 - XML, JSON, PMML
 - Images, texts, networks
- **Other**
 - Twitter, Google
 - Amazon S3, Azure Blob Store
 - Sharepoint, Salesforce
 - Kafka
 - REST, Web services

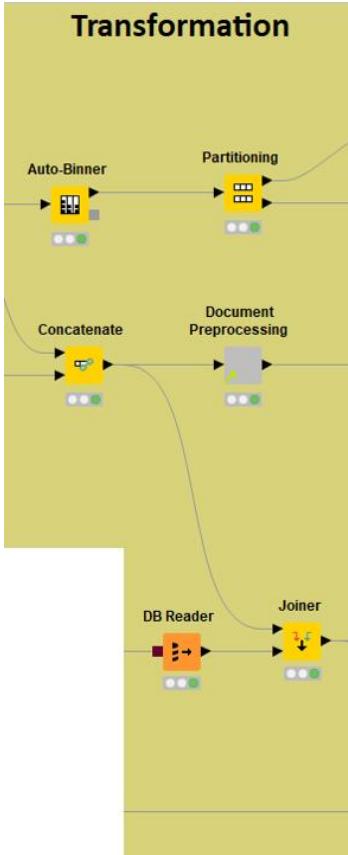
Big Data



- Spark & Databricks
- HDFS support
- Hive
- Impala
- In-database processing

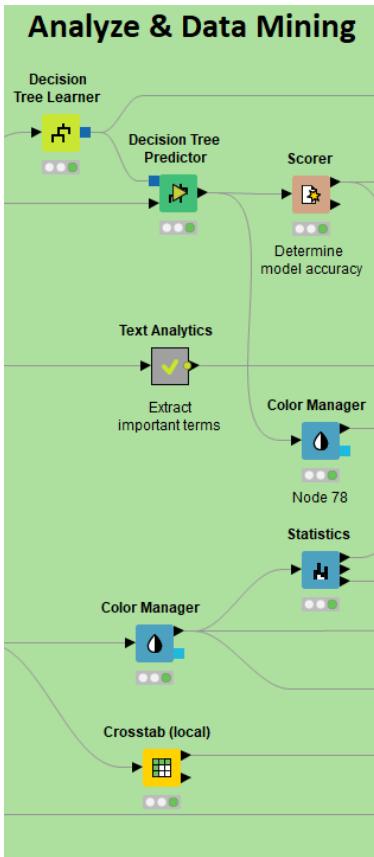


Transformation



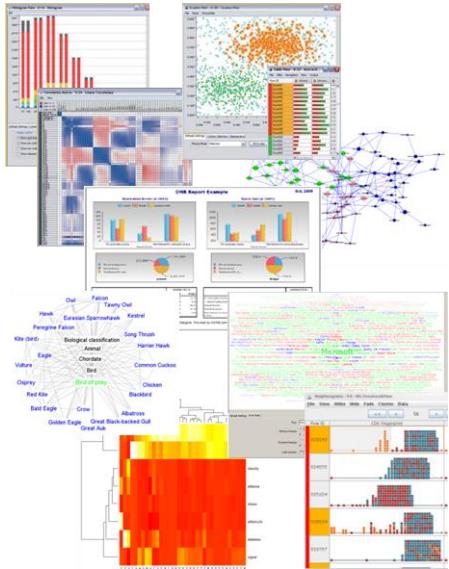
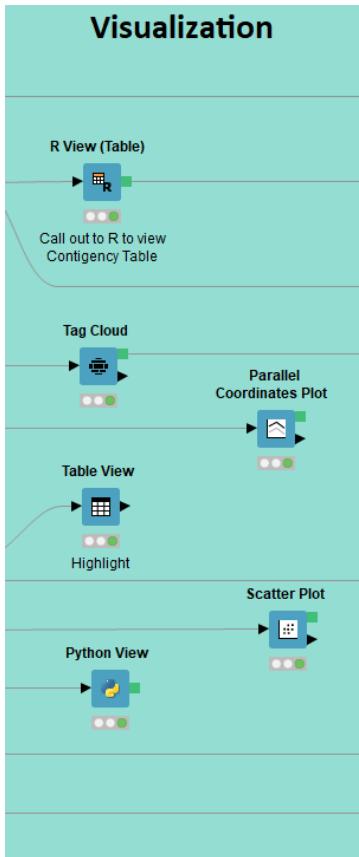
- Preprocessing
 - Row, column, matrix based
- Data blending
 - Join, concatenate, append
- Aggregation
 - Grouping, pivoting, binning
- Feature Creation and Selection

Analysis & Data Mining



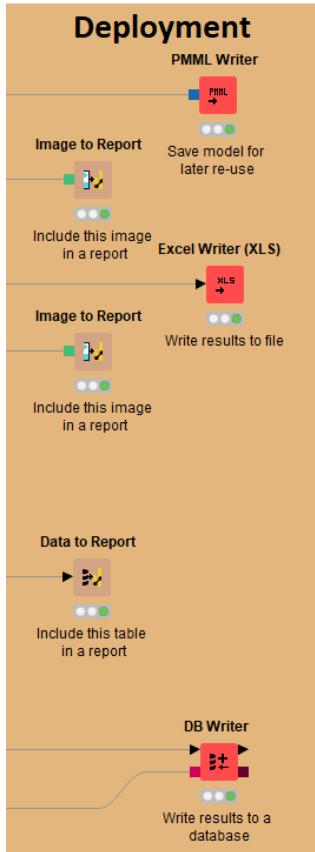
- Regression
 - Linear, logistic
- Classification
 - Decision tree, ensembles, SVM, MLP, Naïve Bayes
- Clustering
 - k-means, DBSCAN, hierarchical
- Validation
 - Cross-validation, scoring, ROC
- Deep Learning
 - Keras, DL4J
- External
 - R, Python, Weka, H2O, Keras

Visualization



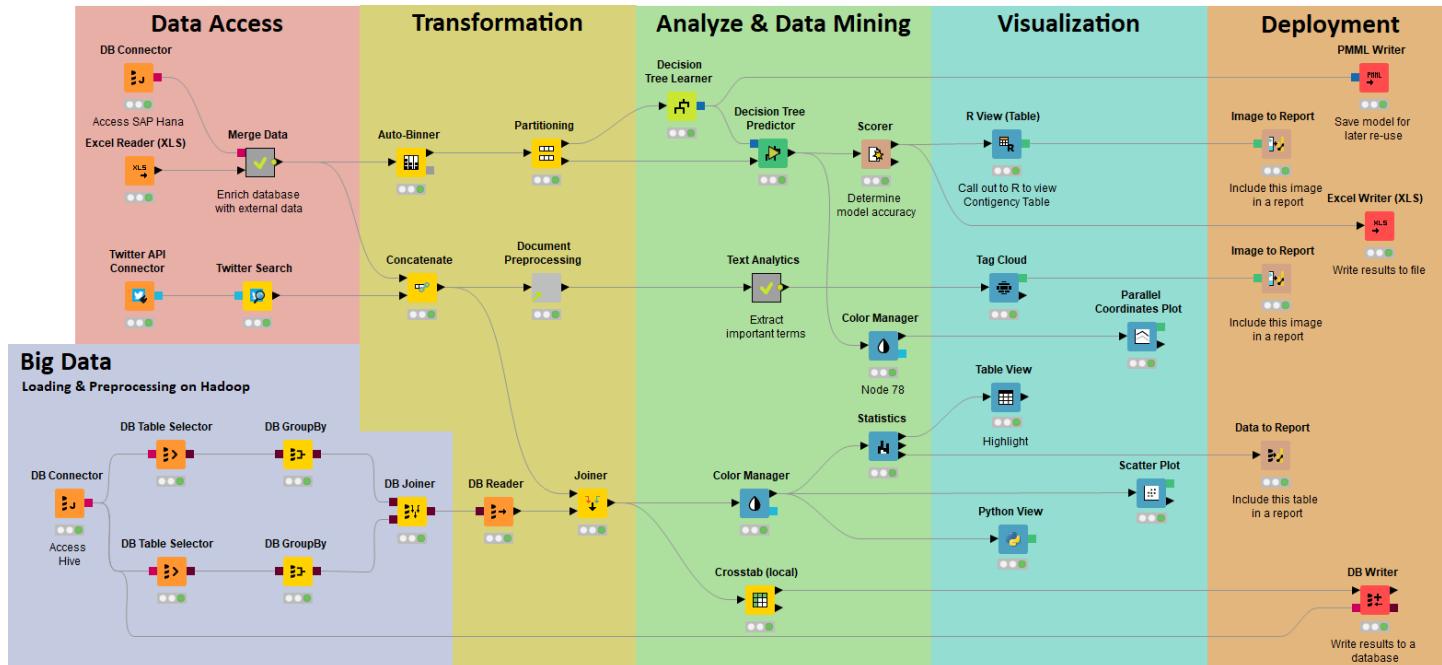
- Interactive Visualizations
- JavaScript-based nodes
 - Scatter Plot, Box Plot, Line Plot
 - Networks, ROC Curve, Decision Tree
 - Plotly Integration
 - Adding more with each release!
- Misc
 - Tag cloud, open street map, molecules
- Script-based visualizations
 - R, Python

Deployment



- Database
- Files
 - Excel, CSV, txt
 - XML
 - PMML
 - to: local, KNIME Server, Amazon S3, Azure Blob Store
- BIRT Reporting

Over 2000 Native and Embedded Nodes Included:



Data Access

MySQL, Oracle, ...
SAS, SPSS, ...
Excel, Flat, ...
Hive, Impala, ...
XML, JSON, PMML
Text, Doc, Image, ...
Web Crawlers
Industry Specific
Community / 3rd

Transformation

Row
Column
Matrix
Text, Image
Time Series
Java
Python
Community / 3rd

Analysis & Mining

Statistics
Data Mining
Machine Learning
Web Analytics
Text Mining
Network Analysis
Social Media
Analysis
R, Weka, Python
Community / 3rd

Visualization

R
JFreeChart
JavaScript
Plotly
Community / 3rd

Deployment

via BIRT
PMML
XML, JSON
Databases
Excel, Flat, etc.
Text, Doc, Image
Industry Specific
Community / 3rd

Install KNIME Analytics Platform

- Select the KNIME version for your computer:
 - Mac
 - Windows – 32 or 64 bit
 - Linux
- Download the archive and extract the file, or download installer package and run it

Windows

KNIME Analytics Platform for Windows (installer)
The installer adds an icon to the desktop and suggests suitable memory settings

[Download](#) (459 MB)

KNIME Analytics Platform for Windows (self-extracting archive)
The self-extracting archive only creates a folder holding the KNIME installation

[Download](#) (463 MB)

KNIME Analytics Platform for Windows (zip archive)

[Download](#) (547 MB)

Linux

KNIME Analytics Platform for Linux

[Download](#) (583 MB)

Mac

KNIME Analytics Platform for macOS (10.13 and above)

[Download](#) (438 MB)

Find out what's new in the latest KNIME 4.4 release [here](#).

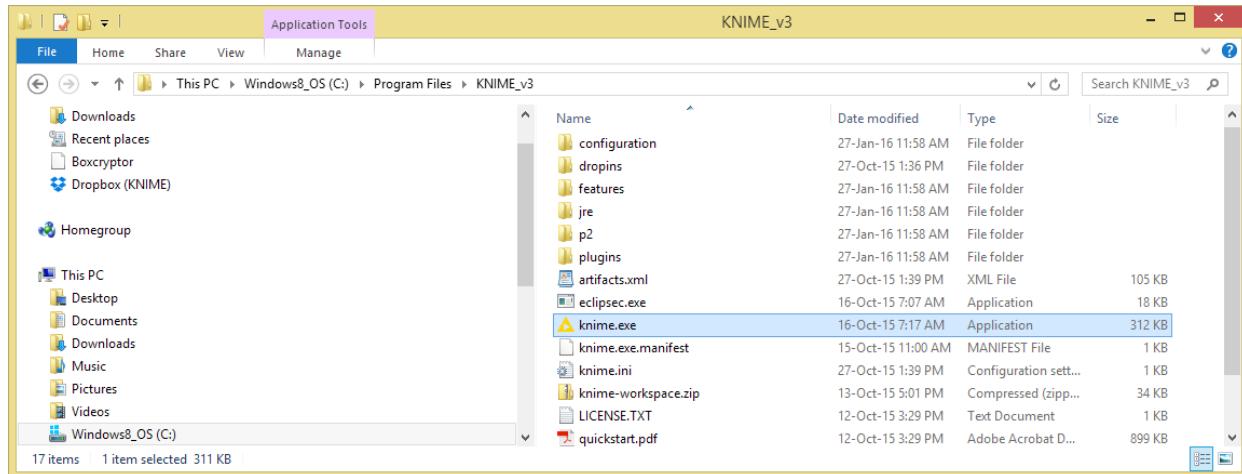
If you are interested in a previous version of KNIME Analytics Platform, please click [here](#).

Start KNIME Analytics Platform

- Use the shortcut created by the installer

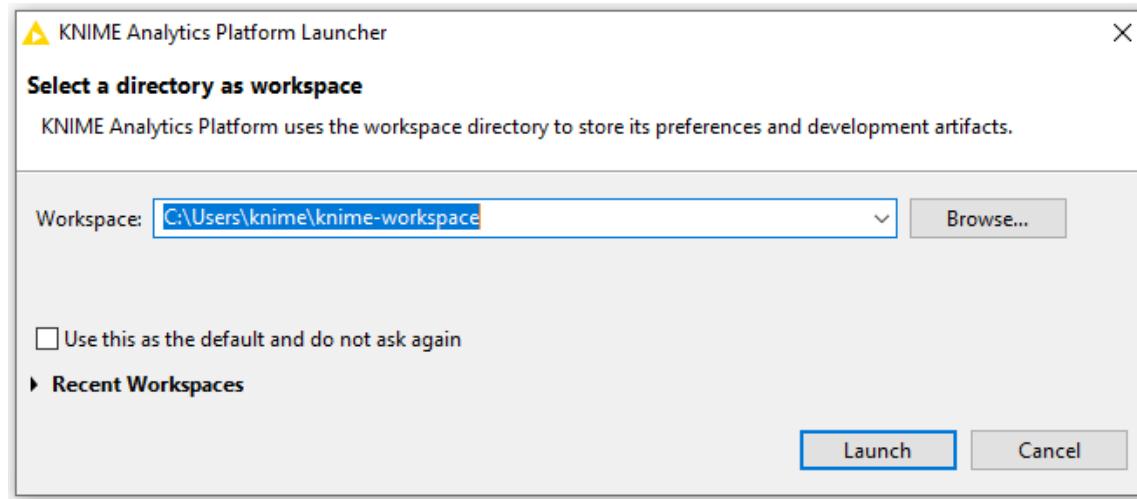


- Or go to the installation directory and launch KNIME via the knime.exe



The KNIME Workspace

- The workspace is the **folder/directory** in which workflows (and potentially data files) are stored for the current KNIME session
- Workspaces are portable (just like KNIME)

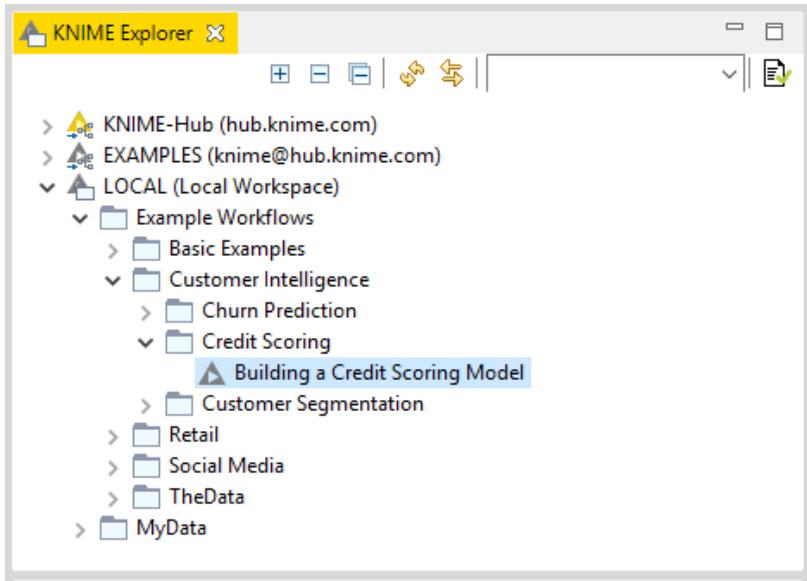


The KNIME Analytics Platform Workbench

The screenshot displays the KNIME Analytics Platform Workbench interface, which includes several key components:

- KNIME Explorer**: A sidebar containing links to the KNIME Hub, EXAMPLES, LOCAL Local Workspace, and Example Workflows.
- Workflow Coach**: A sidebar listing recommended nodes categorized by domain (Community, Data Manipulation, etc.) and their usage percentages.
- Node Repository**: A sidebar listing various node categories such as IO, Manipulation, Views, Analytics, DB, and Reporting.
- Outline**: A panel showing a hierarchical tree structure of the current workflow.
- Workflow Editor**: The central workspace where a workflow named "My first Workflow" is displayed. The workflow consists of four nodes connected sequentially:
 - File Reader: read adult.csv
 - Row Filter: keep only records born in the US
 - Column Filter: remove gender
 - Table Writer: write table
- Console & Node Monitor**: A panel at the bottom showing the execution status of the Row Filter node (3/6 rows processed, EXECUTED). It also displays a preview of the data being processed, showing columns like ID, age, workclass, fnlwgt, education, education-num, marital-status, occupation, relationship, race, sex, capital-gain, capital-loss, and hours.
- Description**: A panel on the right providing detailed information about the Row Filter node, including its configuration steps and dialog options.
- Welcome to the KNIME Hub**: A sidebar for navigating the KNIME Hub, featuring links for Open for Innovation, KNIME Hub, Sign In, and a search bar.

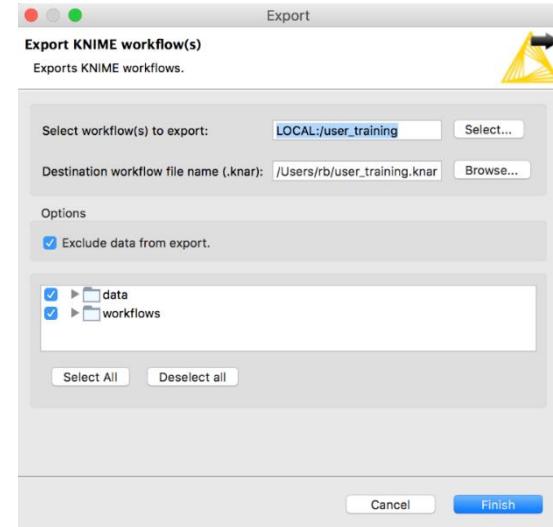
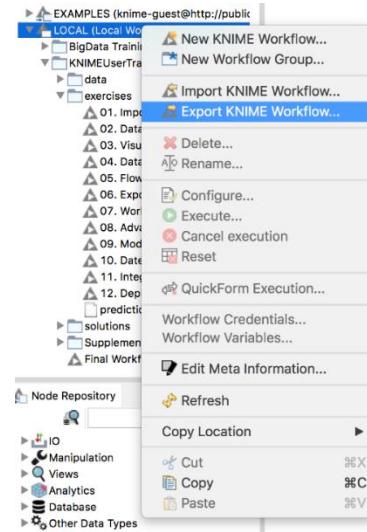
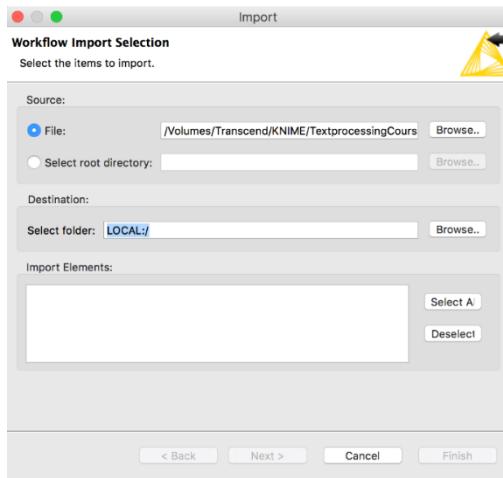
KNIME Explorer



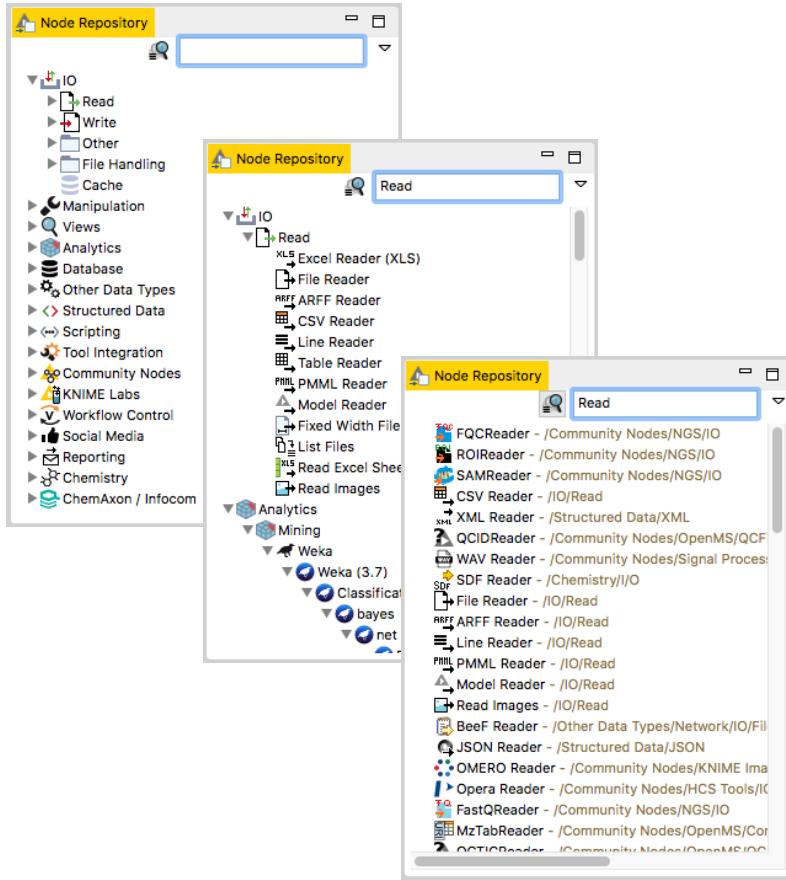
- In LOCAL you can access your own workflow projects.
- Other mountpoints allow you to connect to
 - EXAMPLE Server
 - KNIME Hub
 - KNIME Server
- The Explorer toolbar on the top has a search box and buttons to
 - select the workflow displayed in the active editor
 - refresh the view
- The KNIME Explorer can contain 4 types of content:
 - Workflows
 - Workflow groups
 - Data files
 - Shared Components

Creating New Workflows, Importing, and Exporting

- Right-click inside the KNIME Explorer to create a new workflow or a workflow group, or to import a workflow
- Right-click the workflow or workflow group to export

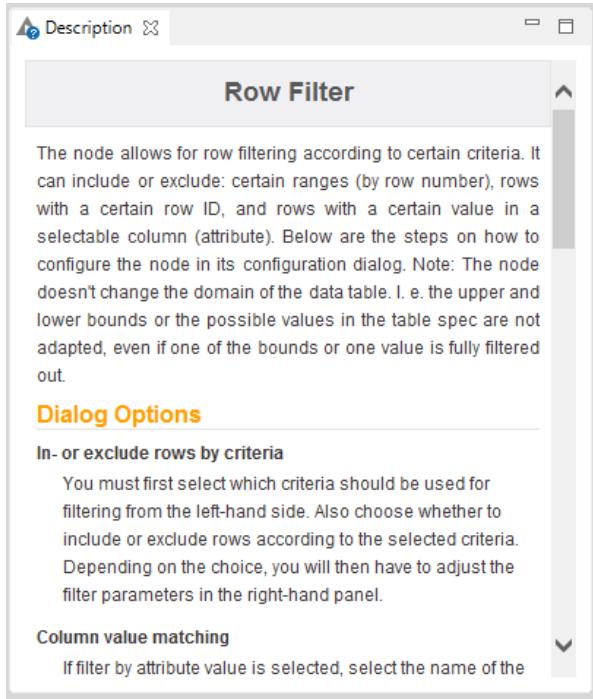


Node Repository



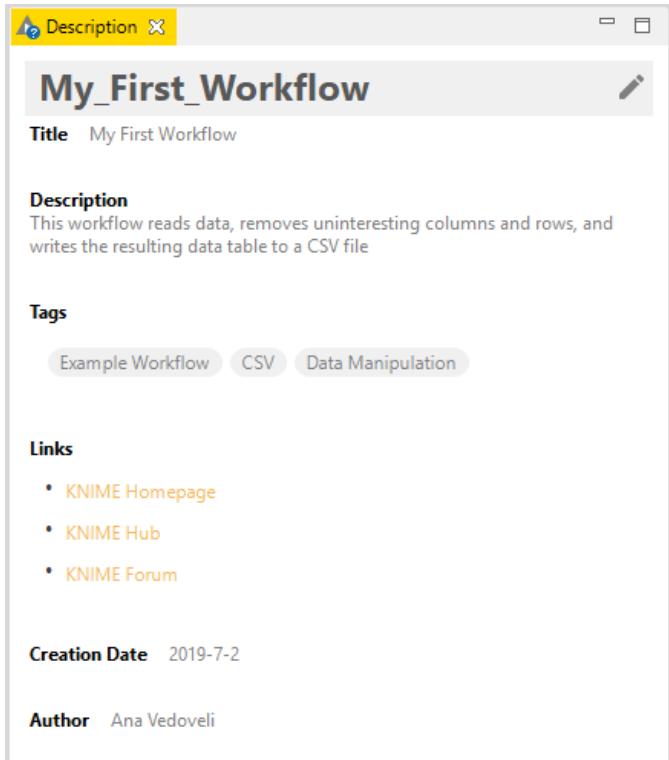
- The Node Repository lists all KNIME nodes
- The search box has 2 modes
 - Standard Search – exact match of node name
 - Fuzzy Search – finds the most similar node name

Description



- The Description view provides information about:
 - Node functionality
 - Input & output
 - Node settings
 - Ports
 - References to literature

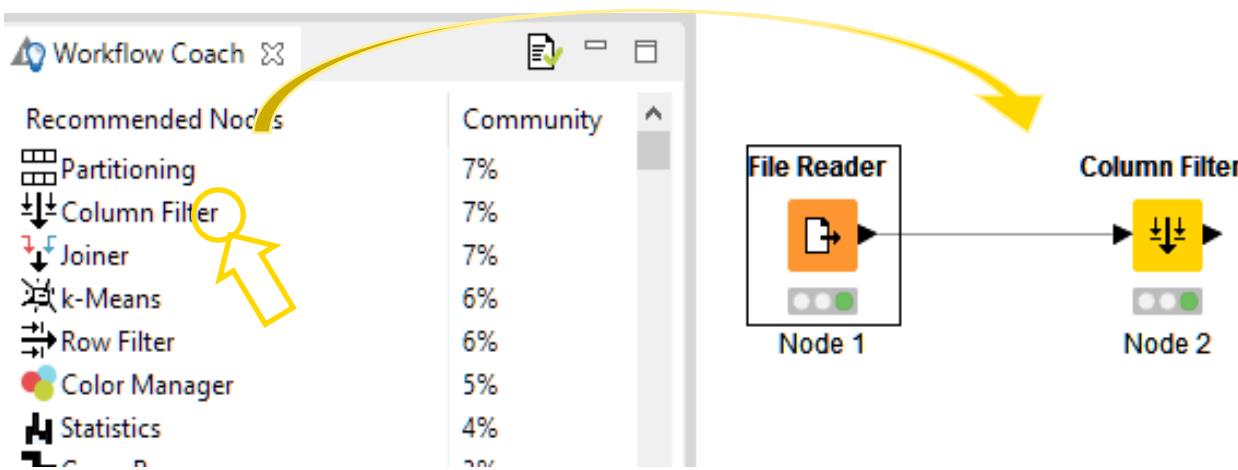
Workflow Description



- When selecting the workflow, the Description view gives you information about the workflow:
 - Title
 - Description
 - Associated tags and links
 - Creation date
 - Author

Workflow Coach

- Node recommendation engine
 - Gives hints about which node to use next in the workflow
 - Based on KNIME communities' usage statistics
 - Based on own KNIME workflows



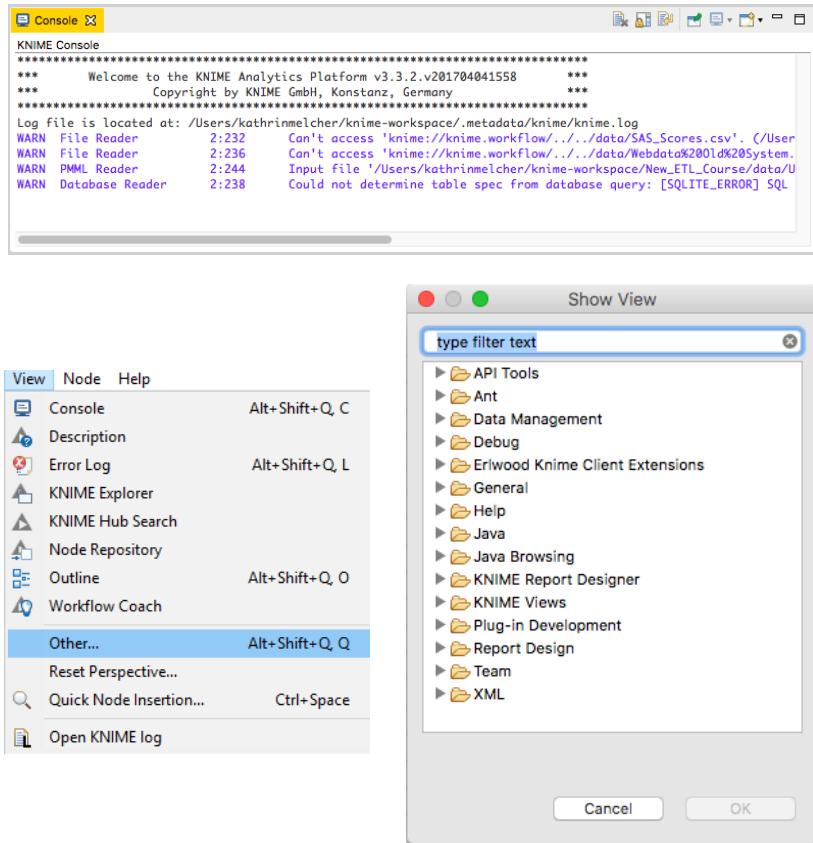
Node Monitor

- By default the Node Monitor shows you the output table of the node selected in the workflow editor
- Click on the three dots on the upper right to show the flow variables, configuration, etc.

The screenshot shows the KNIME Node Monitor interface. At the top, there are tabs for 'Console' and 'Node Monitor'. Below the tabs, the node information is displayed: 'Node: Get Customers from Database (0:1207)' and 'State: EXECUTED'. A dropdown menu labeled 'Port Output' is set to 'Port 0', and a 'Load data' button is visible. To the right of the table, a context menu is open, with the 'Show Output Table' option checked. Other options in the menu include 'Show Variables', 'Show Configuration', 'Show Entire Configuration', 'Show Node Timing Information', and 'Show Graph Annotations'. The main area displays a table of customer data:

ID	CustomerID	MaritalStatus	Gender	EstimatedYearlyIncome	NumberOfContracts	Age	Available401K	CustomerV	Products
	722204	S	F	80000	4	42	1	1	4
	489847	M	M	60000	2	46	1	1	4
	8444723	M	M	40000	1	32	1	2	3
	1487427	M	M	30000	2	63	1	1	2
	4693433	M	M	20000	2	63	1	1	3
	7724940	M	M	30000	2	33	1	2	3
	9784443	M	M	60000	2	34	1	2	3
	3177757	M	M	70000	2	57	1	1	5

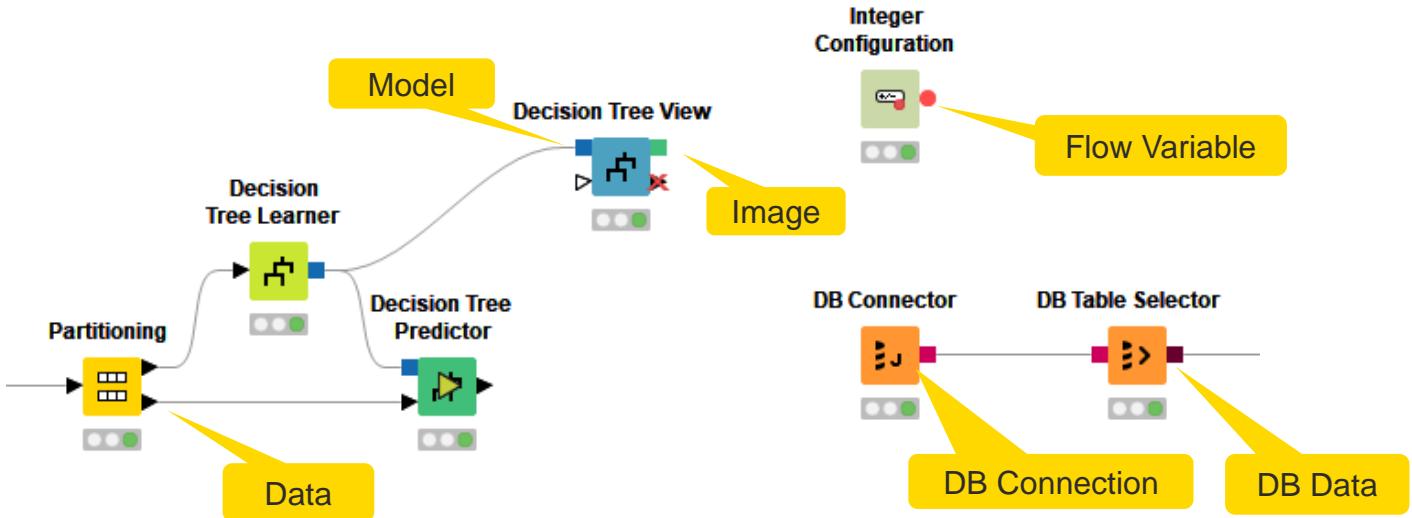
Console and Other Views



- Console view prints out error and warning messages about what is going on under the hood
- Click View and select Other... to add different views
 - Node Monitor, Licenses, etc.

Inserting and Connecting Nodes

- Insert nodes into workspace by dragging them from the Node Repository or by double-clicking in the Node Repository
- Connect nodes by left-clicking the output port of Node A and dragging the cursor to the (matching) input port of Node B
- Common port types:



More on Nodes...

- A node can have 4 states:

File Reader



Not Configured:

The node is waiting for configuration or incoming data.

File Reader



Configured:

The node has been configured correctly, and can be executed.

File Reader



Executed:

The node has been successfully executed. Results may be viewed and used in downstream nodes.

File Reader

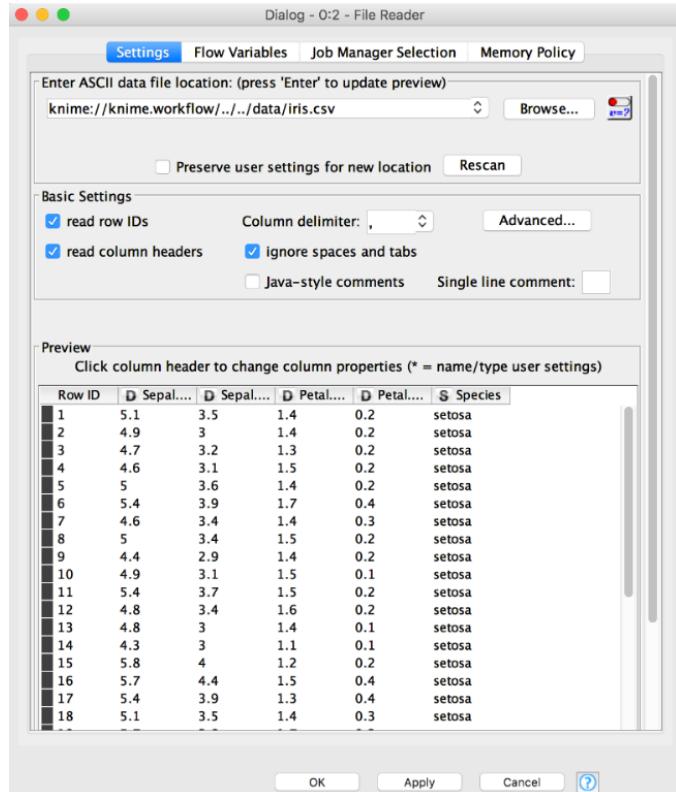


Error:

The node has encountered an error during execution.

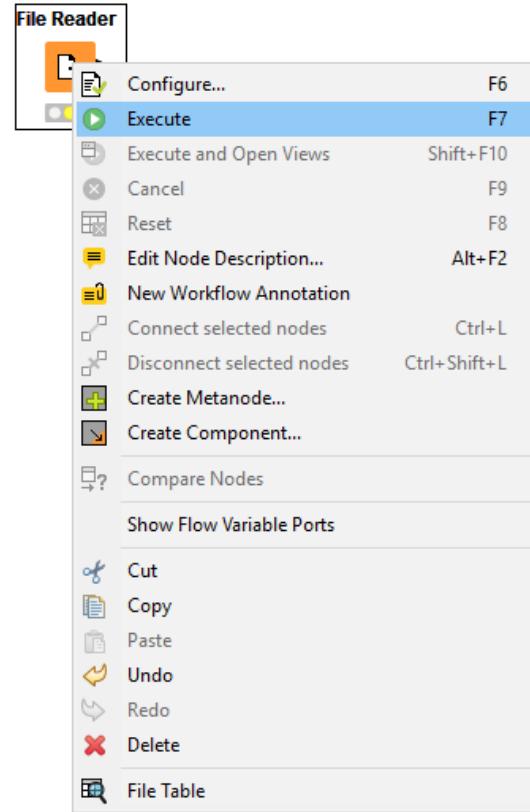
Node Configuration

- Most nodes need to be configured
- To access a node configuration dialog:
 - Double-click the node
 - Right-click -> Configure



Node Execution

- Right-click node
- Select Execute in the context menu
- If execution is successful, status shows green light
- If execution encounters errors, status shows red light



Tool Bar

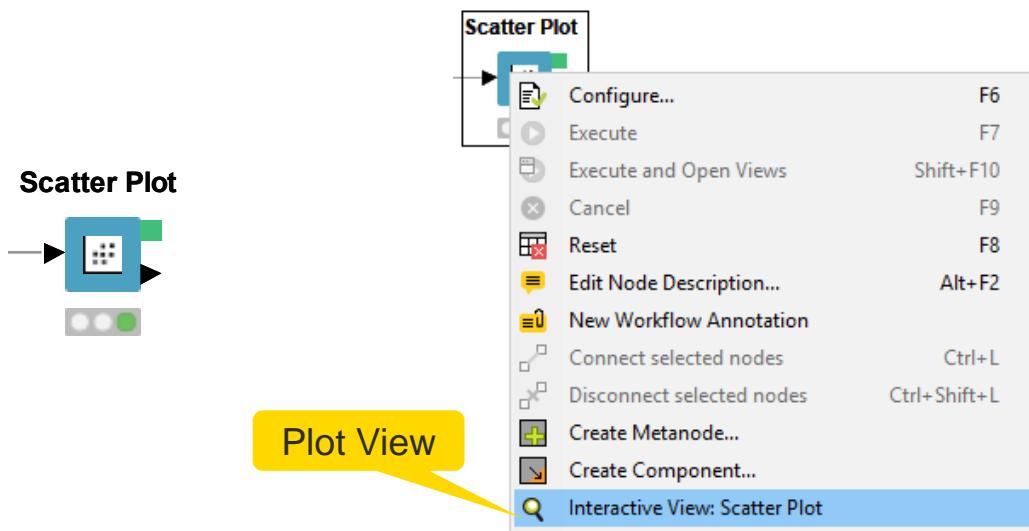


The buttons in the toolbar can be used for the active workflow. The most important buttons are:

- Execute selected and executable nodes (F7)
- Execute all executable nodes
- Execute selected nodes and open first view
- Cancel all selected, running nodes (F9)
- Cancel all running nodes

Node Views

- Right-click node to inspect the execution results by
 - selecting output ports (last option in the context menu) to inspect tables, images, etc.
 - selecting Interactive View to open visualization results in a browser



Configure...	F6
Execute	F7
Execute and Open Views	Shift+F10
Cancel	F9
Reset	F8
Edit Node Description...	Alt+F2
New Workflow Annotation	
Connect selected nodes	Ctrl+L
Disconnect selected nodes	Ctrl+Shift+L
Create Metanode...	
Create Component...	
Interactive View: Scatter Plot	
Compare Nodes	
Show Flow Variable Ports	
Cut	
Copy	
Paste	
Undo	
Redo	
Delete	
Image	
Input data and view selection	

KNIME File Extensions

Dedicated file extensions for workflows and workflow groups associated with KNIME Analytics Platform

- ***.knwf** for KNIME Workflow Files



- ***.knar** for KNIME Archive Files



Getting Started: KNIME Hub

- Place to search and share
 - Workflows
 - Nodes
 - Components
 - Extensions

The screenshot shows the KNIME Hub search interface with the query "Sentiment Analysis". It displays 350 results. The results are categorized by type: All, Nodes, Components, Workflows, and Extensions. Three specific workflows are listed:

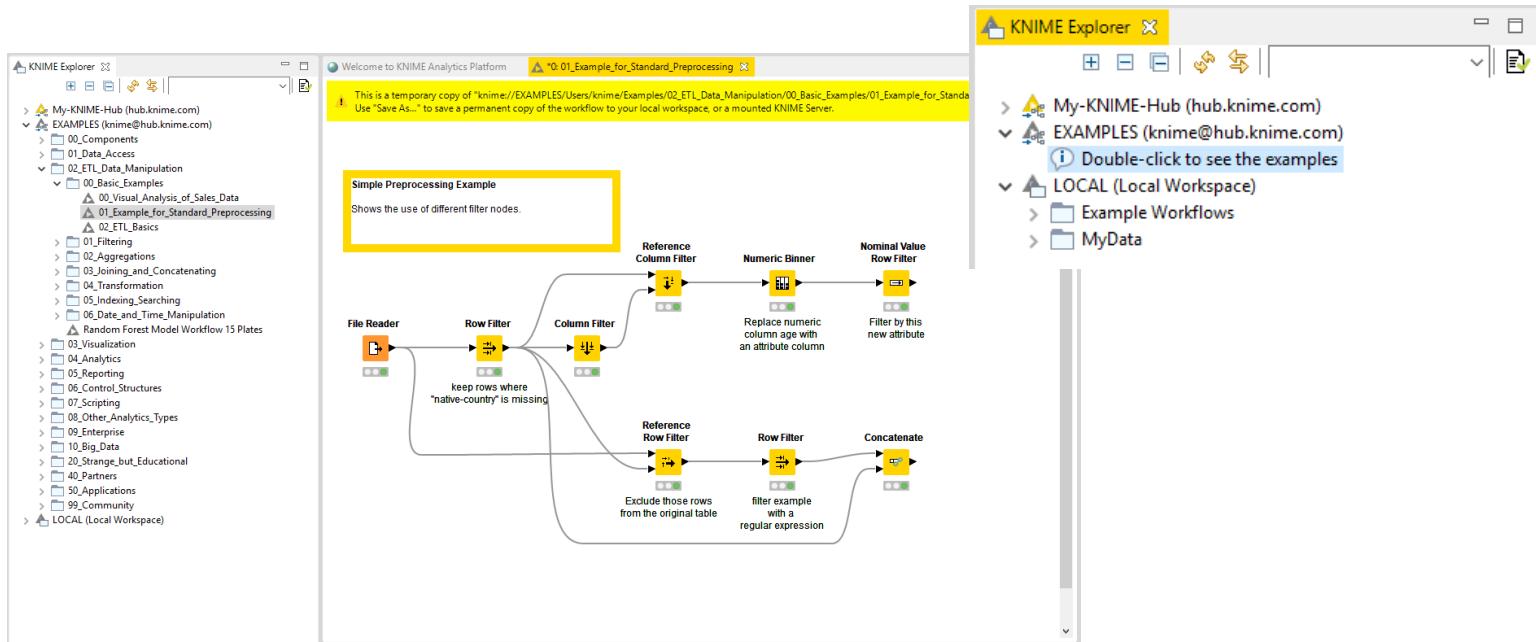
- Sentiment Analysis**: This workflow shows how to train a simple neural network for text classification, in this case sentiment analysis. The used network learns a 128 dimensional word embedding followed by an LSTM. This example is located at `knime > Examples > 04_Analytics > 14_Deep_Learning > 02_Keras > 07_Sentiment_Analysis_with_Deep_Learning_NODES`.
- Sentiment Analysis**: This workflow shows how to train a simple neural network for text classification, in this case sentiment analysis. The used network learns a 128 dimensional word embedding followed by an LSTM. This example is located at `knime > Examples > 04_Analytics > 14_Deep_Learning > 02_Keras > 07_Sentiment_Analysis_with_Deep_Learning`.
- Sentiment Analysis (Classification) of Documents**: This workflow shows how to import text from a csv file, convert it to documents, preprocess the documents and transform them.

The screenshot shows the main homepage of the KNIME Hub. It features a welcome message: "Welcome to the KNIME Hub. The place to find and collaborate on KNIME workflows and nodes. Here you can find solutions for your data science questions." Below this is a search bar with the placeholder "Search workflows, nodes and more...". To the right, there are statistics: 3 993 Nodes, 265 Components, 2 541 Workflows, and 211 Extensions. There are also two cards: one for "How to Getting started From downloading through to building your first workflow" and another for "Forum Get help from our community and help others".

<https://hub.knime.com>

Getting Started: KNIME Example Server

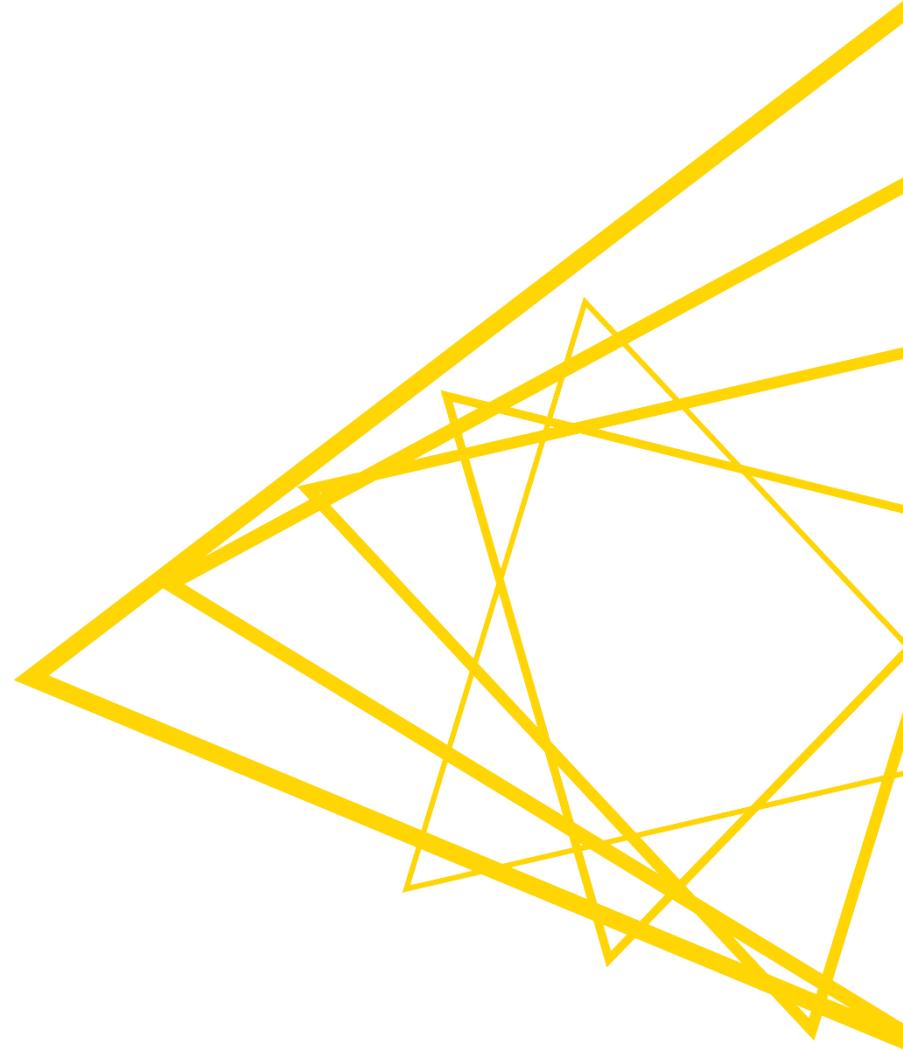
- Connect via KNIME Explorer to a public repository with large selection of example workflows for many, many applications



Hot Keys (for Future Reference)

Task	Hot key	Description
Node Configuration	F6	opens the configuration window of the selected node
	F7	executes selected configured nodes
	Shift + F7	executes all configured nodes
Node Execution	Shift + F10	executes all configured nodes and opens all views
	F9	cancels selected running nodes
	Shift + F9	cancels all running nodes
Node Connections	Ctrl + L	connects selected nodes
	Ctrl + Shift + L	disconnects selected nodes
Move Nodes and Annotations	Ctrl + Shift + Arrow	moves the selected node in the arrow direction
	Ctrl + Shift + PgUp/PgDown	moves the selected annotation in the front or in the back of all overlapping annotations
Workflow Operations	F8	resets selected nodes
	Ctrl + S	saves the workflow
	Ctrl + Shift + S	saves all open workflows
	Ctrl + Shift + W	closes all open workflows
Metanode	Shift + F12	opens metanode wizard

Introduction to the Big Data Course



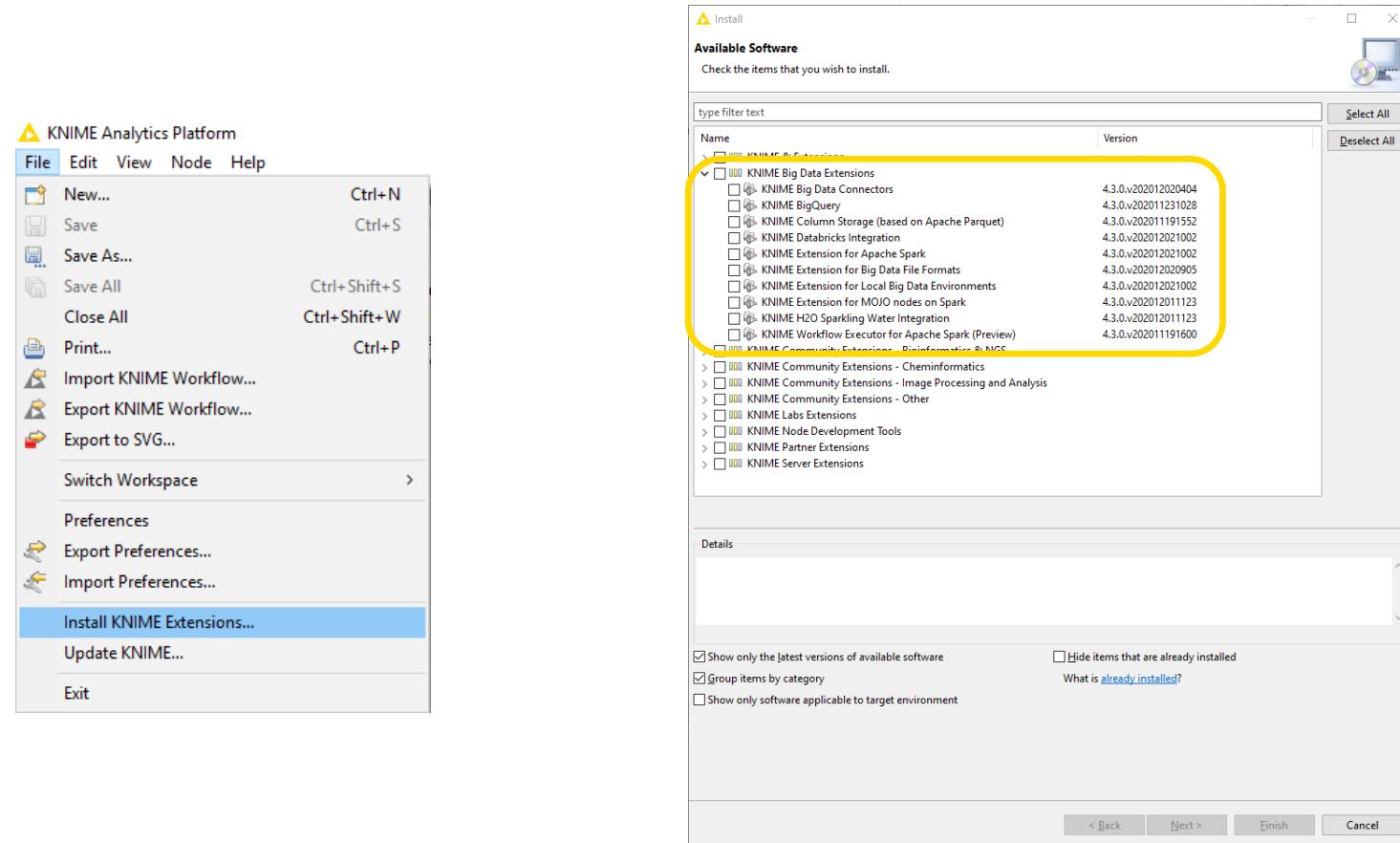
Goal of this Course

Become familiar with the **Hadoop Ecosystem** and the
KNIME Big Data Extensions

What you need:

- KNIME Analytics Platform with
 - KNIME Big Data Extensions
 - KNIME Big Data Connectors
 - KNIME Extension for Apache Spark
 - KNIME Extension for Local Big Data Environment
 - KNIME File Handling Nodes

Installation of Big Data Extensions



...or install via drag-and-drop from the KNIME Hub

The image shows two side-by-side screenshots of the KNIME environment. On the left, the KNIME Hub search results page is displayed, showing 3896 results. The search bar contains the placeholder "Search workflows, nodes and more...". Below the search bar, there are filters for "All", "Nodes", and "Workflows". Three workflow cards are visible: "Model building 1", "Generate artificial association rules into existing shopping baskets", and "Access YouTube REST API". Each card has a preview thumbnail, a title, a brief description, and a "View details" button. On the right, the KNIME Analytics Platform interface is shown, displaying a workflow titled "O: My Analysis". The workflow consists of several nodes connected by arrows: a "File Reader" node feeds into a "Partitioning" node, which then feeds into a "Random Forest Learner" node. The output of the "Random Forest Learner" node goes to a "Random Forest Predictor" node, which then feeds into a "Scorer" node. The "Scorer" node has a small preview window showing some data.

Big Data Resources (1)

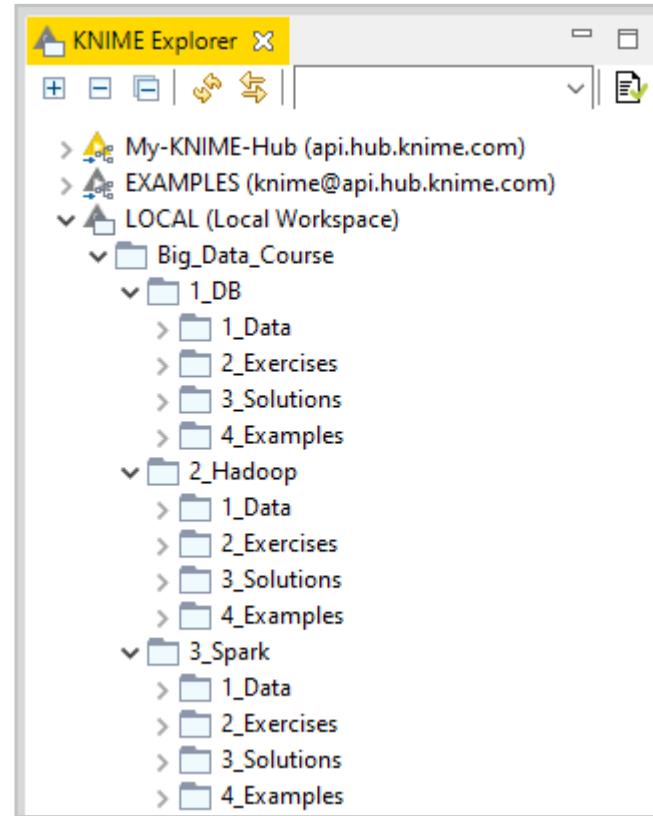
- SQL Syntax and Examples
<https://www.w3schools.com>
- Apache Spark MLlib
<https://spark.apache.org/docs/latest/ml-guide.html>
- KNIME Big Data Extensions (Hadoop + Spark)
<https://www.knime.com/knime-big-data-extensions>
- Example workflows on KNIME Hub
<https://www.knime.com/nodeguide/big-data>

Big Data Resources (2)

- Whitepaper “KNIME opens the Doors to Big Data”
https://www.knime.com/sites/default/files/inline-images/big_data_in_knime_1.pdf
- Blog Posts
 - <https://www.knime.org/blog/Hadoop-Hive-meets-Excel>
 - <https://www.knime.com/blog/SparkSQL-meets-HiveSQL>
 - <https://www.knime.com/blog/speaking-kerberos-with-knime-big-data-extensions>
 - <https://www.knime.com/blog/new-file-handling-out-of-labs-and-into-production>
- Video
 - <https://www.knime.com/blog/scaling-analytics-with-knime-big-data-extensions>

Overview

1. Use a traditional Database, and KNIME Analytics Platform native Machine Learning Nodes
2. Moving In-Database Processing to Hadoop Hive
3. Moving In-Database Processing and Machine Learning to Spark



Today's Example: Missing Values Strategy

- Missing Values are a big problem in Data Science!
- Many strategies to deal with the problem (see “How to deal with missing values” KNIME Blog <https://www.knime.com/blog/how-to-deal-with-missing-values?>)
- We adopt the strategy that predicts the missing values based on the other attributes on the same data row
- CENSUS Data Set with missing COW values from
<http://www.census.gov/programs-surveys/acs/data/pums.html>
<https://www.census.gov/programs-surveys/acs/microdata/documentation.html>

CENSUS Data Set

- CENSUS data contains questions to a sample of US residents (1%) over 10 years
- CENSUS data set description: http://www2.census.gov/programs-surveys/acs/tech_docs/pums/data_dict/PUMSDataDict15.pdf
- *ss13hme* (60K rows) -> questions about housing to Maine residents
- *ss13pme* (60K rows) -> questions about themselves to Maine residents
- *ss13hus* (31M rows) -> questions about housing to all US residents in the sample
- *ss13pus* (31M rows) -> questions about themselves to all US residents in the sample

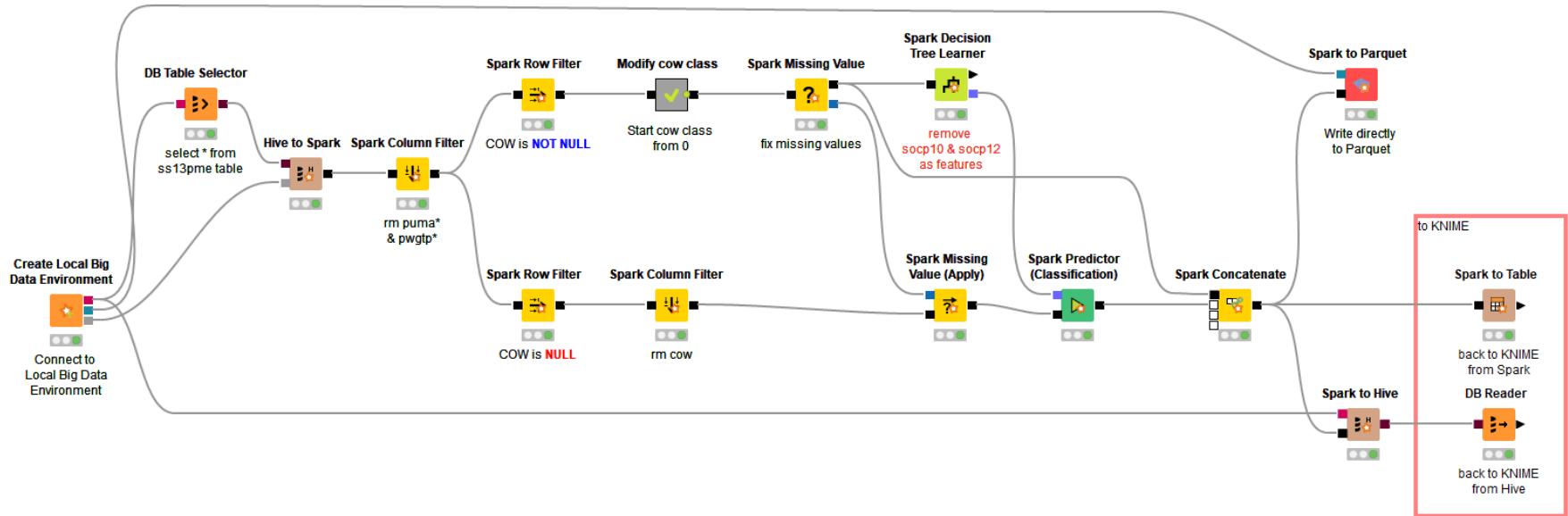
https://www2.census.gov/programs-surveys/acs/tech_docs/pums/data_dict/PUMS_Data_Dictionary_2021.pdf

Missing Values Strategy Implementation

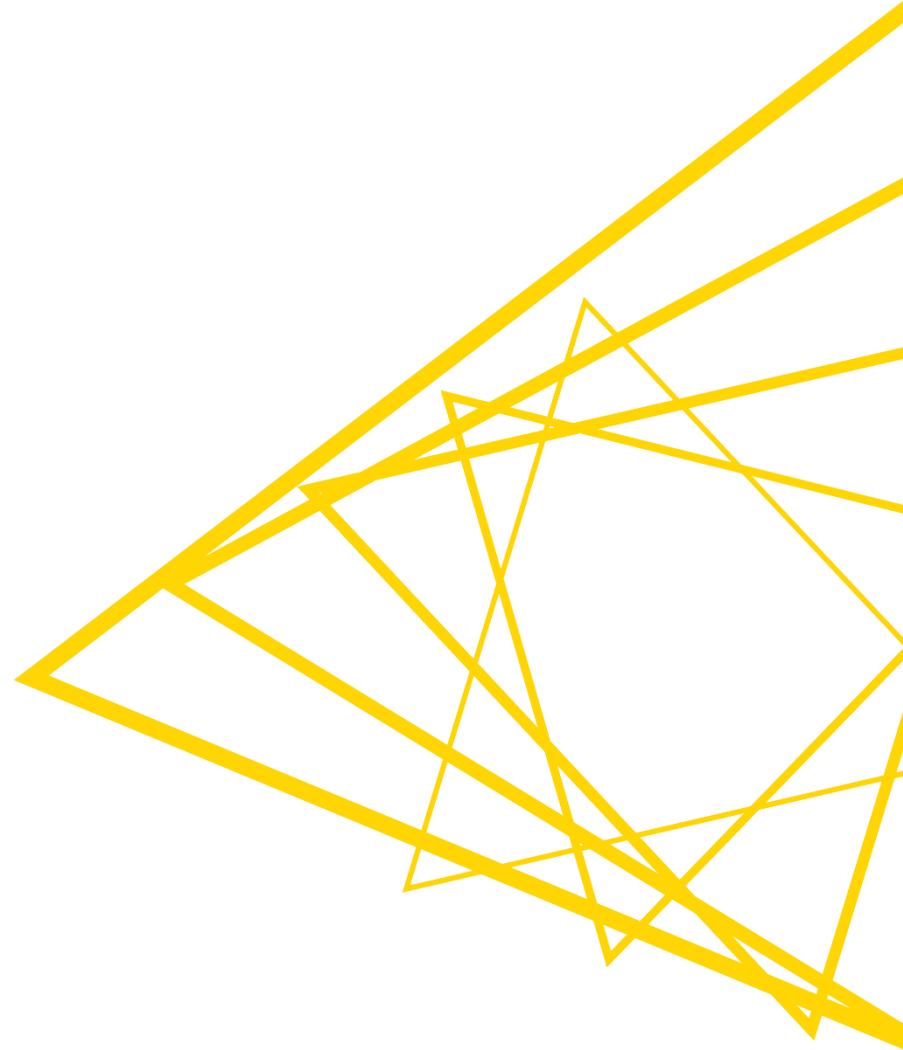
- Connect to the CENSUS data set
- Separate data rows with COW from data rows with missing COW
- Train a decision tree to predict COW (obviously only on data rows with COW)
- Apply decision tree to predict COW where COW is missing
- Update original data set with new predicted COW values

COW... Class of Worker - tipo de vínculo empregatício

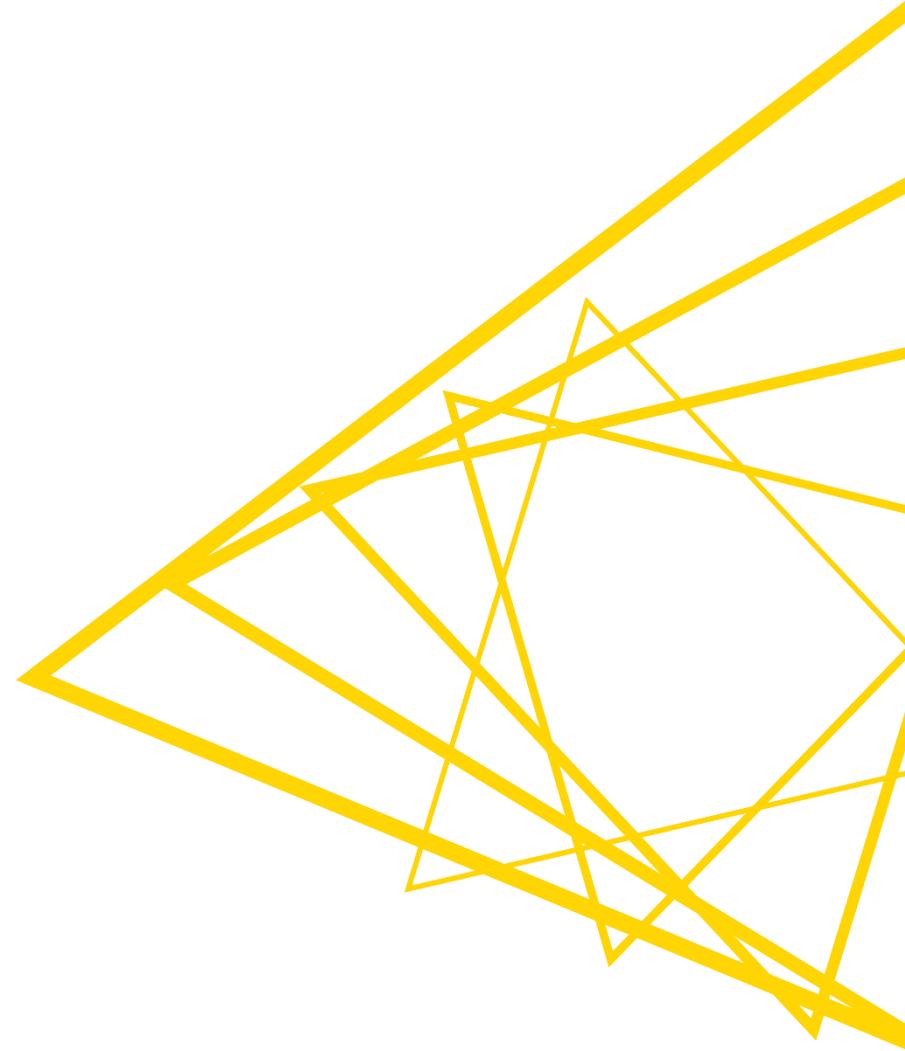
Today's Example: Missing Values Strategy



**Let's Practice First on
a Traditional Database**

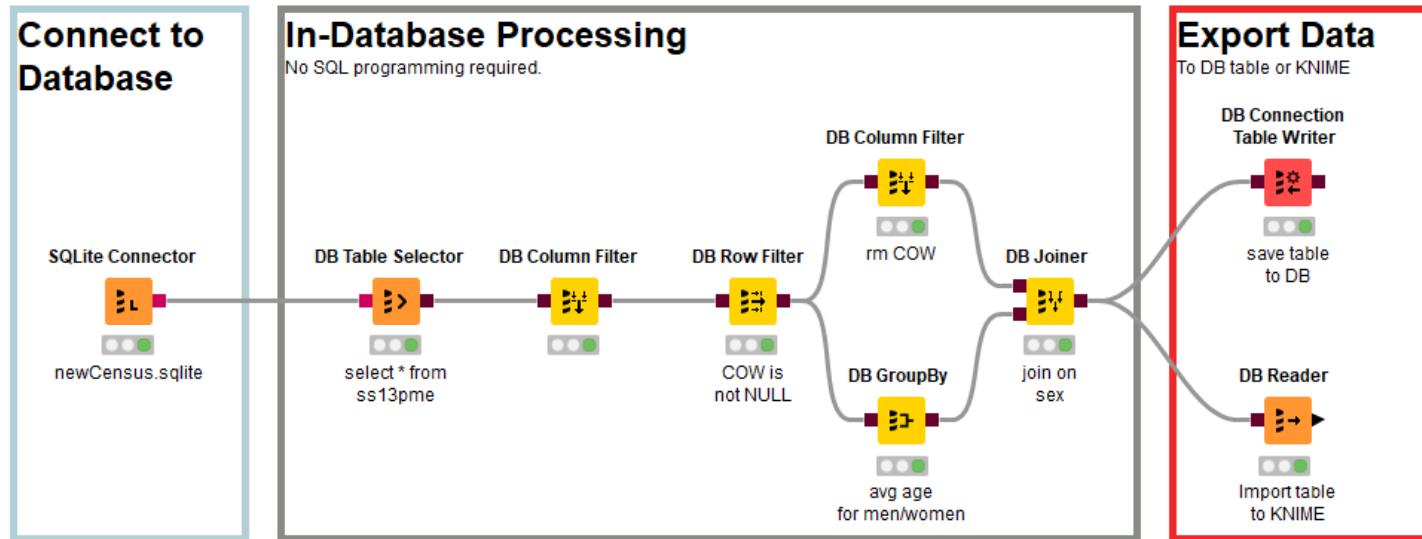


Database Extension



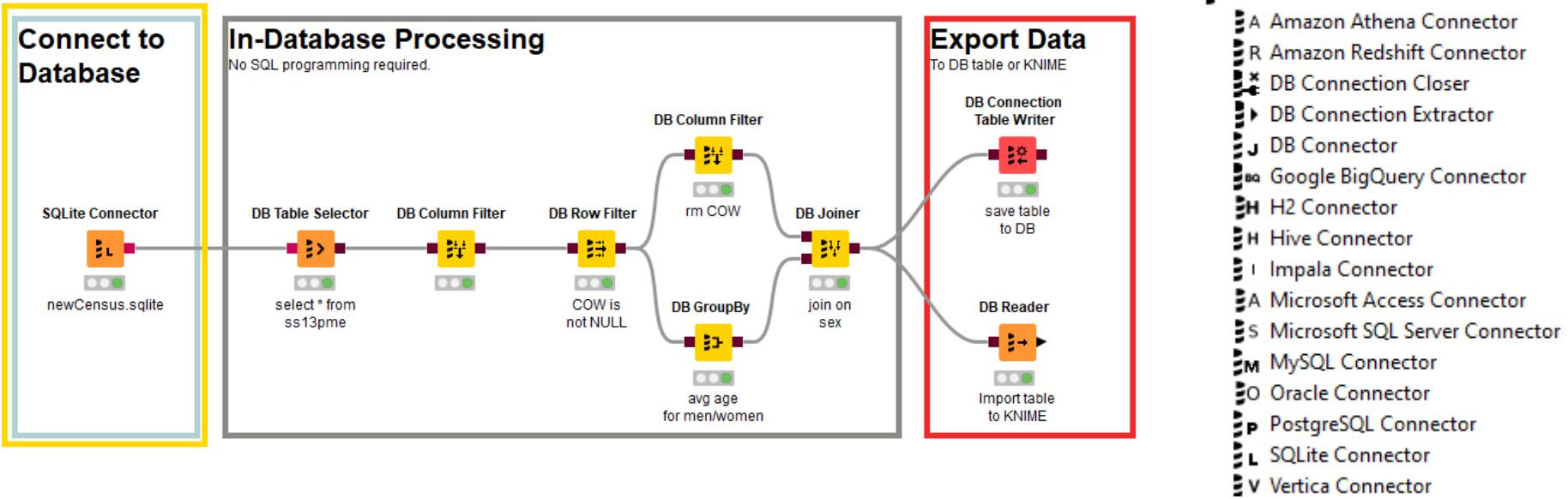
Database Extension

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



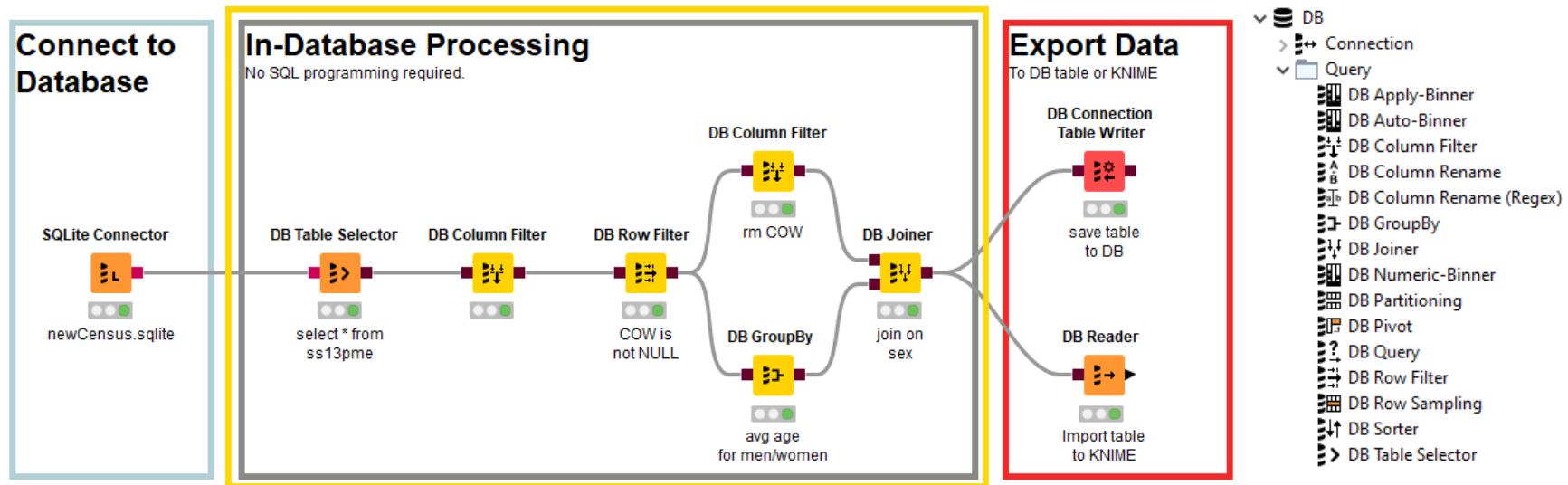
Database Connectors

- Many dedicated DB Connector nodes available
- If connector node missing, use DB Connector node with JDBC driver



In-Database Processing

- Database Manipulation nodes generate SQL query on top of the input SQL query (brown square port)
- SQL operations are executed on the database!



Export Data

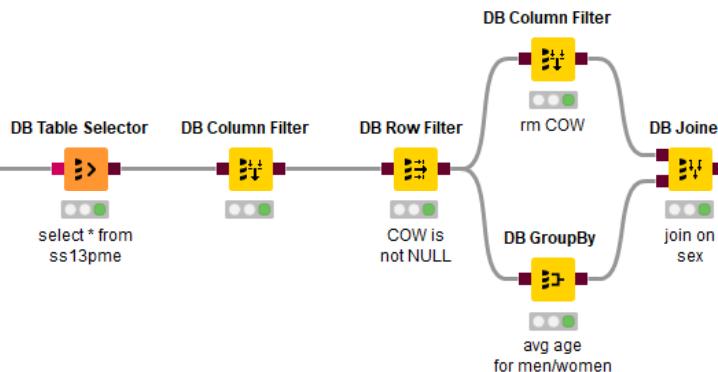
- Writing data back into database
- Exporting data into KNIME

Connect to Database

SQLite Connector
newCensus.sqlite

In-Database Processing

No SQL programming required.



Export Data

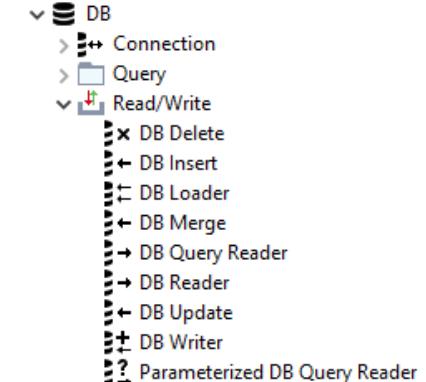
To DB table or KNIME

DB Connection Table Writer

save table to DB

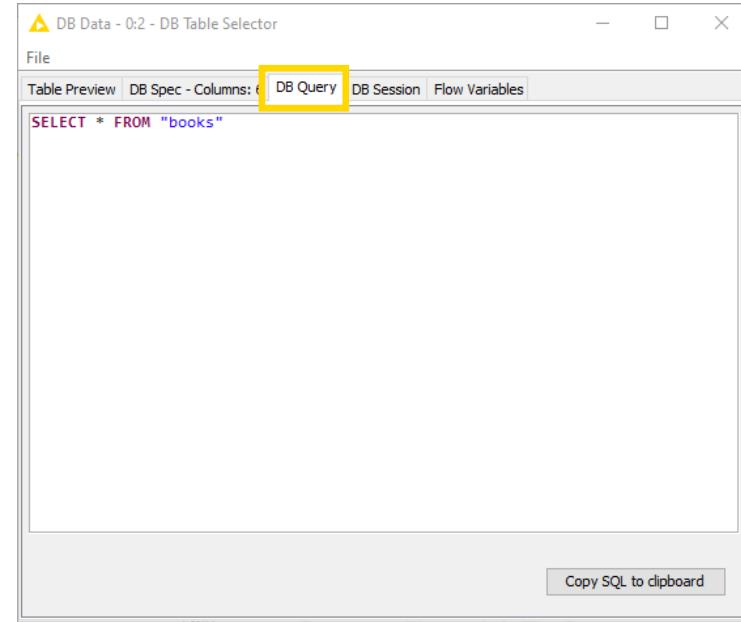
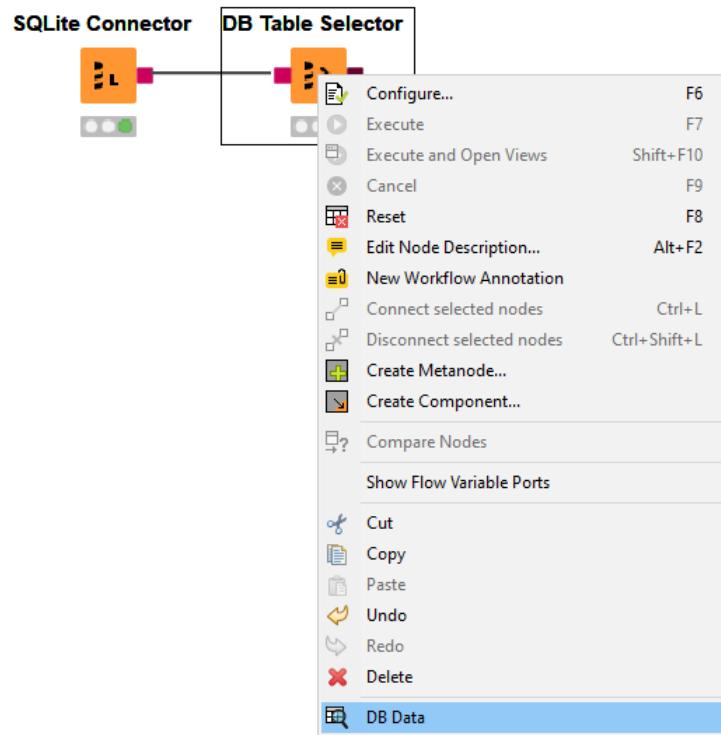
DB Reader

Import table to KNIME

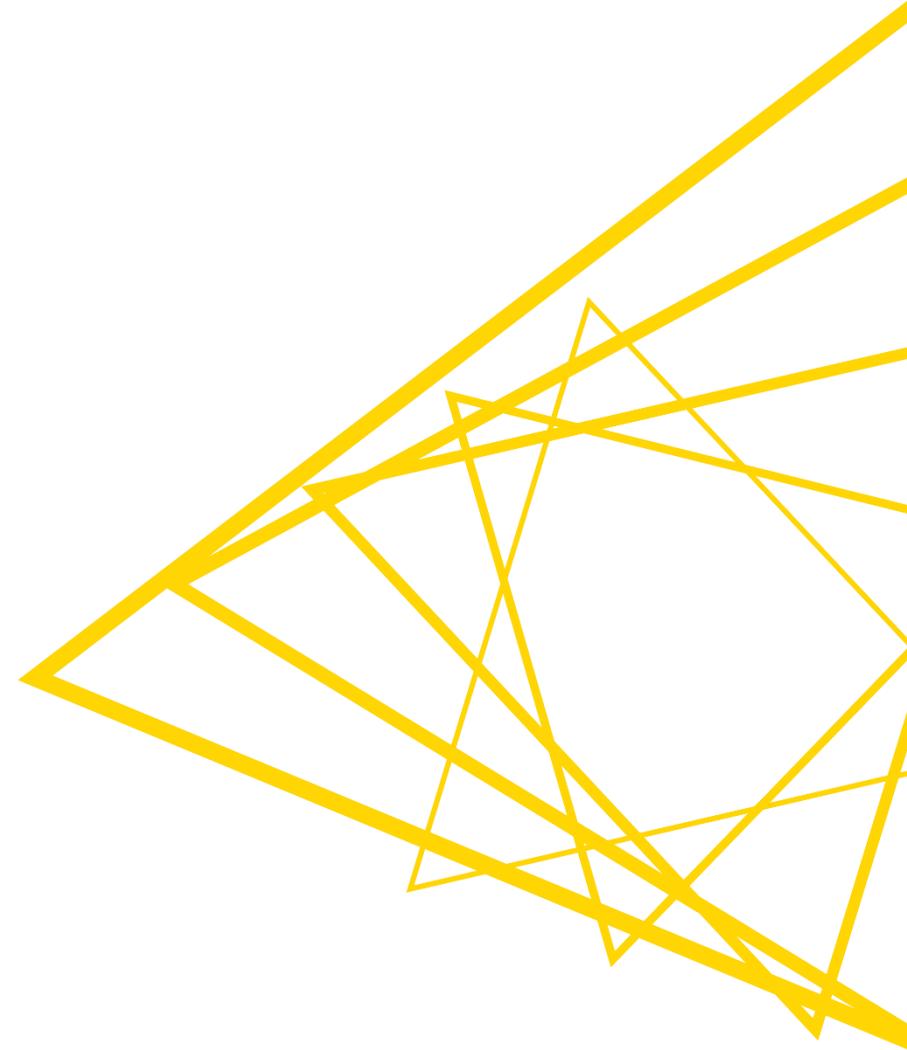


Tip

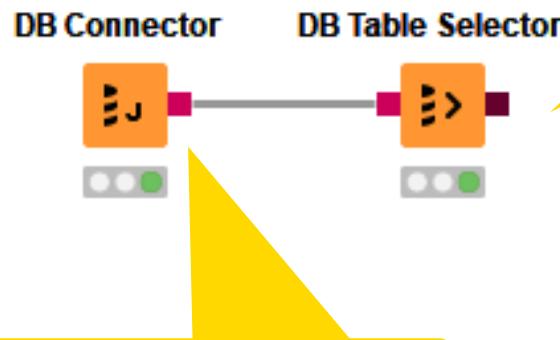
- SQL statements can be easily viewed from the DB node output ports.



Database Port Types



Database Port Types



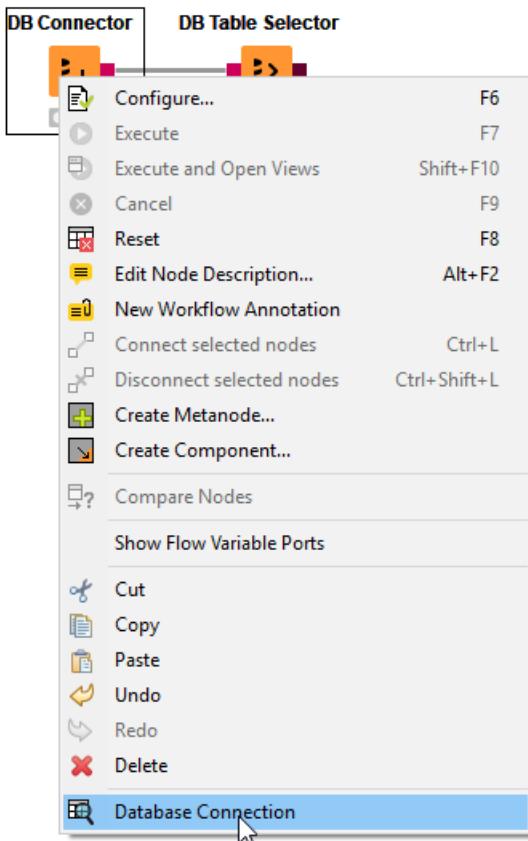
Database Connection Port (brown)

- Connection information
- SQL statement

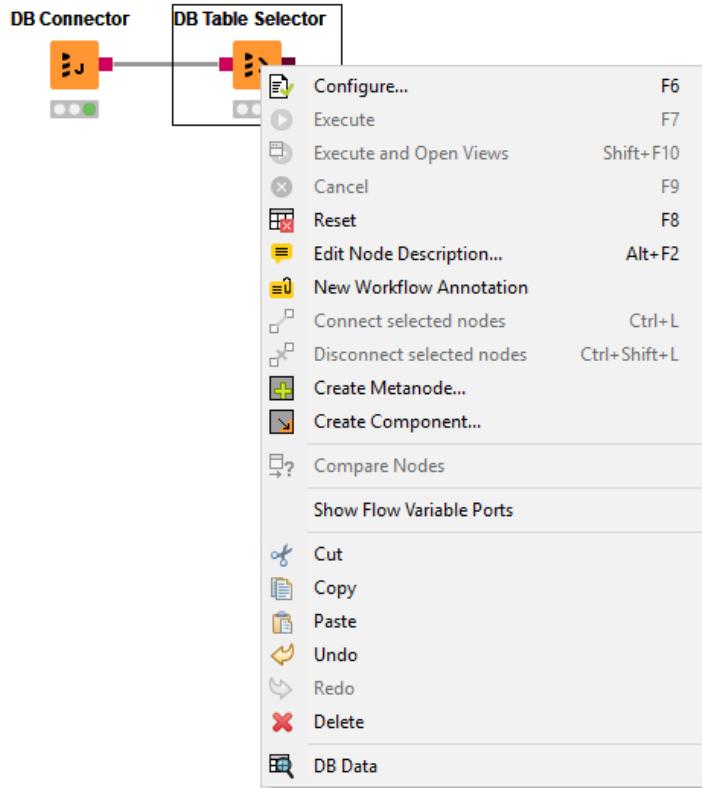
Database JDBC Connection Port (red)

- Connection information

DB Connection Port View



DB Data Port View



The screenshot shows the DB Data - 0:23 - DB Table Selector window. The "Table Preview" tab is selected, displaying a preview of 13 columns from 100 rows. The columns include Row ID, \$assemb..., \$gca_accession, \$commo..., Itaxid, \$ensembl_url, and \$ensembl_id.

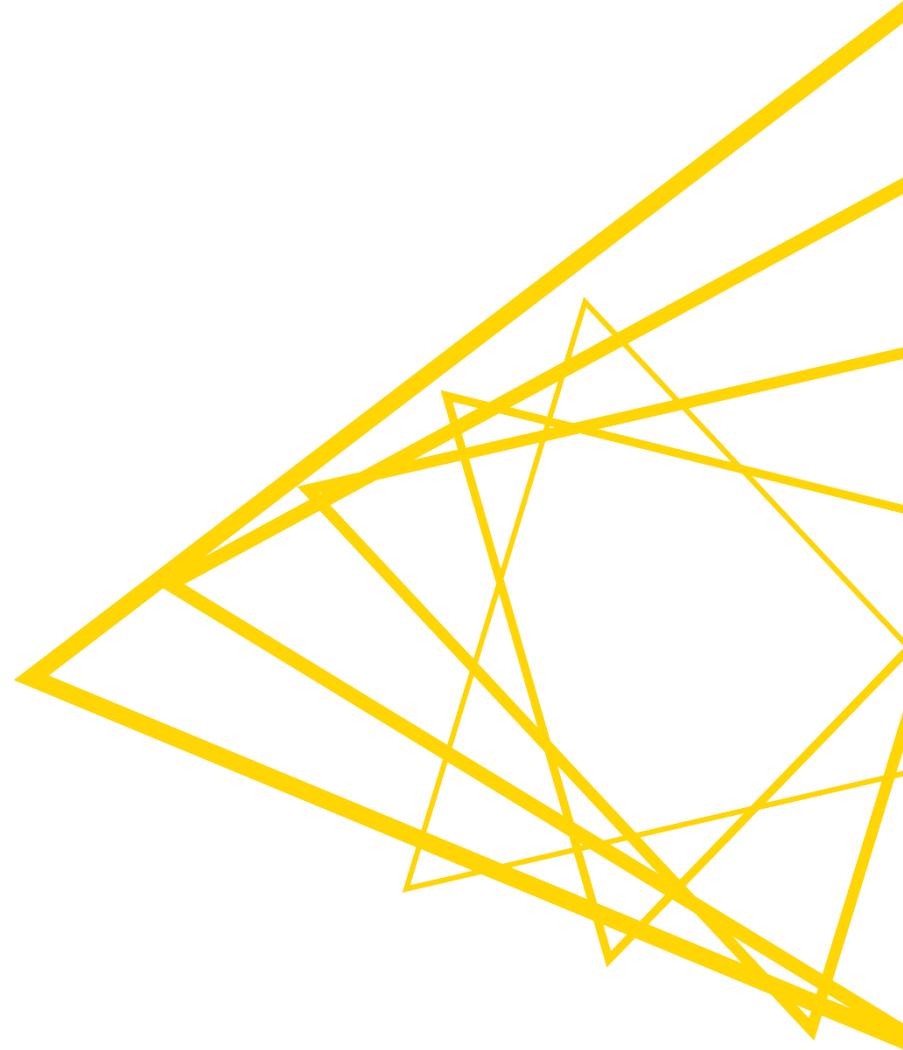
Row ID	\$assemb...	\$assemb...	\$gca_accession	\$assemb...	\$commo...	Itaxid	\$ensembl_url	\$ensembl_id
Row0	ASM1495v1	ASM1495v1	GCA_000149555.1	?	?	334819	fusarium_verticillioides	Er
Row1	Phyt_para...	Phyt_para...	GCA_000365505.1	?	?	1317065	phytophthora_paras...	Er
Row2	Emiliana_hux...	Emiliana_hux...	GCA_000372725.1	?	?	280463	emiliana_huxleyi	Er
Row3	ASMS8656v1	ASMS8656v1	GCA_008586565.1	?	Chinese me...	183150	oryzias_sinensis	Er
Row4	H_comes_Q...	H_comes_Q...	GCA_001891065.1	?	tiger tail sea...	109280	hippocampus_comes	Er
Row5	dgr1_caf1	dgr1_caf1	GCA_000005155.1	?	?	7222	drosophila_grimshawi	Er
Row6	SaiBo1.0	SaiBo1.0	GCA_000235385.1	?	Bolivian squi...	39432	saimiri_boliviensis_b...	Er
Row7	OchPri2.0-Ens	OchPri2.0-Ens	?	ochPri2	American pika	9978	ochotona_princeps	Er
Row8	Oryza_brac...	Oryza_brac...	GCA_000231095.2	?	malo sina	4533	oryza_brahyantha	Er
Row9	DMR_v1.0	DMR_v1.0	GCA_000743615.1	?	Damara mol...	885580	fukomys_damarensis	Er
Row10	SAMN03320...	SAMN03320...	GCA_001515625.1	?	horned gold...	307959	sinocyclocheilus_rhin...	Er
Row11	fDenClu							
Row12	Meloposit							
Row13	AaloF1							
Row14	STF_HIC							

The screenshot shows the DB Data - 0:23 - DB Table Selector window with the "DB Query" tab selected. The query is:

```
SELECT * FROM "rnacen"."ensembl_assembly"
```

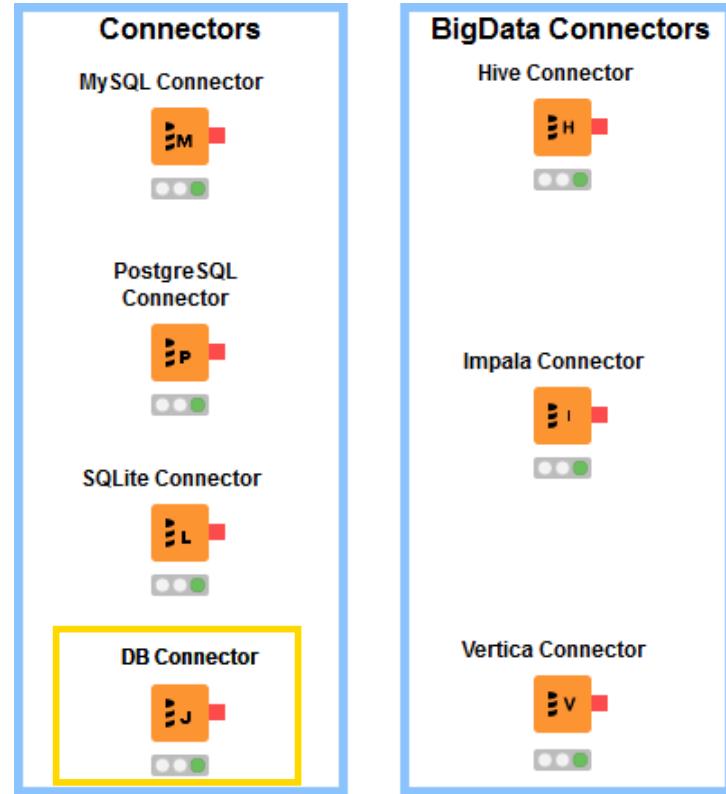
A yellow callout bubble points to the SQL statement with the text "Copy SQL statement".

Connect to Database and Import Data

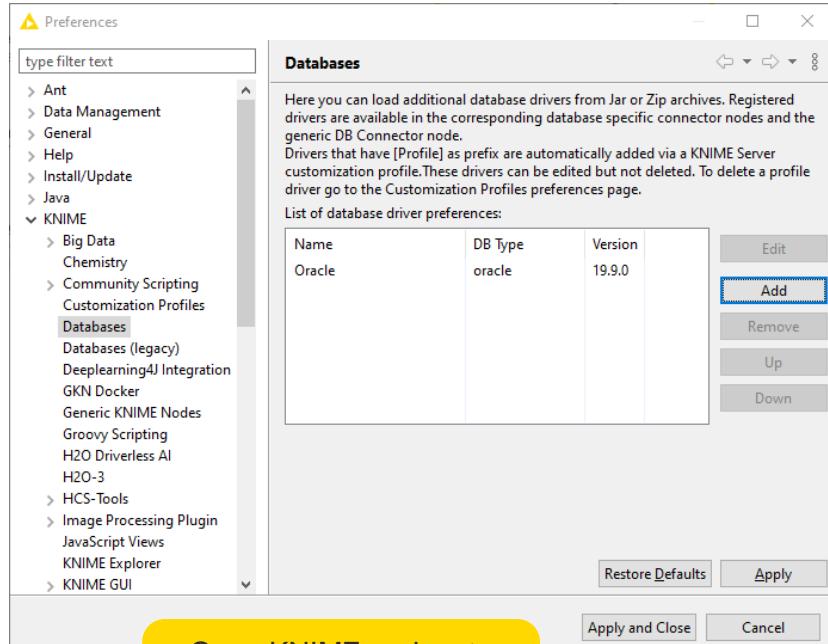


Database Connectors

- 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



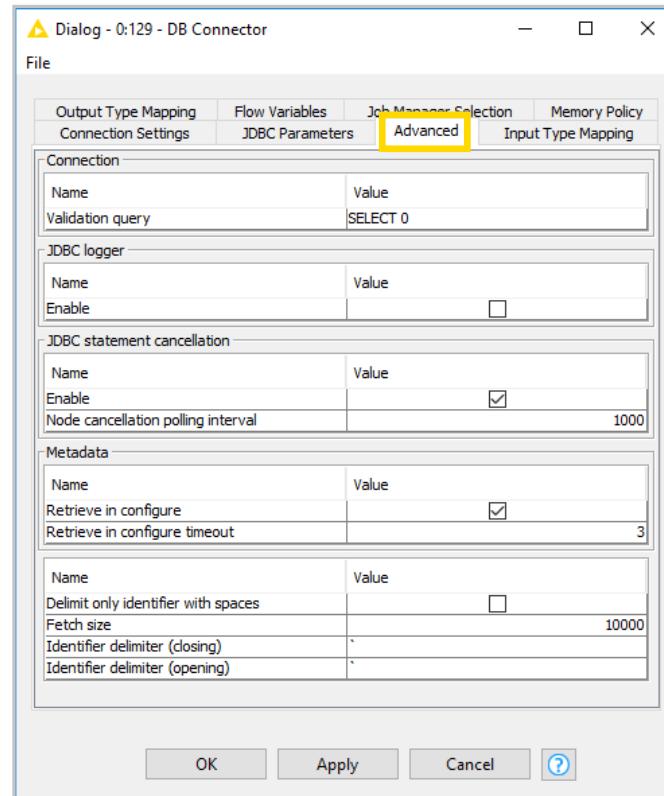
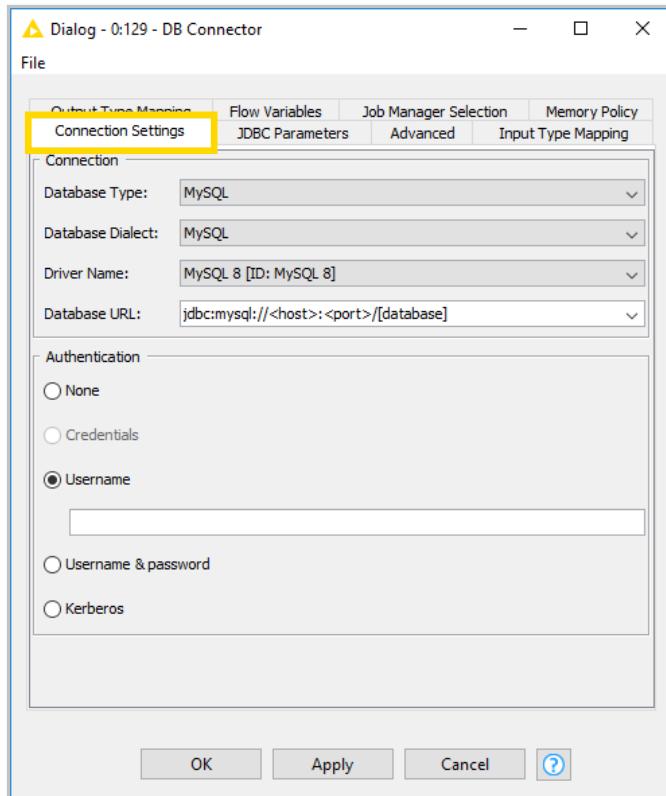
Register JDBC Driver



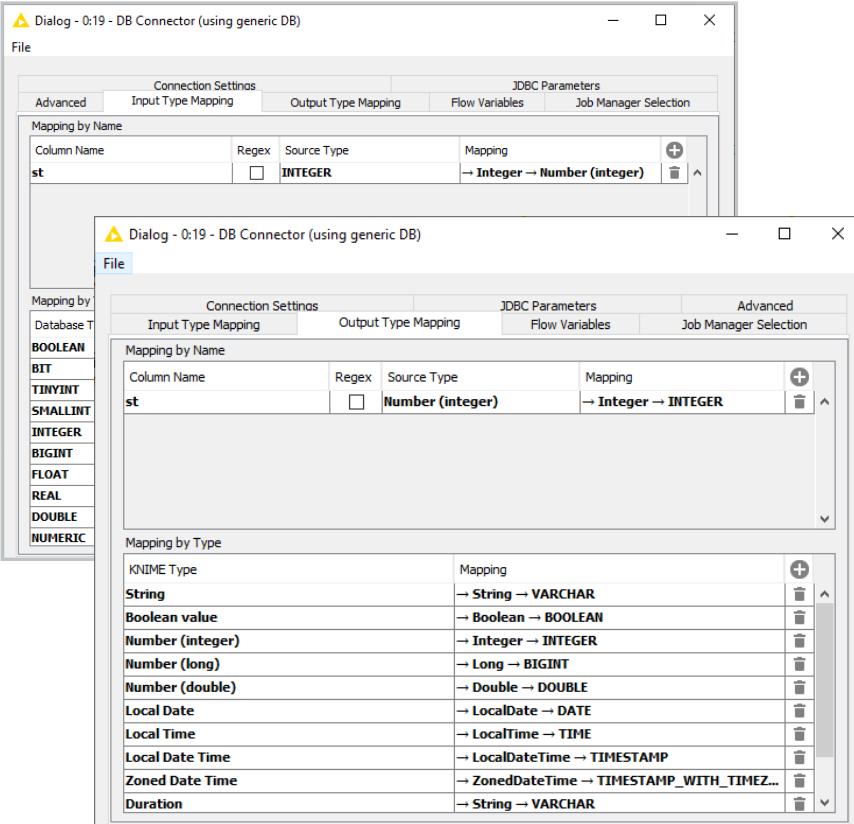
Open KNIME and go to
File -> Preferences, then
KNIME -> Databases



DB Connector Node



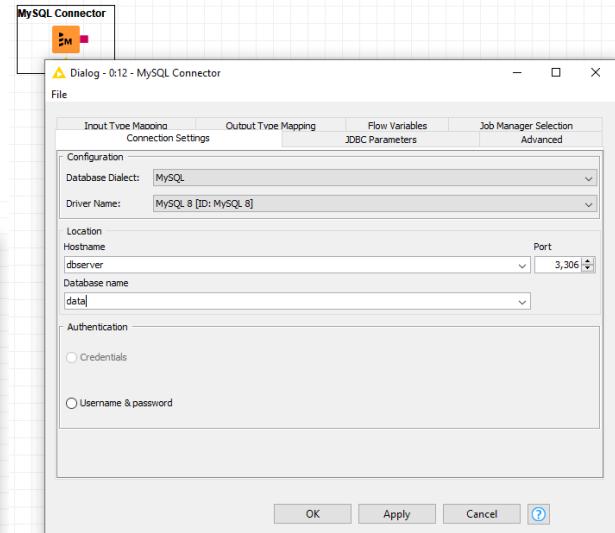
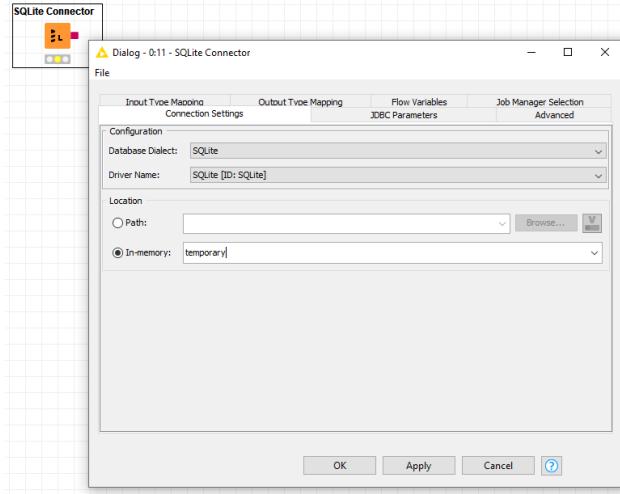
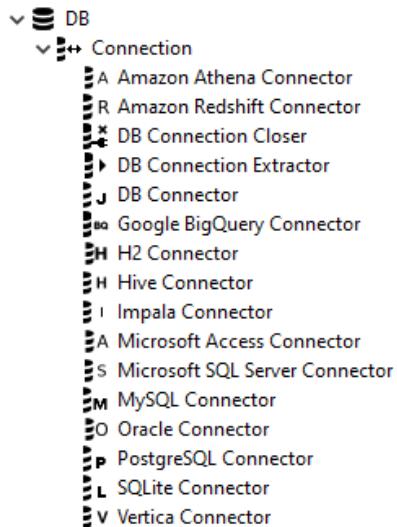
DB Connector Node – Type Mapping



- KNIME will do its best to guess what type mappings are appropriate based on what it knows about your database
- If you need more control, you can specify type mappings manually in two ways
 - By name, for individual fields – or groups of fields using RegEx
 - By type
- Two separate tabs to govern input and output type mappings

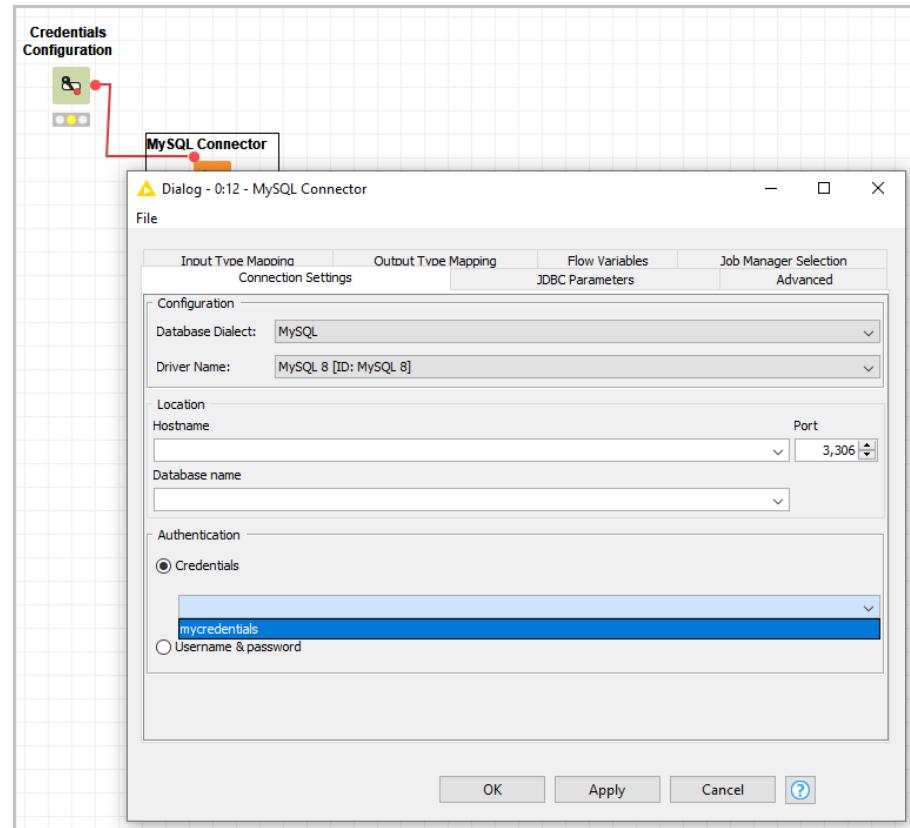
Dedicated Database Connectors

- MS SQL Server, MySQL, Postgres, SQLite, ...
- Propagate connection information to other DB nodes



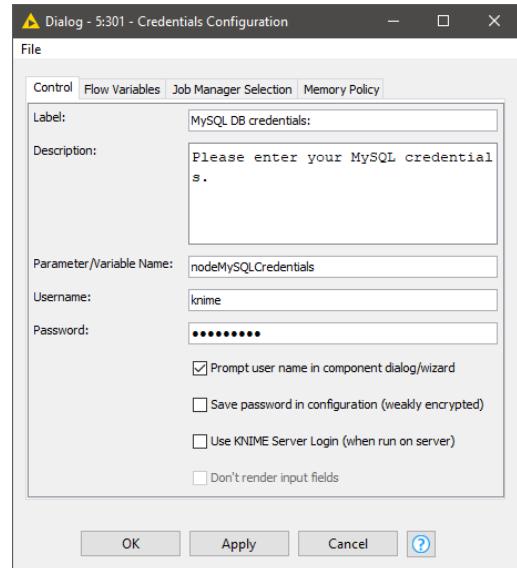
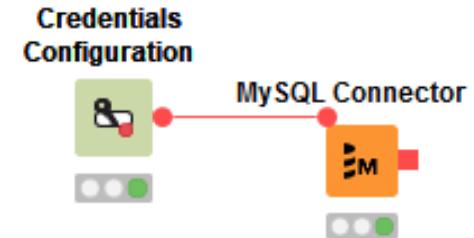
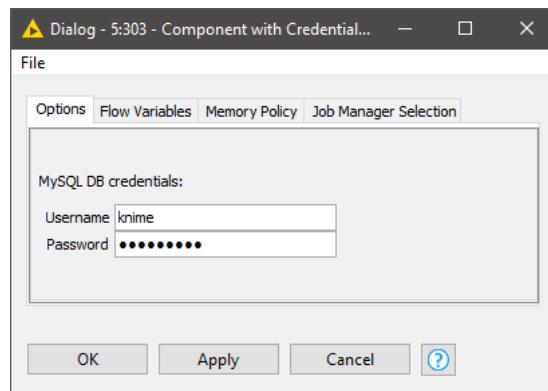
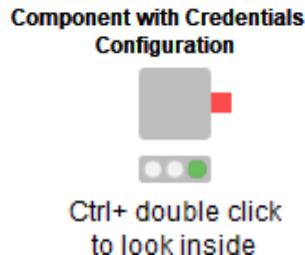
Workflow Credentials – Usage

- Replaces username and password fields
- Supported by several nodes that require login credentials
 - DB connectors
 - Remote file system connectors
 - Send mail
 - ...



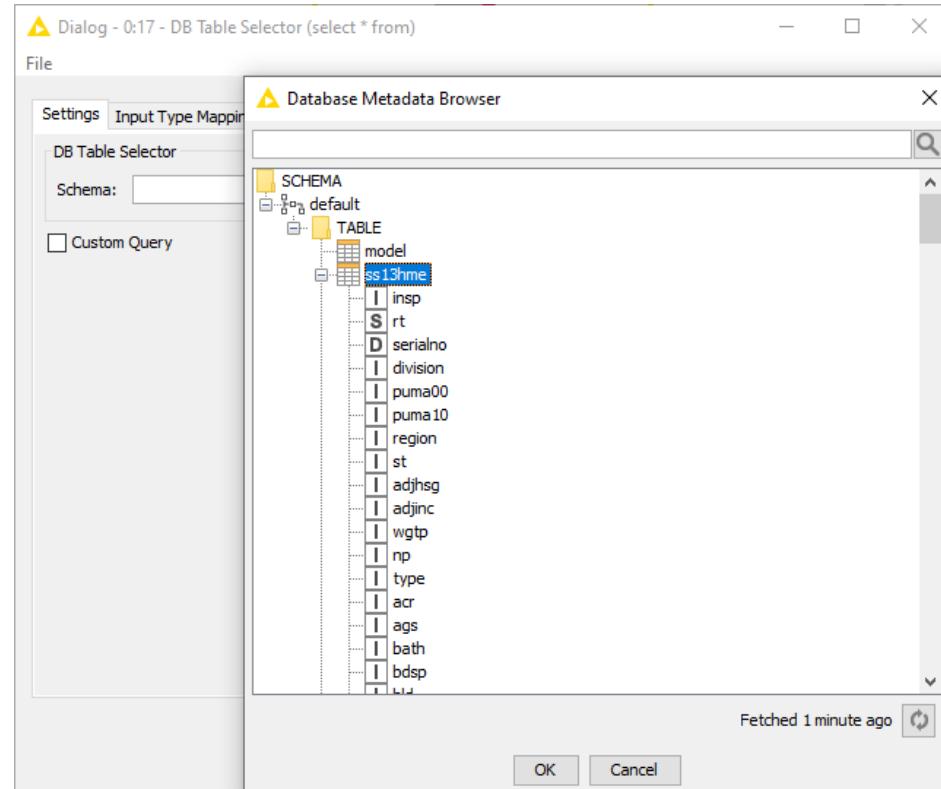
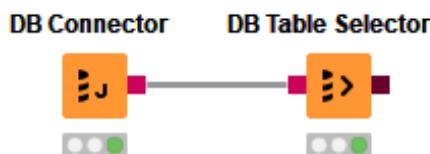
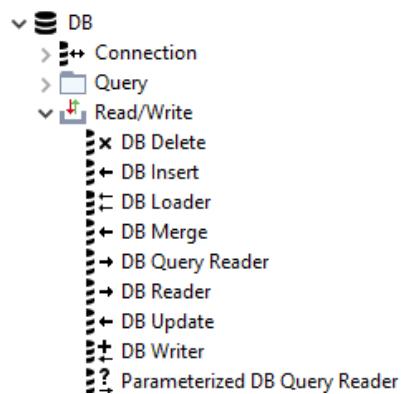
Credentials Configuration Node

- Works together with all nodes that support workflow credentials



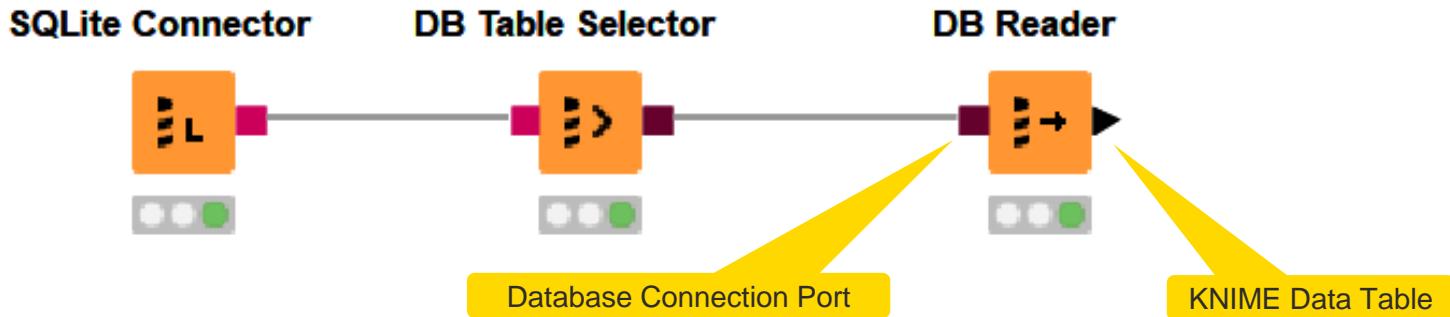
DB Table Selector Node

- Takes connection information and constructs a query
- Explore DB metadata
- Outputs a SQL query



DB Reader node

- Executes incoming SQL Query on Database
- Reads results into a KNIME data table



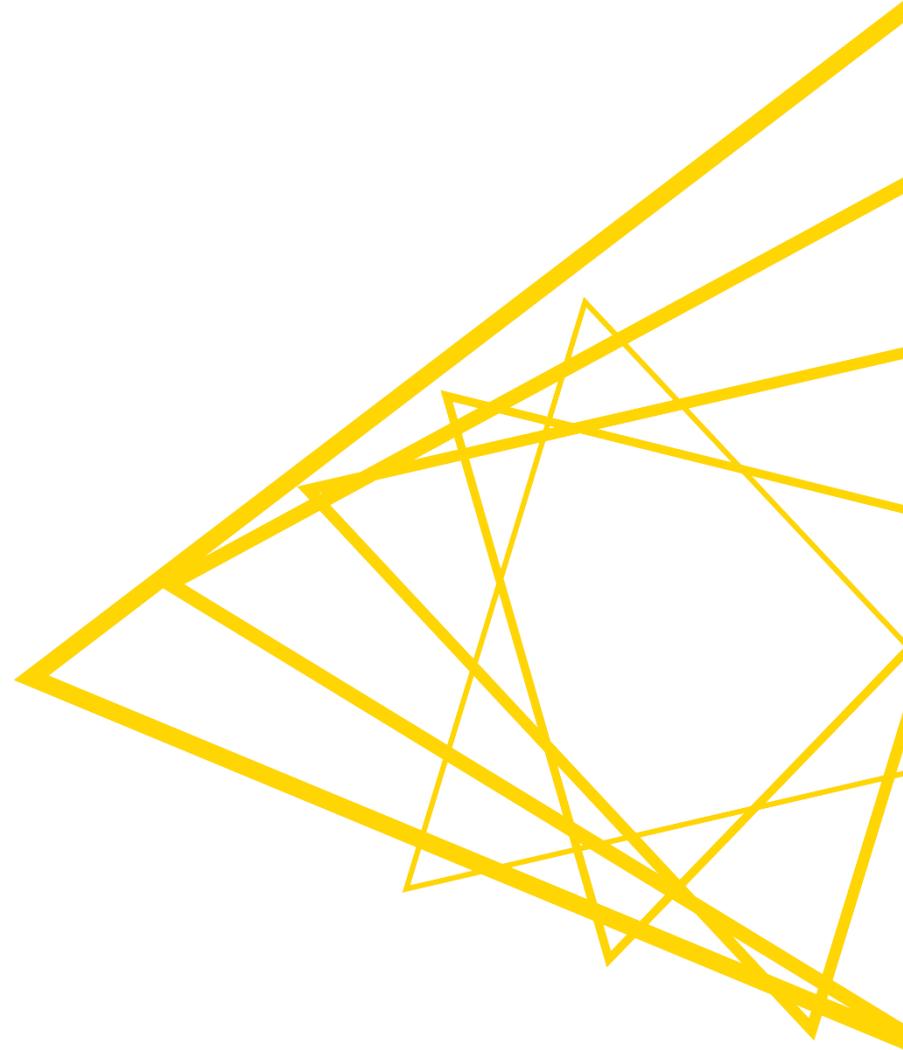
Section Exercise – 01_DB_Connect

- Connect to the database (SQLite) newCensus.sqlite in folder 1_Data
 - Use SQLite Connector (Note: SQLite Connector supports knime:// protocol)
- Explore DB metadata
- Select table ss13pme (person data in Maine)
- Import the data into a KNIME data table

Optional: Create a Credentials Input node and use it in a MySQL Connector instead of user name and password.

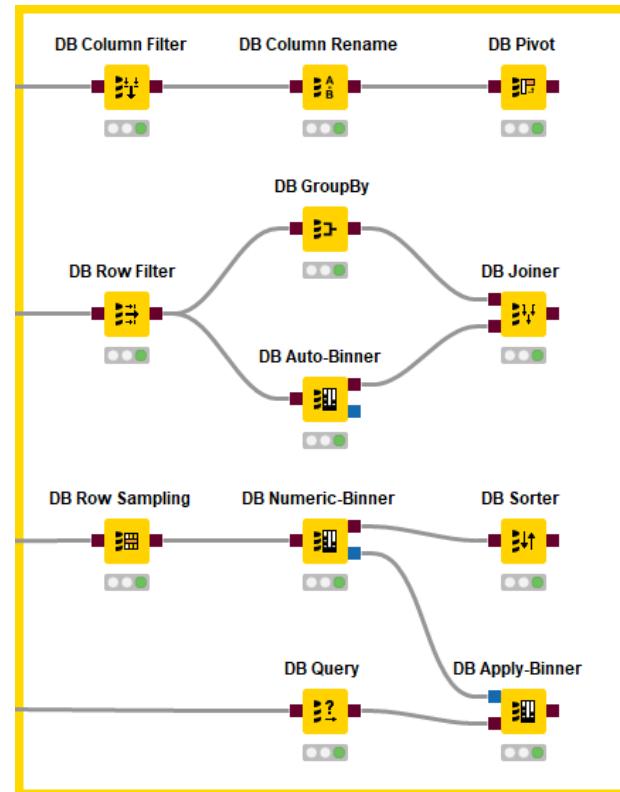
You can download the training workflows from the KNIME Hub:
<https://hub.knime.com/knime/spaces/Education/latest/Courses/>

In-Database Processing



Query Nodes

- Filter rows and columns
- Join tables/queries
- Extract samples
- Bin numeric columns
- Sort your data
- Write your own query
- Aggregate your data



Data Aggregation

RowID	Group	Value
r1	m	2
r2	f	3
r3	m	1
r4	f	5
r5	f	7
r6	m	5

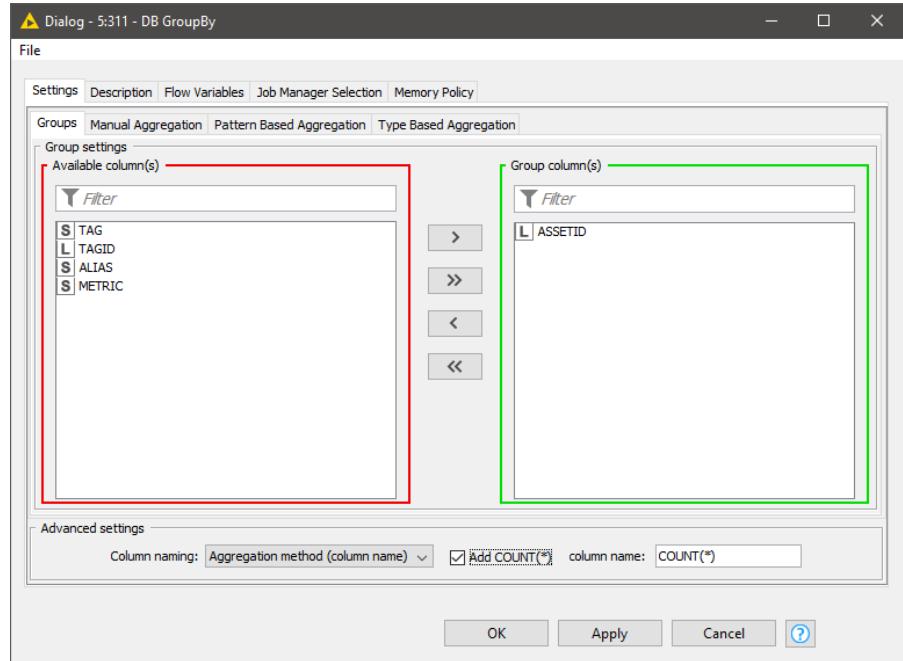


RowID	Group	Sum(Value)
r1+r3+r6	m	8
r2+r4+r5	f	15

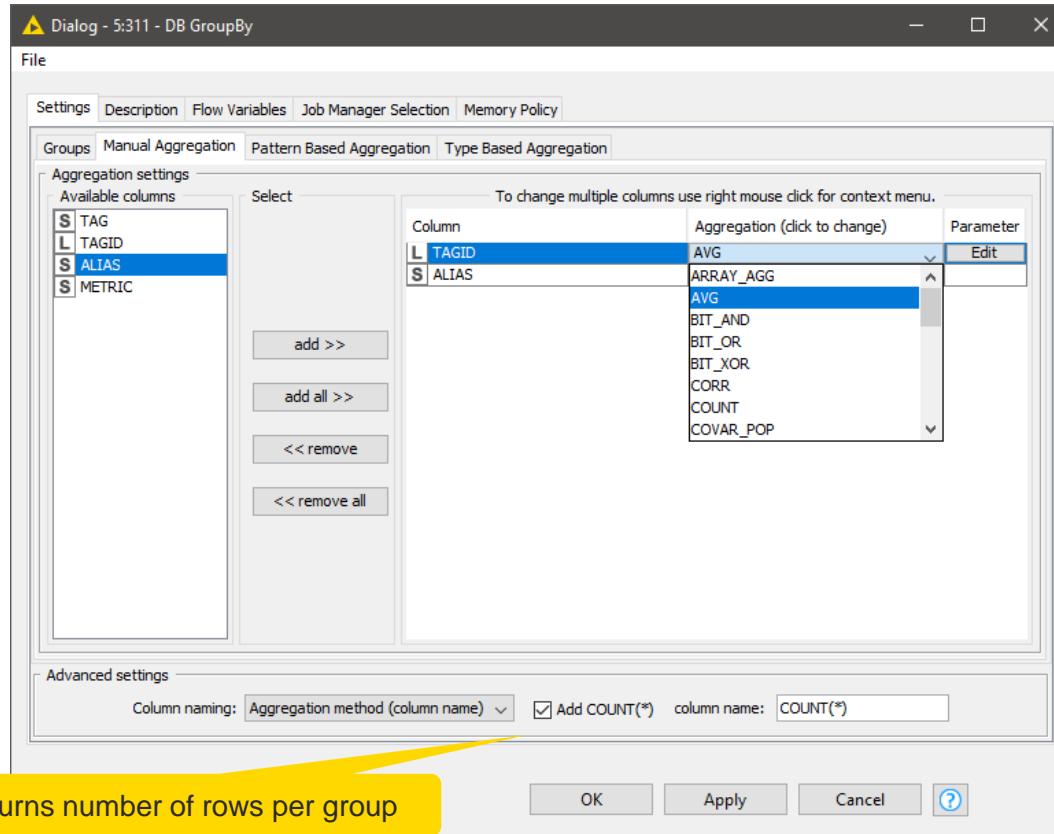
Aggregated on “Group” by method:
 $\text{sum}(\text{“Value”})$

Database GroupBy Node

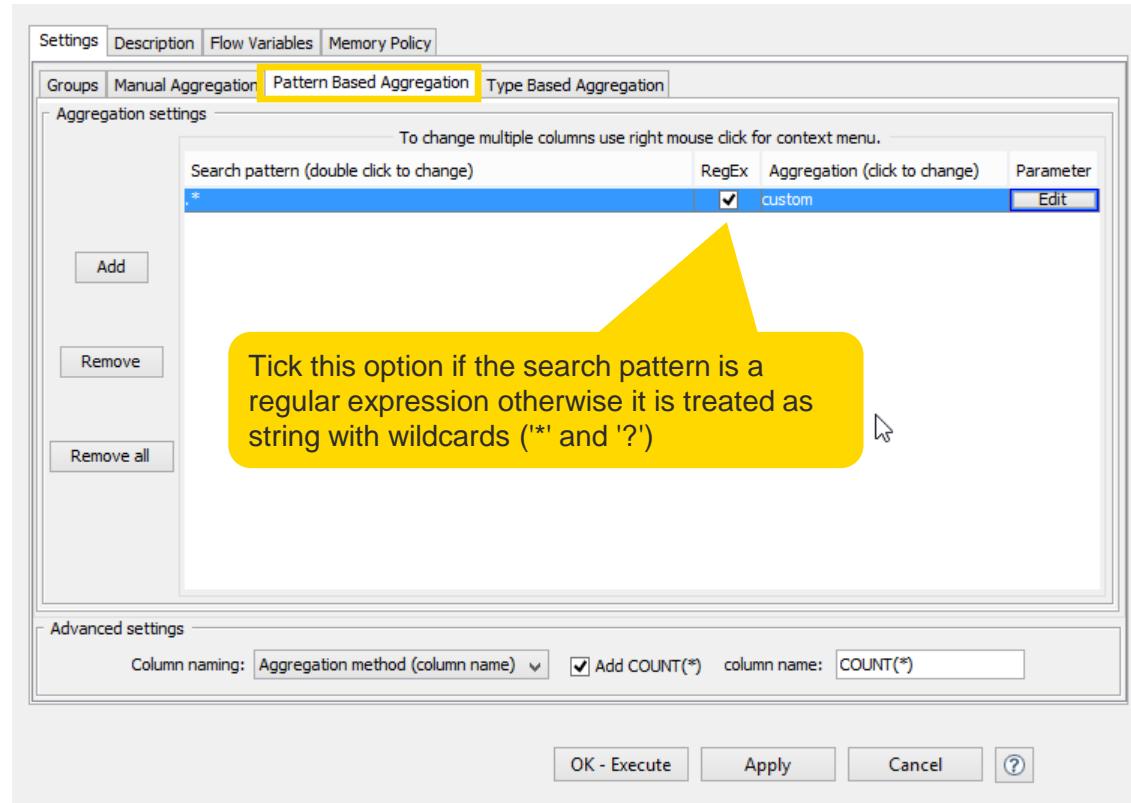
- Aggregate to summarize data



DB GroupBy Node – Manual Aggregation



Database GroupBy Node – Pattern Based Aggregation



Database GroupBy Node – Type Based Aggregation

The screenshot shows the 'Dialog - 0:3 - DB GroupBy' window with the 'Type Based Aggregation' tab selected. A yellow box highlights the 'Type Based Aggregation' tab. A yellow callout points to the 'Number (double)' item in the 'Available data types' list with the text 'Matches all numeric columns'. Another yellow callout points to the 'Number (double)' row in the 'Select' table with the text 'Matches all columns'.

Available data types

- Byte vector
- DataCell
- Double Vector (Col)
- Duration
- Image
- JSON
- Labeling
- List (Collection of:)
- Local Date
- Local Date Time
- Local Time
- Number (double)
- Number (integer)
- Number (long)
- PMML
- URI

Select

Data type	Aggregation (click to change)	Parameter
Number (double)	AVG	Edit
Number (double)	SUM	Edit

To change multiple columns use right mouse click for context menu.

Advanced settings

Column naming: Aggregation method (column name) Add COUNT(*) column name: COUNT(*)

Type matching: Strict

OK Apply Cancel ?

Database GroupBy Node – DB Specific Aggregation Methods

Dialog - 0:131 - Database GroupBy(test all)

File

Settings Description Flow Variables Memory Policy

Groups Manual Aggregation Pattern Based Aggregation Type Based Aggregation

Aggregation settings

Available columns

Column	Aggregation (click to change)	Parameter
D sepal length	Avg	Edit
D sepal width	COUNT	Edit
D petal length	GROUP_CONCAT	Edit
D petal width	MAX	Edit
S id	MIN	Edit
D sepal width	SUM	Edit
D petal length	TOTAL	Edit
S id	custom	Edit
D petal length	GROUP_CONCAT	Edit

Select

To change multiple columns use right mouse click for context menu.

add >> add all >> << remove << remove all

Advanced settings

Column naming: Aggregation method (column name) Add COUNT(*) column name: COUNT(*)

OK - Execute Apply Cancel ?

This screenshot shows the 'Database GroupBy' node dialog for SQLite. The 'Aggregation settings' tab is selected. The 'Available columns' list includes 'sepal length', 'sepal width', 'petal length', 'petal width', and 'id'. The 'Select' section lists seven aggregation functions: 'Avg', 'COUNT', 'GROUP_CONCAT', 'MAX', 'MIN', 'SUM', and 'TOTAL'. The 'Advanced settings' section has a checkbox for 'Add COUNT(*)' and a text input for 'column name: COUNT(*)'.

SQLite: 7 aggregation functions

Dialog - 0:135 - Database GroupBy(test all)

File

Settings Description Flow Variables Memory Policy

Groups Manual Aggregation Pattern Based Aggregation Type Based Aggregation

Aggregation settings

Available columns

Column	Aggregation (click to change)	Parameter
D sepal length	ARRAY_AGG	Edit
D sepal width	ARRAY_AGG	Edit
D petal length	Avg	Edit
D petal width	Corr	Edit
I rowids	Count	Edit
S id	Max	Edit
D sepal length	Min	Edit
D sepal width	Regr_Avgx	Edit
D petal length	Regr_Avgy	Edit
D petal width	Regr_Count	Edit
D sepal length	Regr_Intercept	Edit
D sepal width	Regr_R2	Edit
D petal length	Regr_Slope	Edit
D petal width	Regr_Sxx	Edit
D sepal length	Regr_Sxy	Edit

Select

To change multiple columns use right mouse click for context menu.

add >> add all >> << remove << remove all

Advanced settings

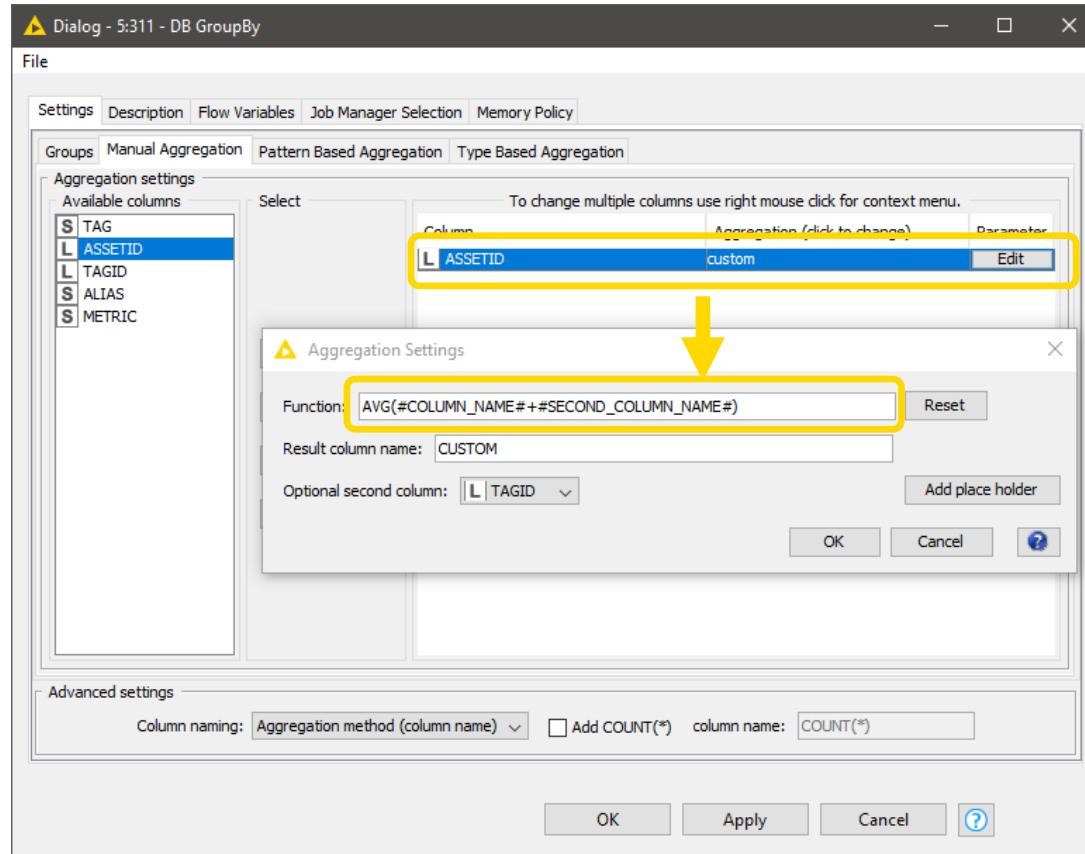
Column naming: Aggregation method (column name) Add COUNT(*) column name: COUNT(*)

OK - Execute Apply Cancel ?

This screenshot shows the 'Database GroupBy' node dialog for PostgreSQL. The 'Aggregation settings' tab is selected. The 'Available columns' list includes 'sepal length', 'sepal width', 'petal length', 'petal width', and 'id'. The 'Select' section lists 25 aggregation functions, including 'ARRAY_AGG', 'AVG', 'CORR', 'COUNT', 'MAX', 'MIN', 'Regr_Avgx', 'Regr_Avgy', 'Regr_Count', 'Regr_Intercept', 'Regr_R2', 'Regr_Slope', 'Regr_Sxx', and 'Regr_Sxy'. The 'Advanced settings' section has a checkbox for 'Add COUNT(*)' and a text input for 'column name: COUNT(*)'.

PostgreSQL: 25 aggregation functions

DB GroupBy Node – Custom Aggregation Function



Joining Columns of Data

Left Table

Manually created table - 3:49 - Table Creator

File Hilite Navigation View

Table "default" – Rows: 4 Spec – Columns: 4

Row ID	ID	age	\$ income	\$ class
Row0	1	23	<=50K	F1
Row1	3	25	<=50K	F3
Row2	6	22	>50K	A4
Row3	8	21	<=50K	C3

Join by ID

Inner Join

Manually created table - 3:50 - Table Creator

File Hilite Navigation View

Table "default" – Rows: 6 Spec – Columns: 4

Row ID	ID	age	\$ income	\$ sex
Row0	1	23	<=50K	M
Row1	2	25	<=50K	F
Row2	4	23	>50K	M
Row3	5	21	<=50K	F
Row4	6	25	>50K	M
Row5	7	24	<=50K	M

Left Outer Join

Joined table - 3:51 - Joiner

File Hilite Navigation View

Table "default" – Rows: 4 Spec – Columns: 7 Properties Flow Variables

Row ID	ID	age	\$ income	\$ class	age (#1)	\$ incom...
Row0_Row0	1	23	<=50K	F1	23	<=50K M
Row2_Row4	6	22	>50K	A4	25	>50K M
Row1_?	3	25	<=50K	F3	?	?
Row3_?	8	21	<=50K	C3	?	?

Missing values in the right table.

Missing values in the left table.

Right Outer Join

Joined table - 3:51 - Joiner

File Hilite Navigation View

Table "default" – Rows: 6 Spec – Columns: 7 Properties Flow Variables

Row ID	ID	age	\$ income	\$ class	age (#1)	\$ incom...
Row0	1	23	<=50K	F1	23	<=50K M
Row4	6	22	>50K	A4	25	>50K M
?_Row1	?	?	?	?	?	?
?_Row2	?	?	?	?	?	?
?_Row3	?	?	?	?	?	?
?_Row5	?	?	?	?	?	?

Joining Columns of Data

Left Table

Manually created table - 3:49 - Table Creator

File Hilite Navigation View

Table "default" – Rows: 4 Spec – Columns: 4

Row ID	ID	age	\$ income	\$ class
Row0	1	23	<=50K	F1
Row1	3	25	<=50K	F3
Row2	6	22	>50K	A4
Row3	8	21	<=50K	C3

Join by ID

Manually created table - 3:50 - Table Creator

File Hilite Navigation View

Table "default" – Rows: 6 Spec – Columns: 4

Row ID	ID	age	\$ income	\$ sex
Row0	1	23	<=50K	M
Row1	2	25	<=50K	F
Row2	4	23	>50K	M
Row3	5	21	<=50K	F
Row4	6	25	>50K	M
Row5	7	24	<=50K	M

Full Outer Join

Joined table - 3:51 - Joiner

File Hilite Navigation View

Table "default" – Rows: 8 Spec – Columns: 7 Properties Flow Variables

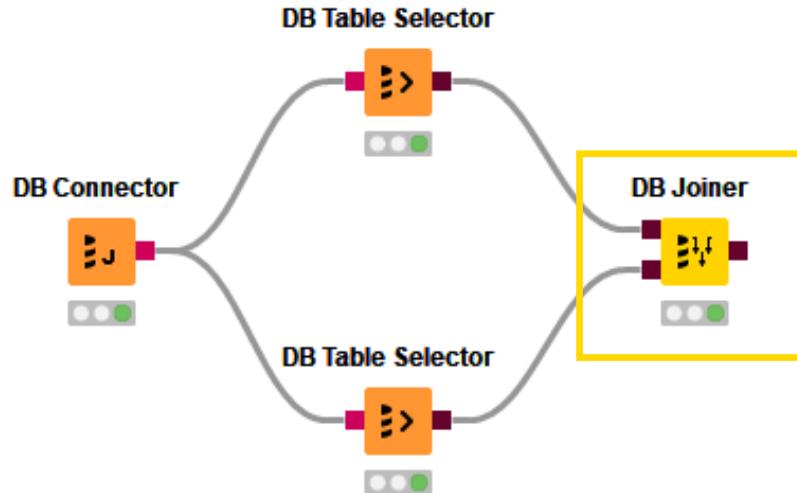
Row ID	ID	age	\$ income	\$ class	age (#1)	\$ incom...
Row0_Row0	1	23	<=50K	F1	23	<=50K M
Row2_Row4	6	22	>50K	A4	25	>50K M
Row1_?	3	25	<=50K	F3	?	?
Row3_?	8	21	<=50K	C3	?	?
?_Row1	?	?	?	?	25	<=50K F
?_Row2	?	?	?	?	23	>50K M
?_Row3	?	?	?	?	21	<=50K F
?_Row5	?	?	?	?	24	<=50K M

Missing values
in the left table

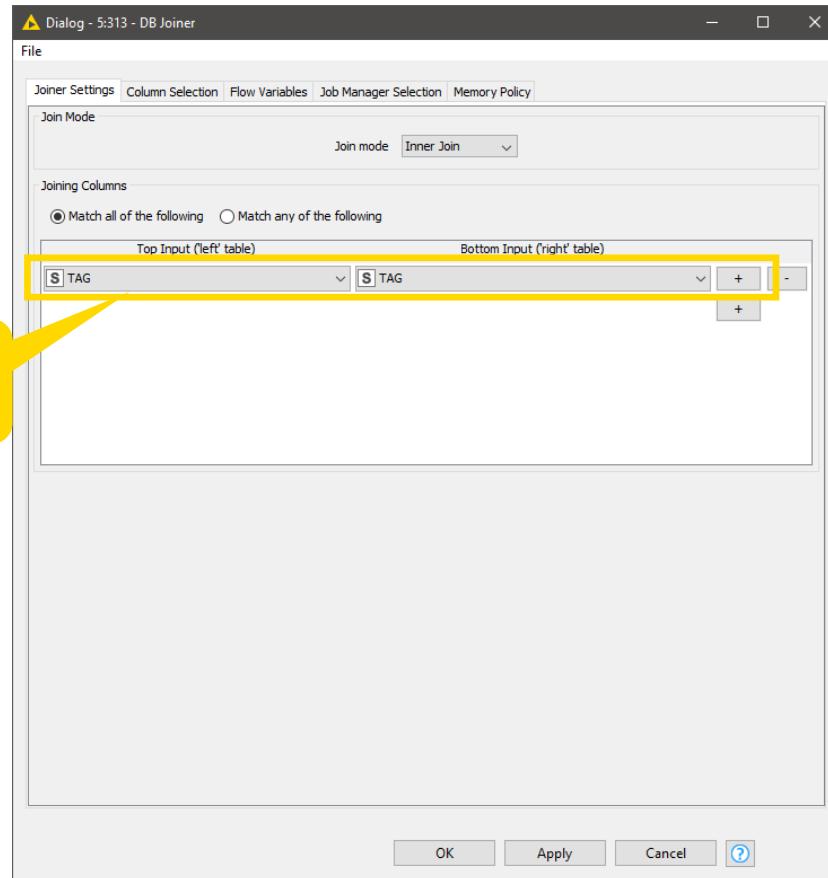
Missing values in
the right table

Database Joiner Node

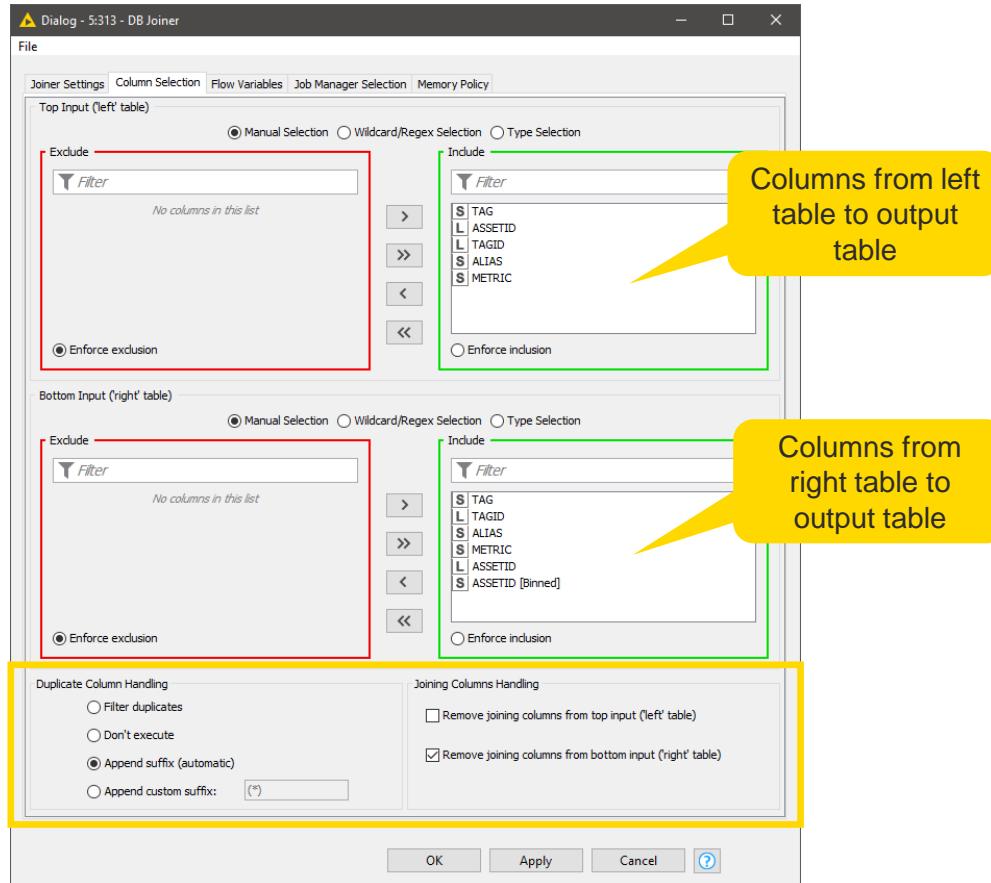
- Combines columns from 2 different tables
- Top port contains “Left” data table
- Bottom port contains the “Right” data table



Joiner Configuration – Linking Rows

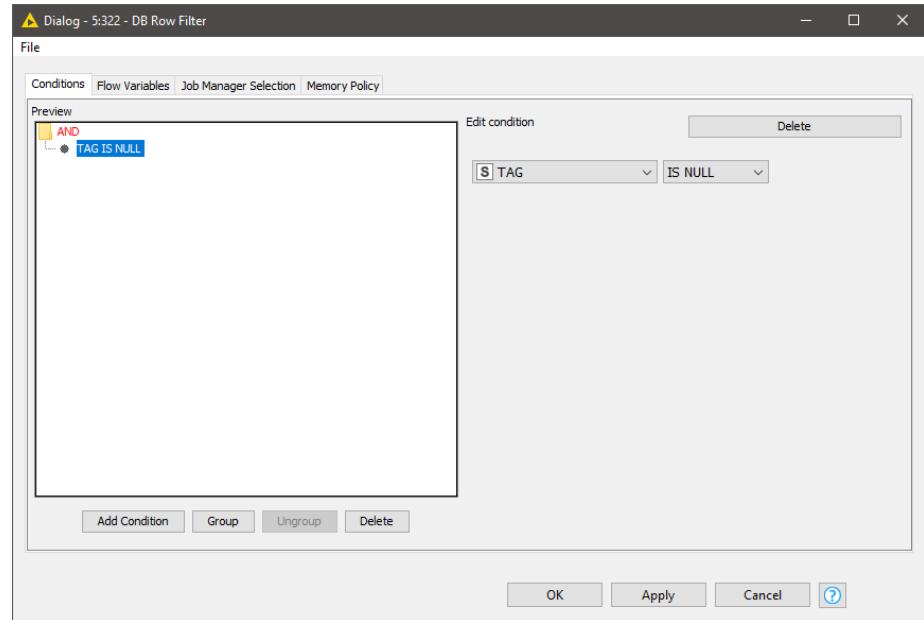
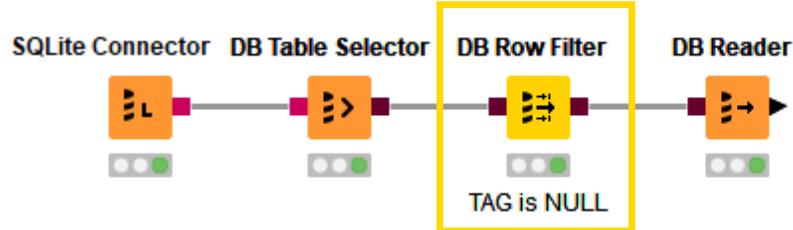


Joiner Configuration – Column Selection



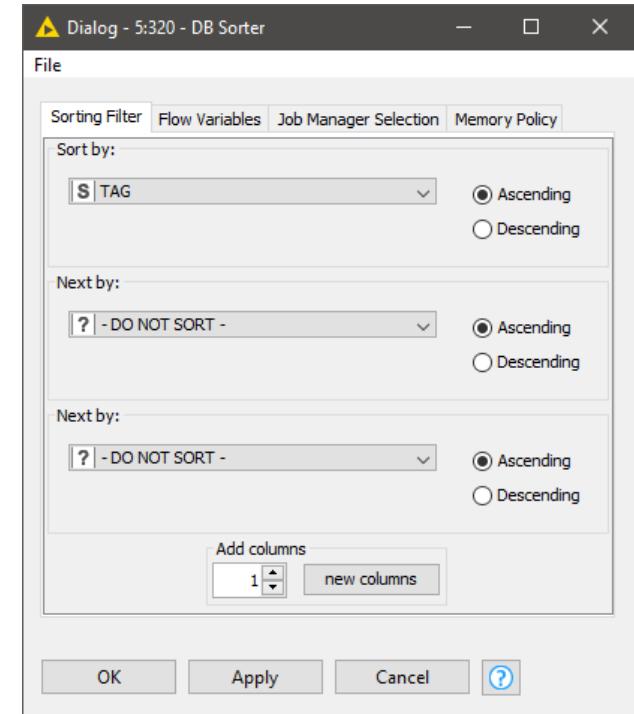
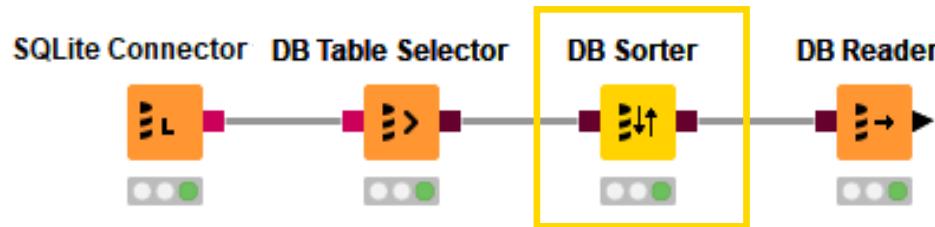
Database Row Filter Node

- Filters rows that do not match the filter criteria
- Use the *IS NULL* or *IS NOT NULL* operator to filter missing values



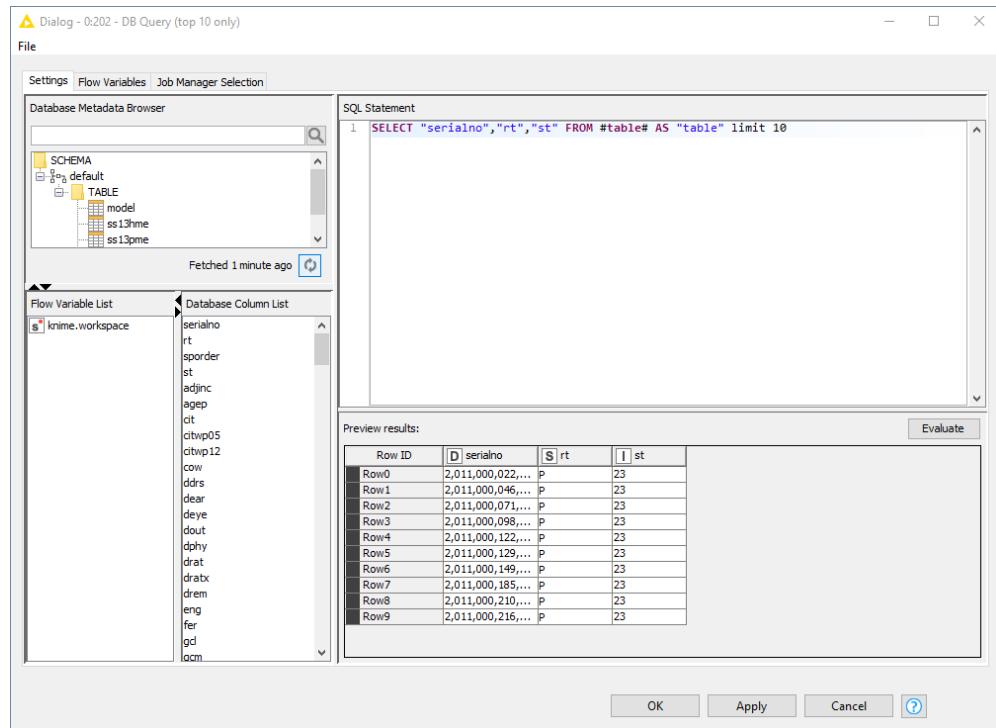
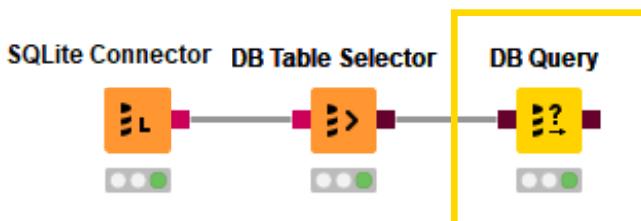
Database Sorter Node

- Sorts the input data by one or multiple columns



Database Query Node

- Executes arbitrary SQL queries
- #table# is replaced with input query



Section Exercise – 02_DB_InDB_Processing

From tables ss13hme (house data) and ss13pme (person data) in database newCensus.sqlite create 4 tables

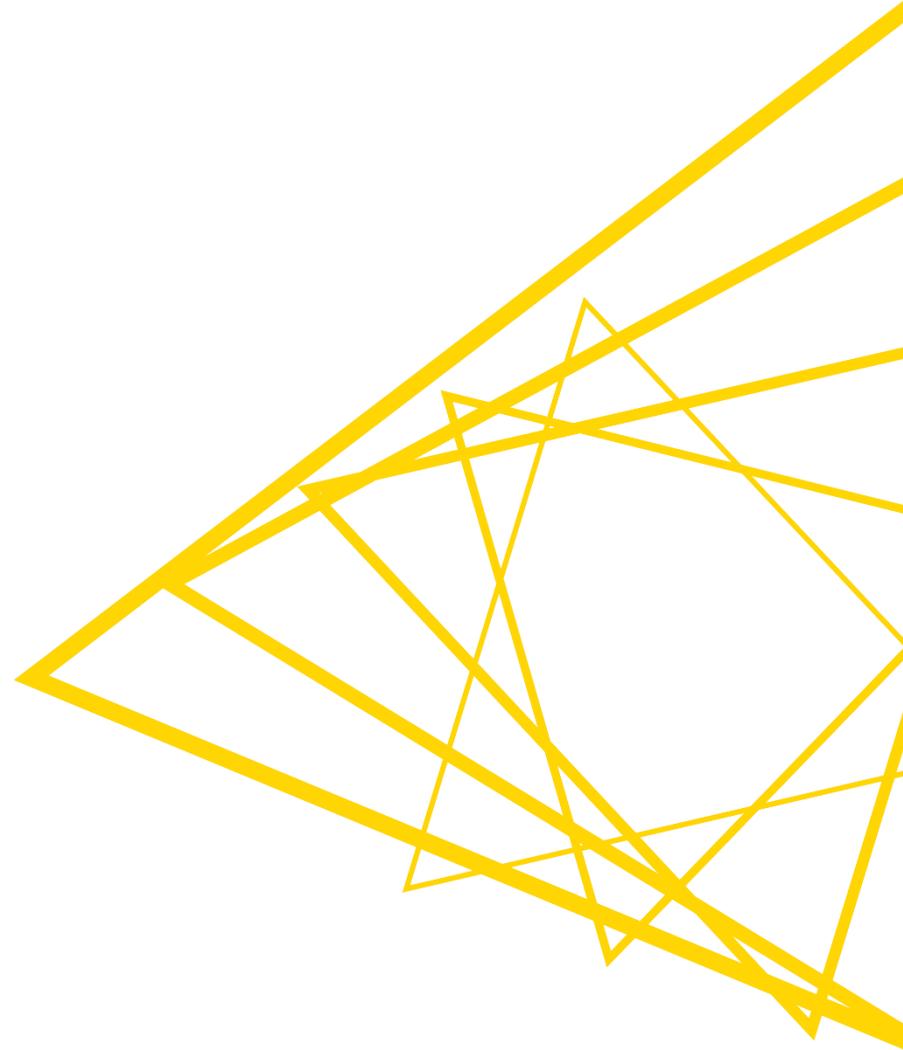
1. Join ss13hme and ss13pme on SERIALNO. Remove all columns named PUMA* and PWGTP* from both tables.
2. Filter all rows from ss13pme where COW is NULL.
3. Filter all rows from ss13pme where COW is NOT NULL.
4. Calculate average AGEP for the different SEX groups.

For all 4 tasks, at the end load data into KNIME.

Optional. Sort the data rows by descending AGEP and extract top 10 only.

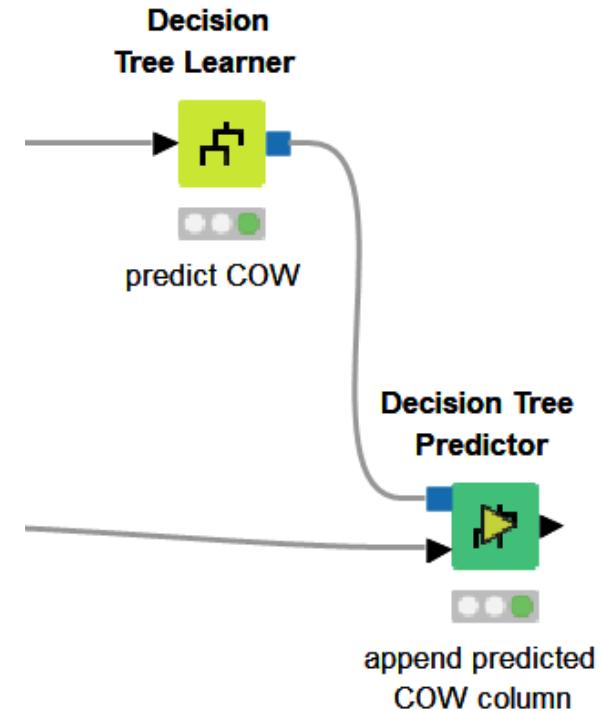
Hint: Use LIMIT to restrict the number of rows returned by the db.

Predicting COW Values with KNIME



Missing Value Implementation Approach

- Remember that after we perform some in-database ETL on the data, a key task is to fill in missing values for the COW field in our dataset
- We could try to do this by applying some simple business rules, but a more sophisticated approach is to build a model to predict COW
- Therefore, we will train and apply a decision tree model for COW



Section Exercise – 03_DB_Modelling

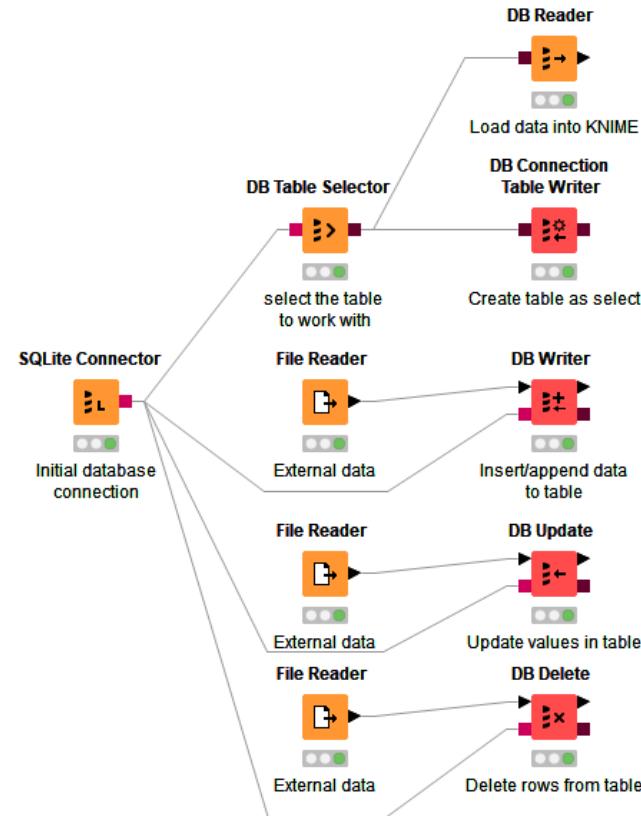
- Train a Decision Tree to predict the COW where COW is not null
- Apply Decision Tree Model to predict COW where COW is missing (null)

Write/Load Data into a Database



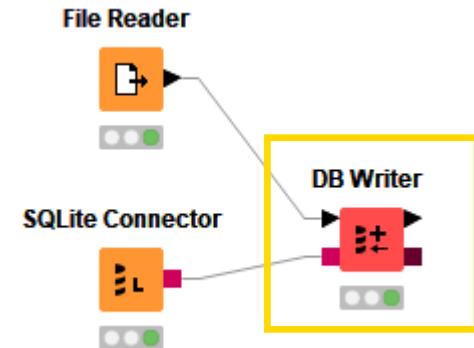
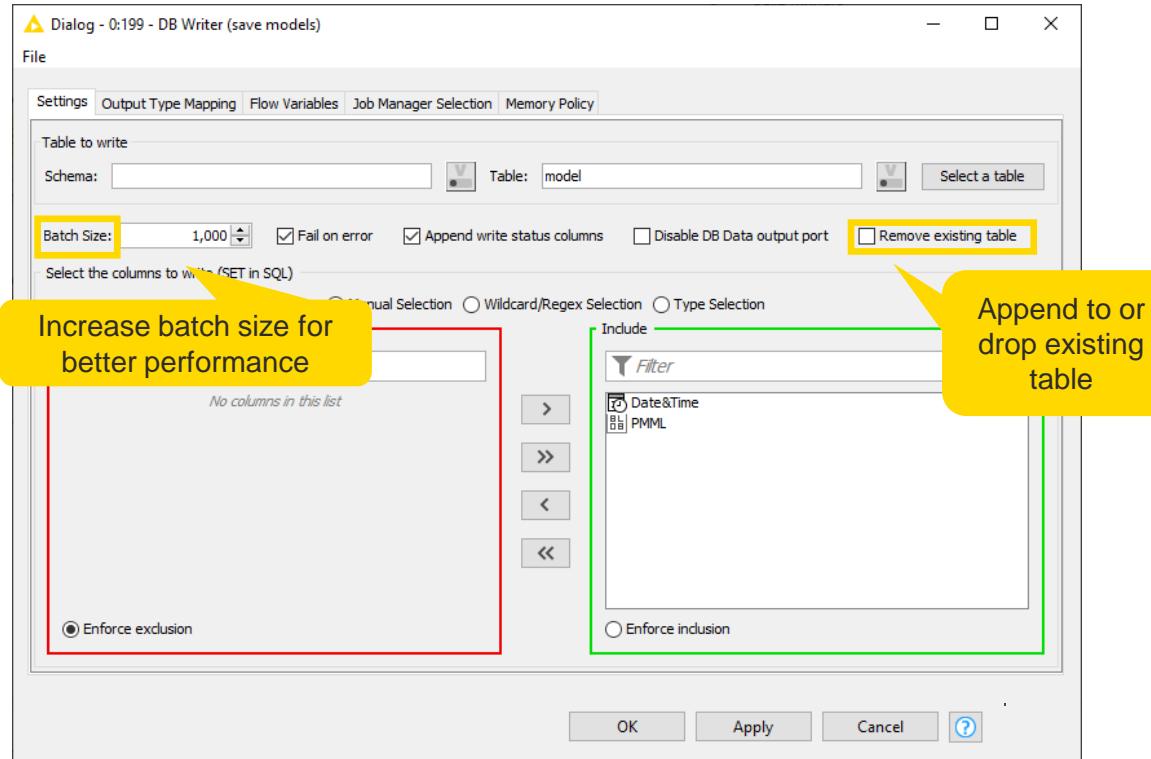
Database Writing Nodes

- Create table as select
- Insert/append data
- Update values in table
- Delete rows from table



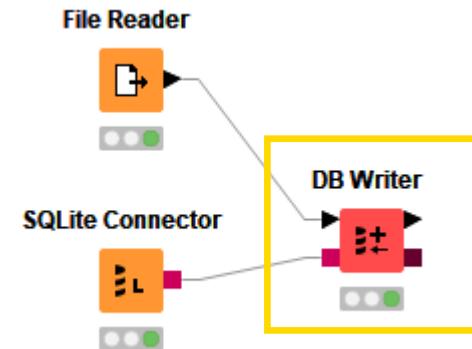
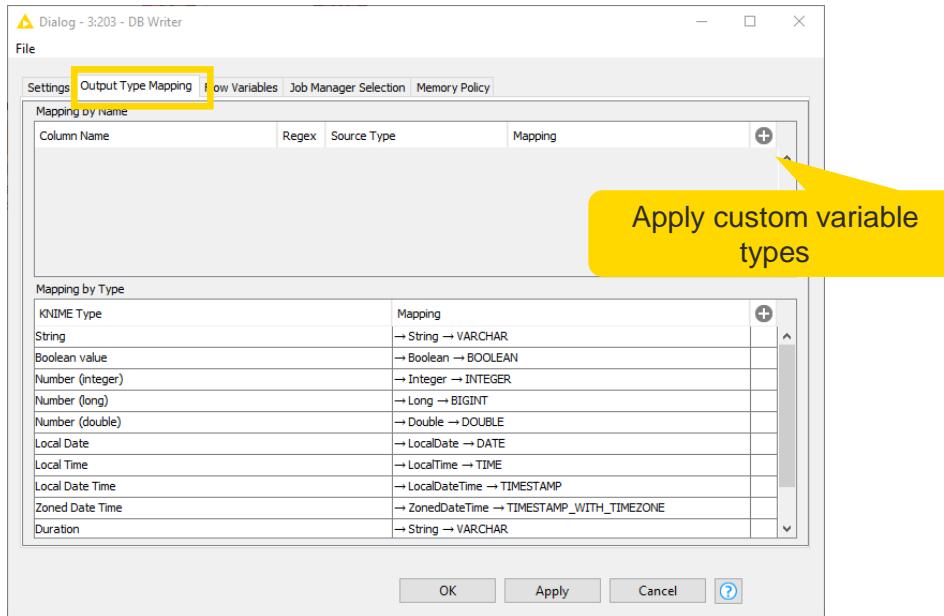
DB Writer Node

- Writes data from a KNIME data table directly into a database table



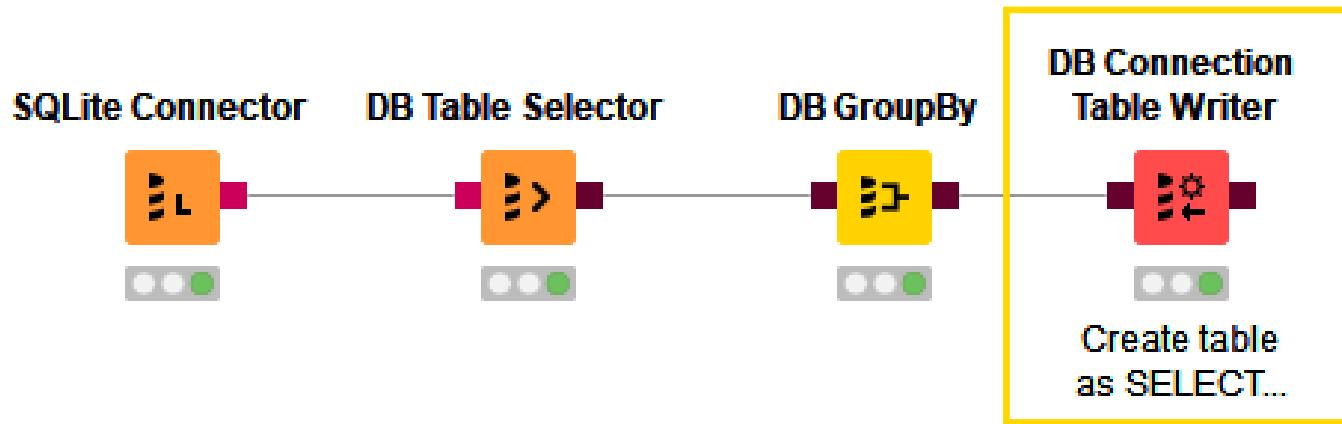
DB Writer Node (continued)

- Writes data from a KNIME data table **directly** into a database table



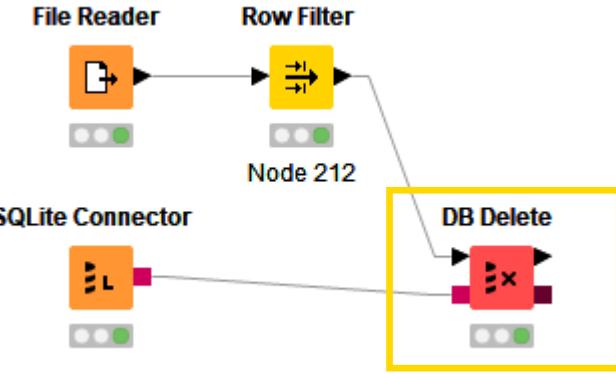
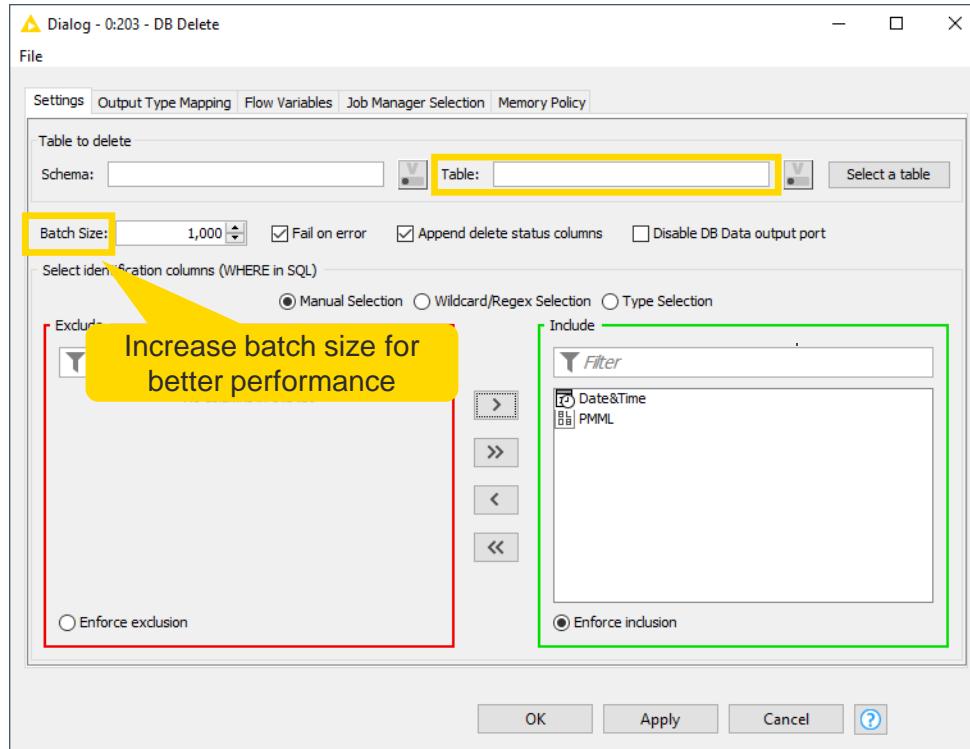
DB Connection Table Writer Node

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



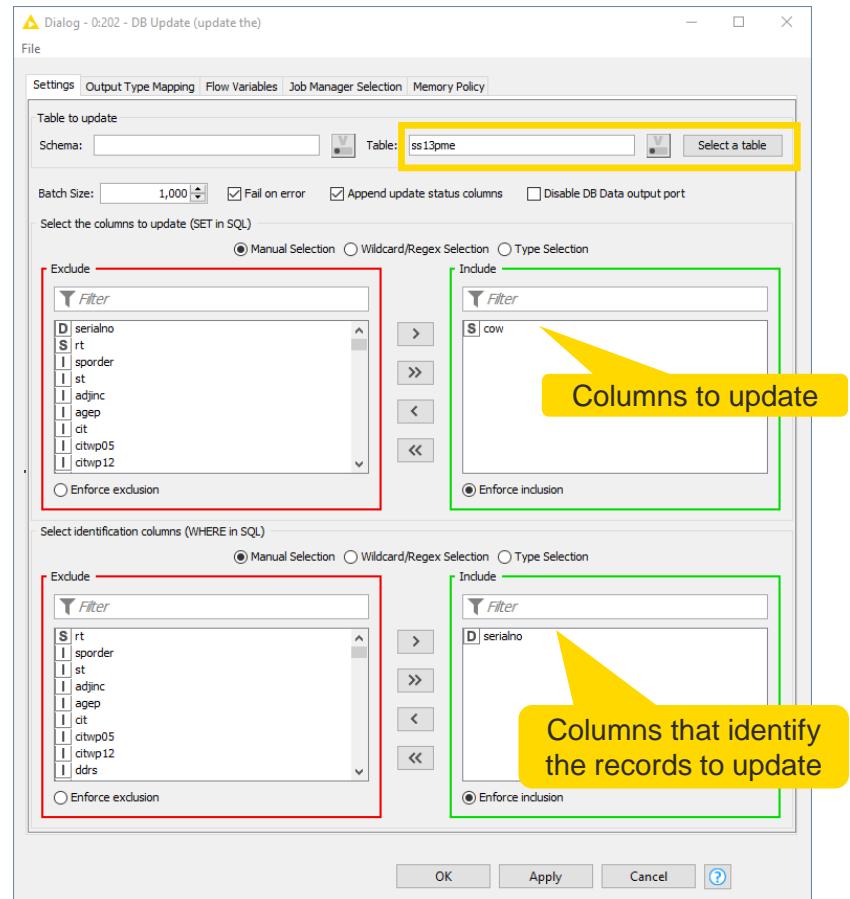
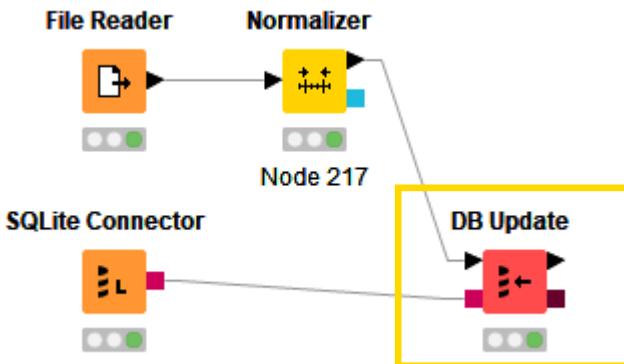
DB Delete Node

- Deletes all database records that match the values of the selected columns



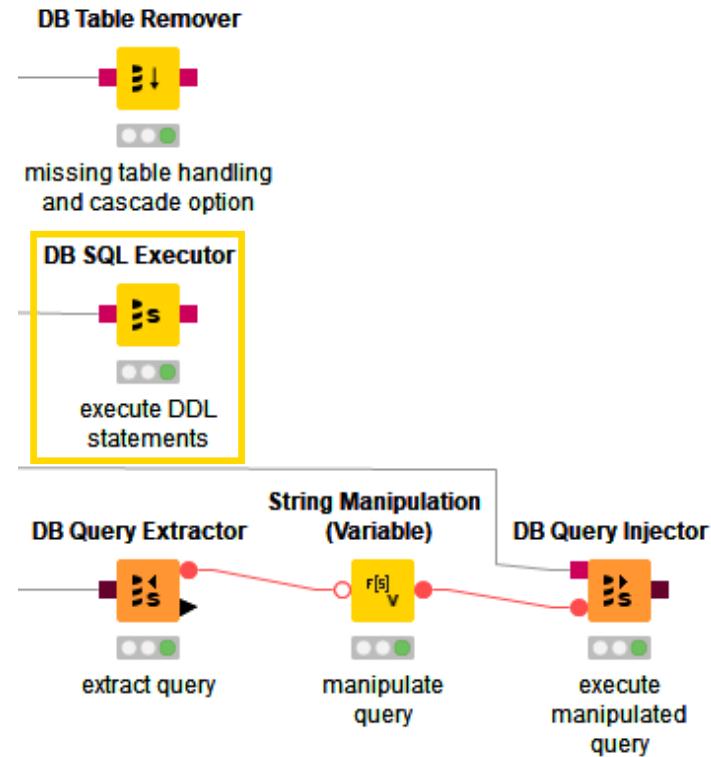
Database Update Node

- Updates all database records that match the update criteria



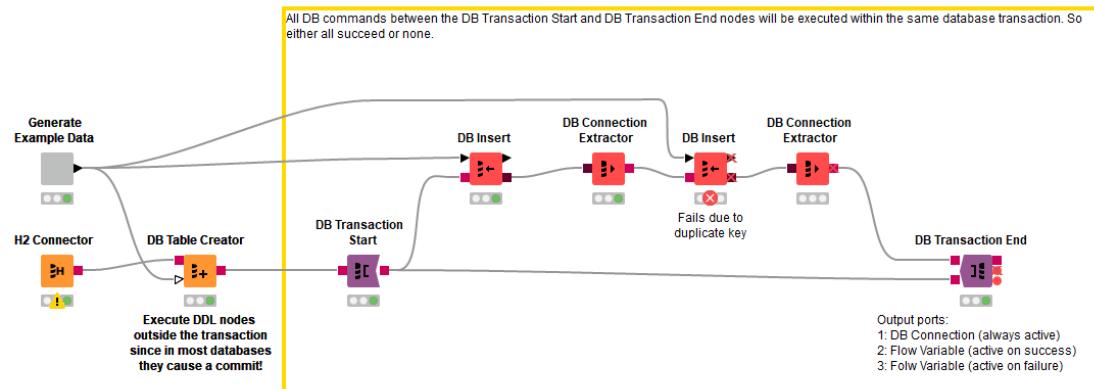
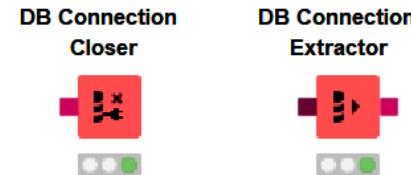
Utility

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



More Utility Nodes and Transaction Support

- DB Connection Extractor
- DB Connection Closer
- DB Transaction Start/End
 - Take advantage of these nodes to group several database data manipulation operations into a single unit of work
 - This transaction either completes entirely or not at all.
 - Uses the default [isolation level](#) of the connected database.



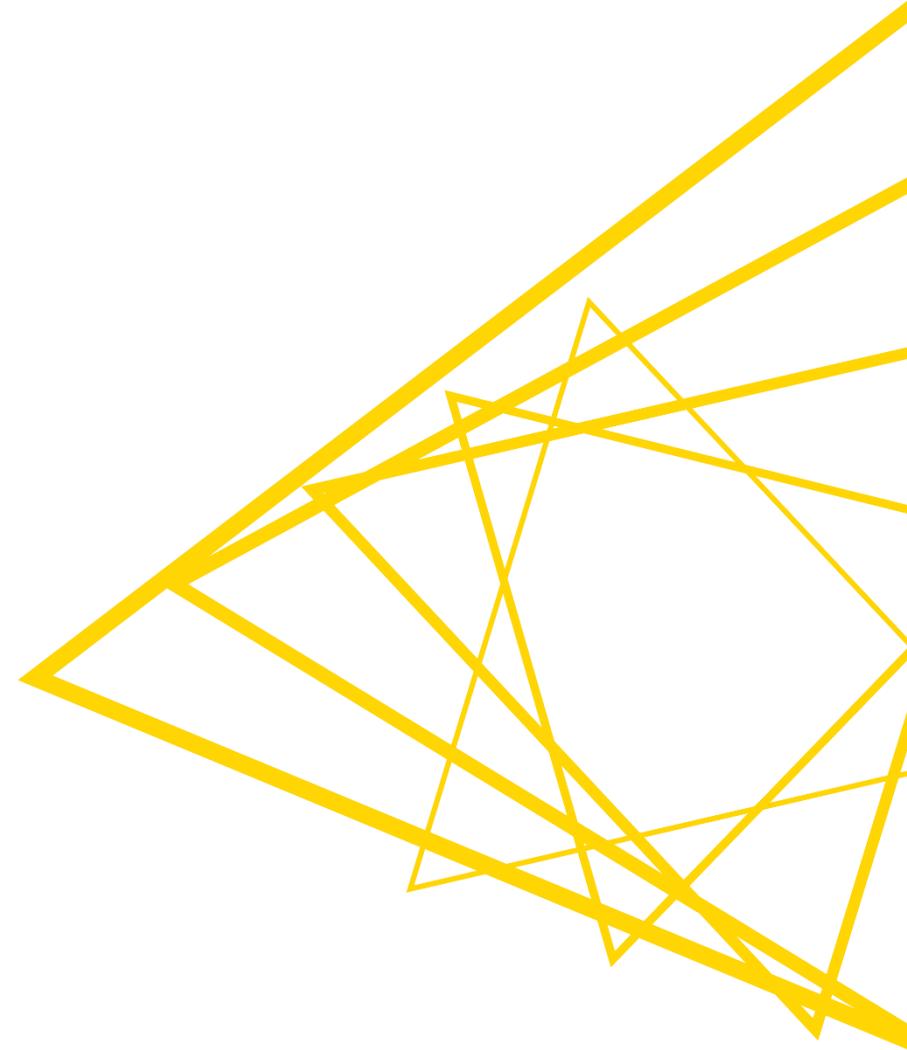
Workflow is available on the KNIME Hub:
<https://kni.me/w/kWP1OhaY4DXK444n>

Section Exercise – 04_DB_WritingToDB

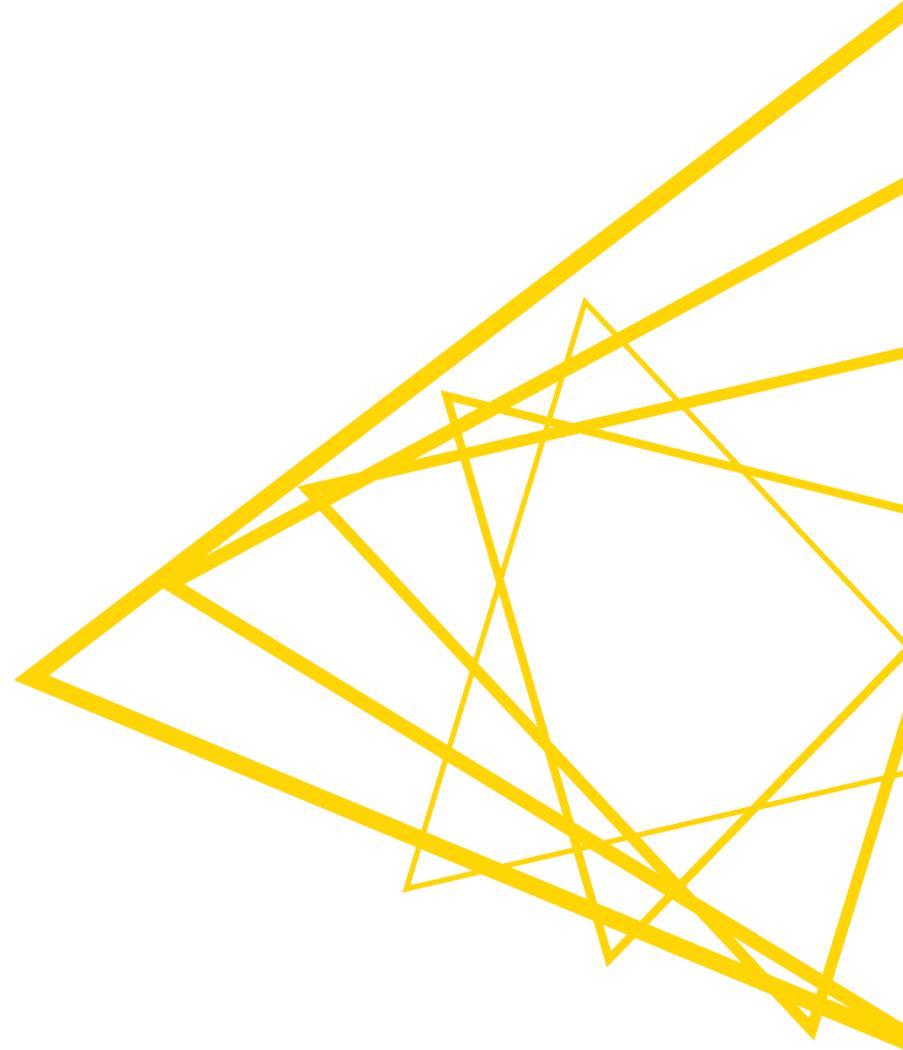
- Write the original table to ss13pme_original table with a Database Connection Table Writer node ... just in case we mess up with the updates in the next step.
- Update all rows in ss13pme table with the output of the predictor node. That is all rows with missing COW value with the predicted COW value, using column SERIALNO for WHERE condition (SERIALNO uniquely identifies each person). Check the UpdateStatus column for success.

Optional: Write the learned Decision Tree Model and the timestamp into a new table named "model"

**Let's Now Try the Same with
Hadoop**



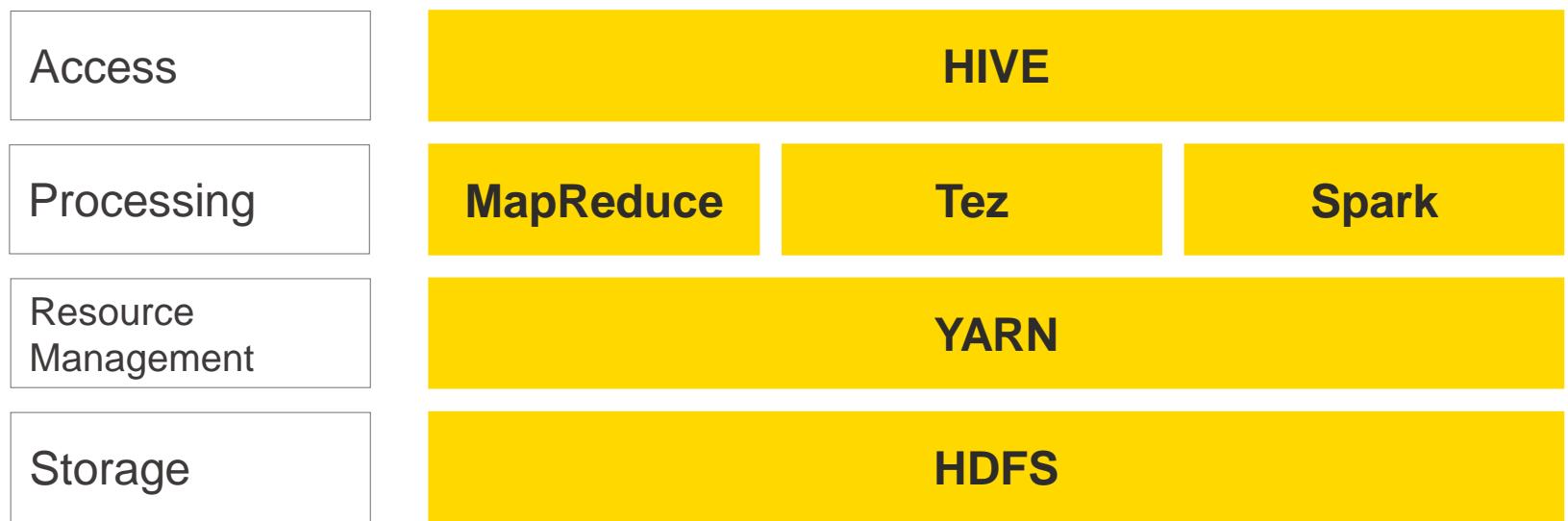
A Quick Intro to Hadoop



Apache Hadoop

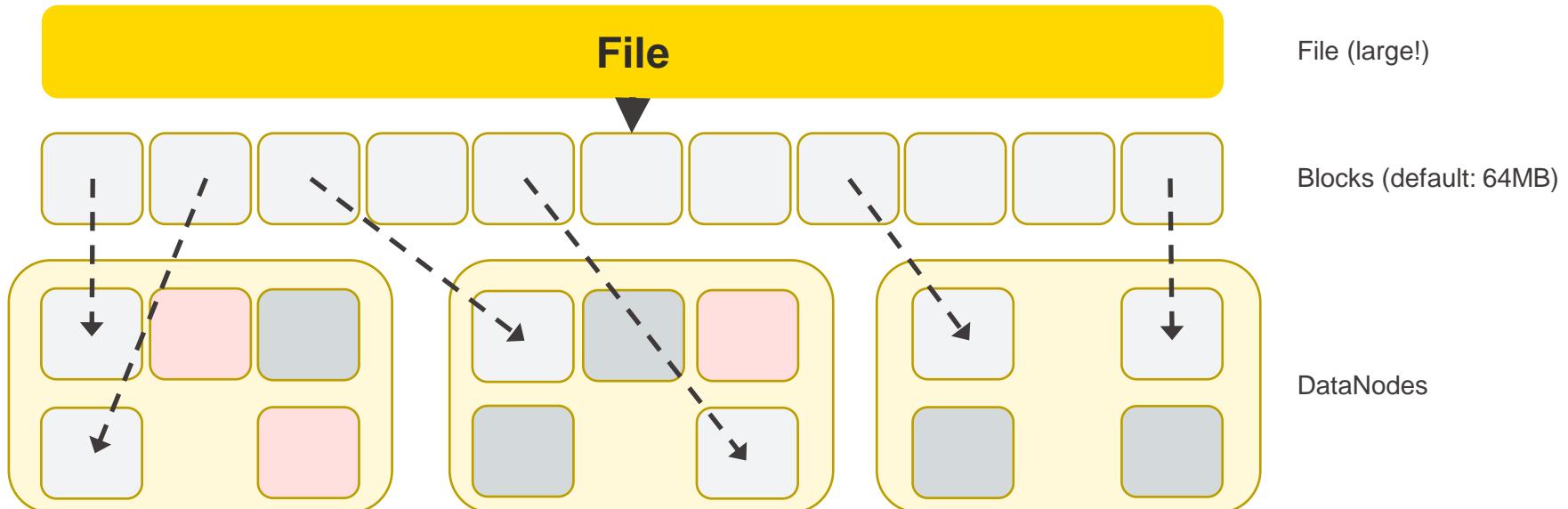
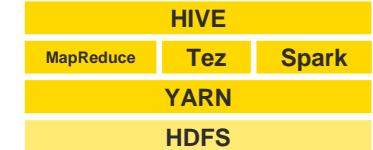
- Open-source framework for distributed storage and processing of large data sets
- Designed to scale up to thousands of machines
- Does not rely on hardware to provide high availability
 - Handles failures at application layer instead
- First release in 2006
 - Rapid adoption, promoted to top level Apache project in 2008
 - Inspired by Google File System (2003) paper
- Spawnsed diverse ecosystem of products

Hadoop Ecosystem



HDFS

- Hadoop distributed file system
- Stores large files across multiple machines



HDFS – NameNode and DataNode

- **NameNode**

- Master service that manages file system namespace
 - Maintains metadata for all files and directories in filesystem tree
 - Knows on which datanode blocks of a given file are located
- Whole system depends on availability of NameNode

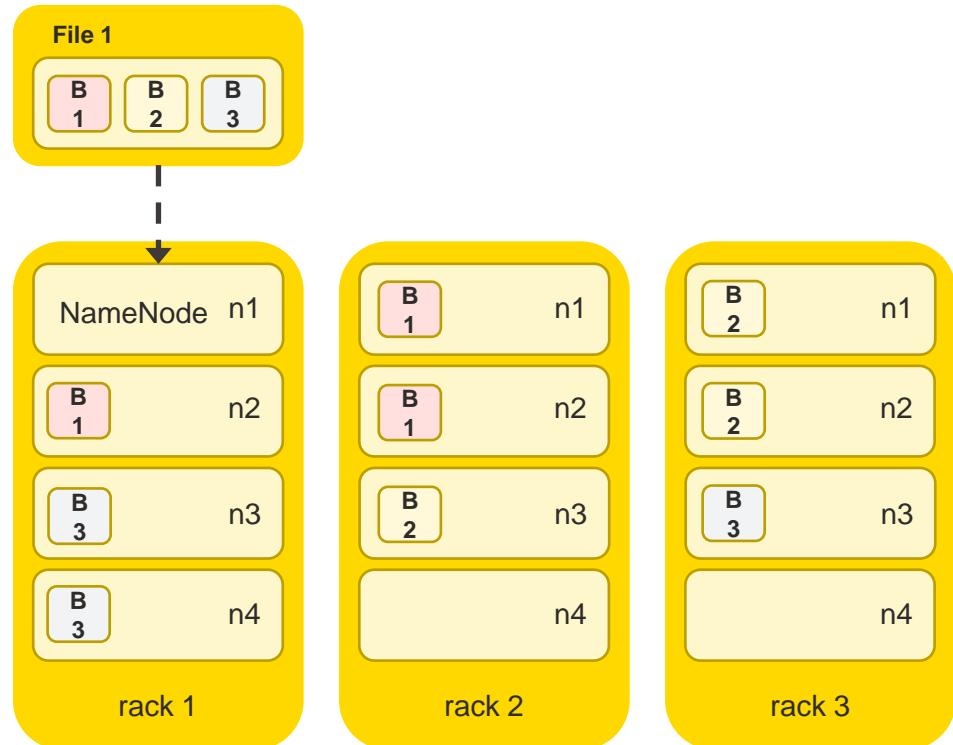
- **DataNodes**

- Workers, store and retrieve blocks per request of client or namenode
- Periodically report to namenode that they are running and which blocks they are storing

HDFS – Data Replication and File Size

Data Replication

- All blocks of a file are stored as sequence of blocks
- Blocks of a file are replicated for fault tolerance (usually 3 replicas)
 - Aims: improve data reliability, availability, and network bandwidth utilization



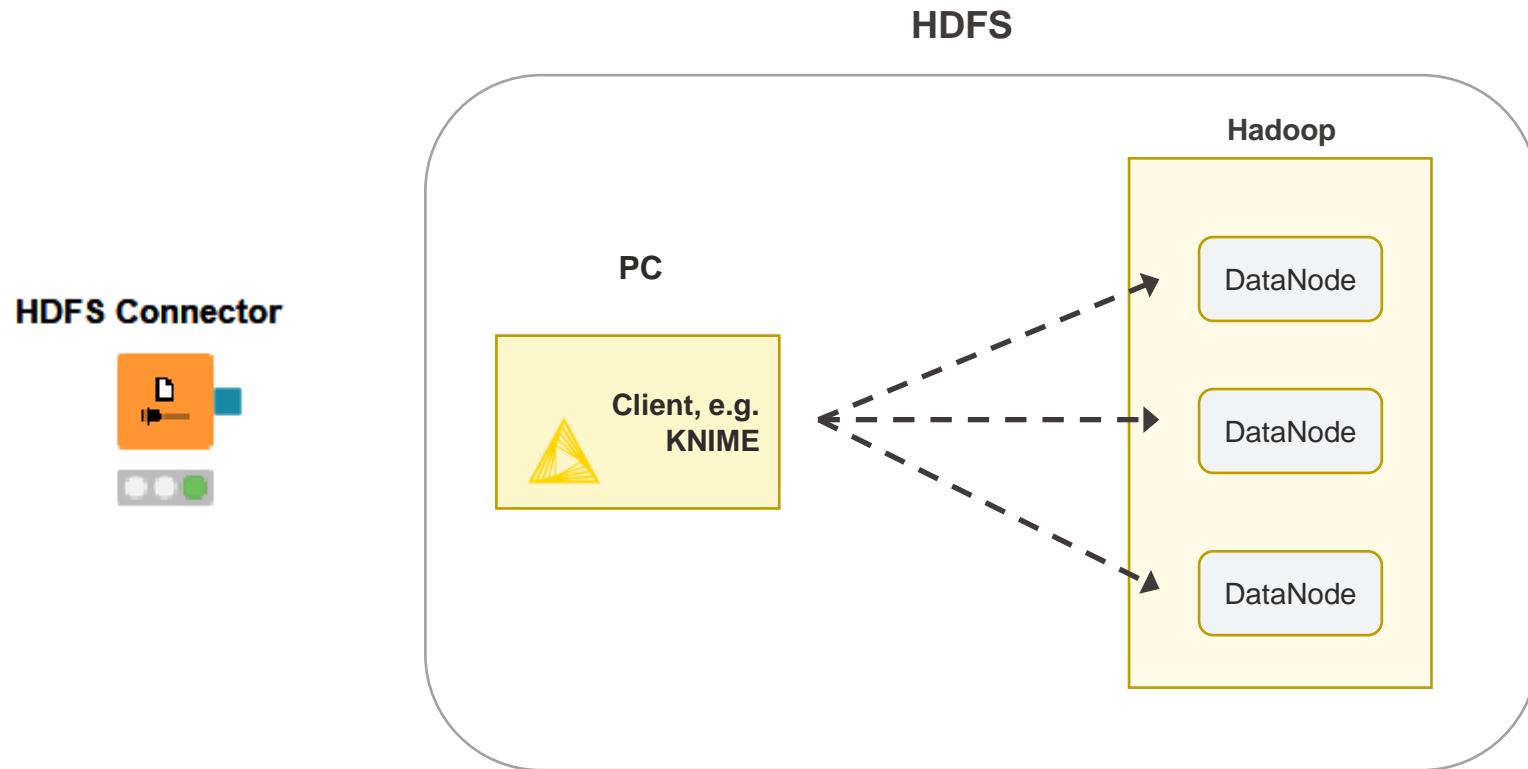
HDFS – Access and File Size

- Several ways to access HDFS data
 - HDFS
 - Direct transmission of data from nodes to client
 - Needs access to all nodes in cluster
 - WebHDFS
 - Direct transmission of data from nodes to client via HTTP
 - Needs access to all nodes in cluster
 - HttpFS
 - All data is transmitted to client via one single gateway node -> HttpFS service

File Size:

- Hadoop is designed to handle fewer large files instead of lots of small files
- Small file: File significantly smaller than Hadoop block size
- Problems:
 - Namenode memory
 - MapReduce performance

HDFS – Access

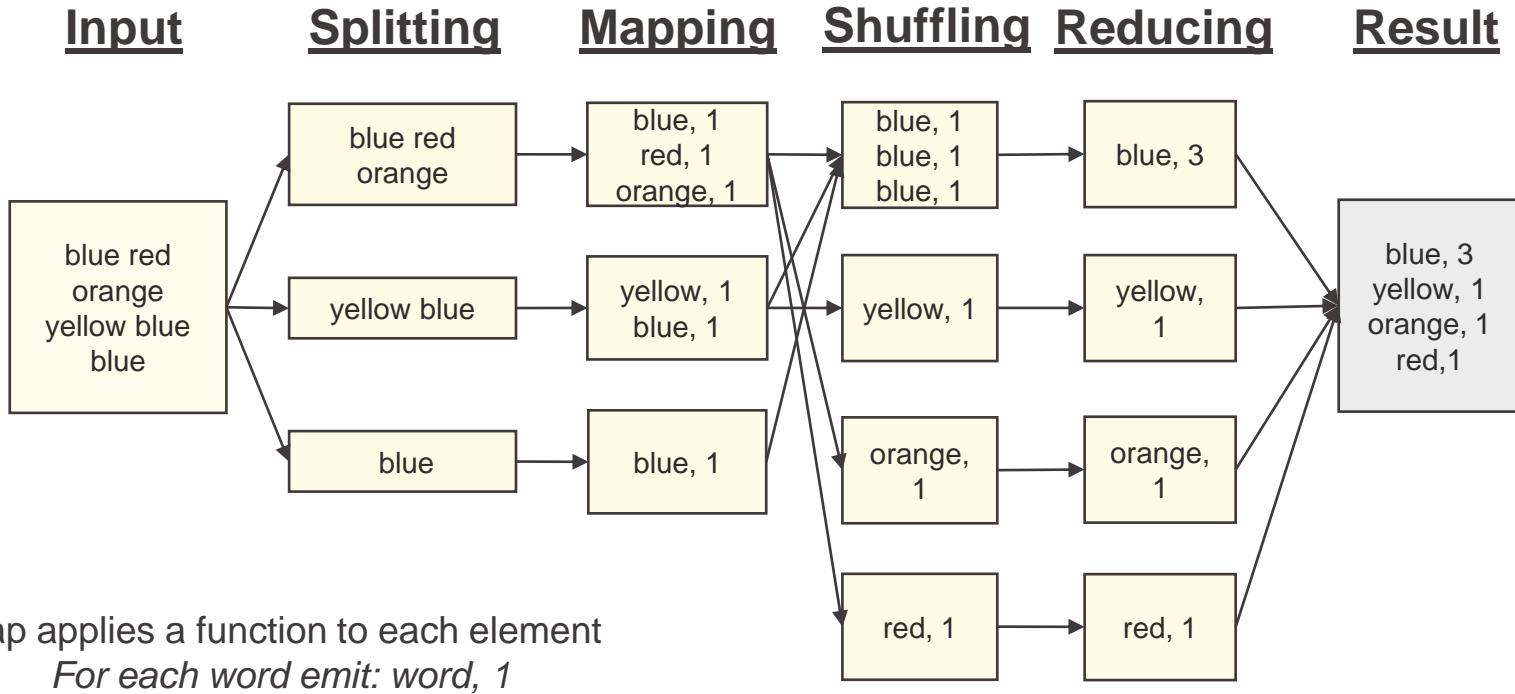


YARN

- Cluster resource management system
- Two elements
 - Resource manager (one per cluster):
 - Knows where worker nodes are located and how many resources they have
 - Scheduler: Decides how to allocate resources to applications
 - Node manager (many per cluster):
 - Launches application containers
 - Monitor resource usage and report to Resource Manager

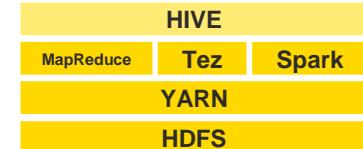


MapReduce

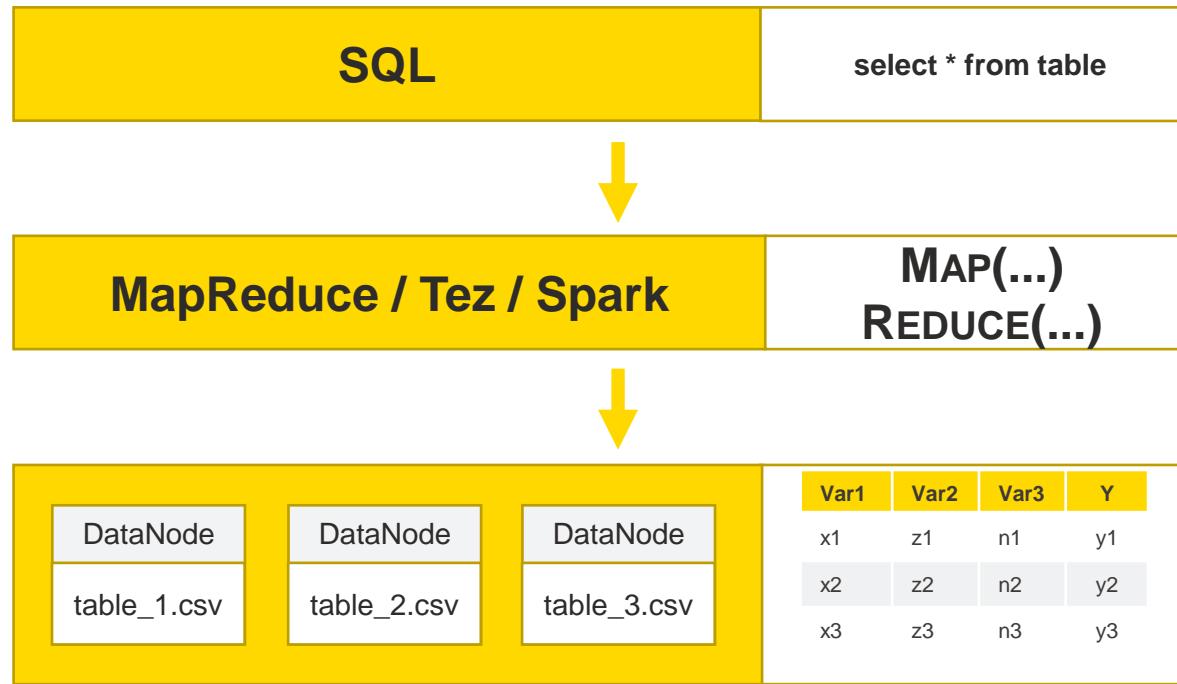


Hive

- **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



Hive



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 – Data Representation

DataFrame:

- *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
- *Lazily evaluated*:
 - Functions are not executed until an *action* is triggered, that requests to actually see the row data
- *Distributed*:
 - Each row belongs to exactly one *partition*
 - Each partition is held by a Spark Executor

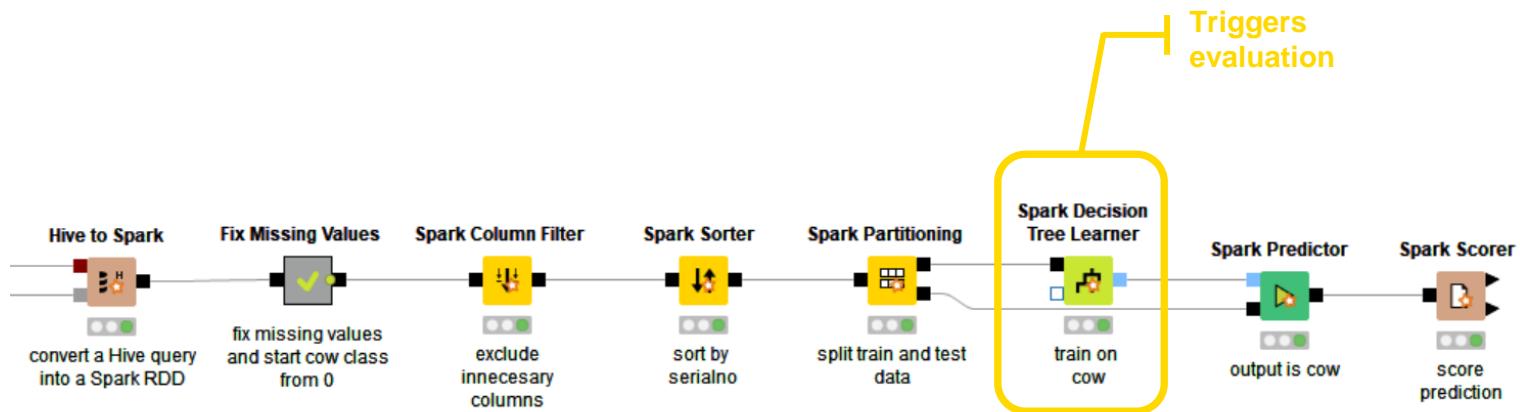
Name	Surname	Age
John	Doe	35
Jane	Roe	29
...

Note:

- Earlier versions of KNIME and Spark used *RDDs* (resilient distributed datasets)
- In KNIME, DataFrames are always used in Spark 2 and later.

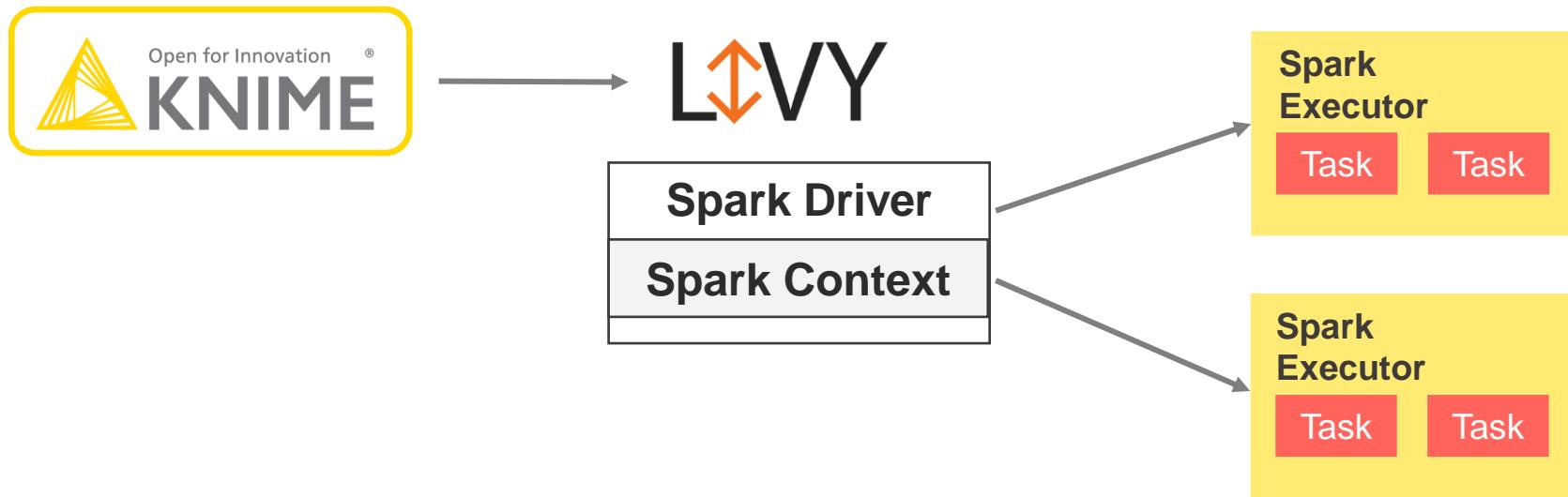
Spark – Lazy Evaluation

- Functions ("transformations") on DataFrames are not executed immediately
- Spark keeps record of the transformations for each DataFrame
- The actual execution is only triggered once the data is needed
- Offers the possibility to optimize the transformation steps

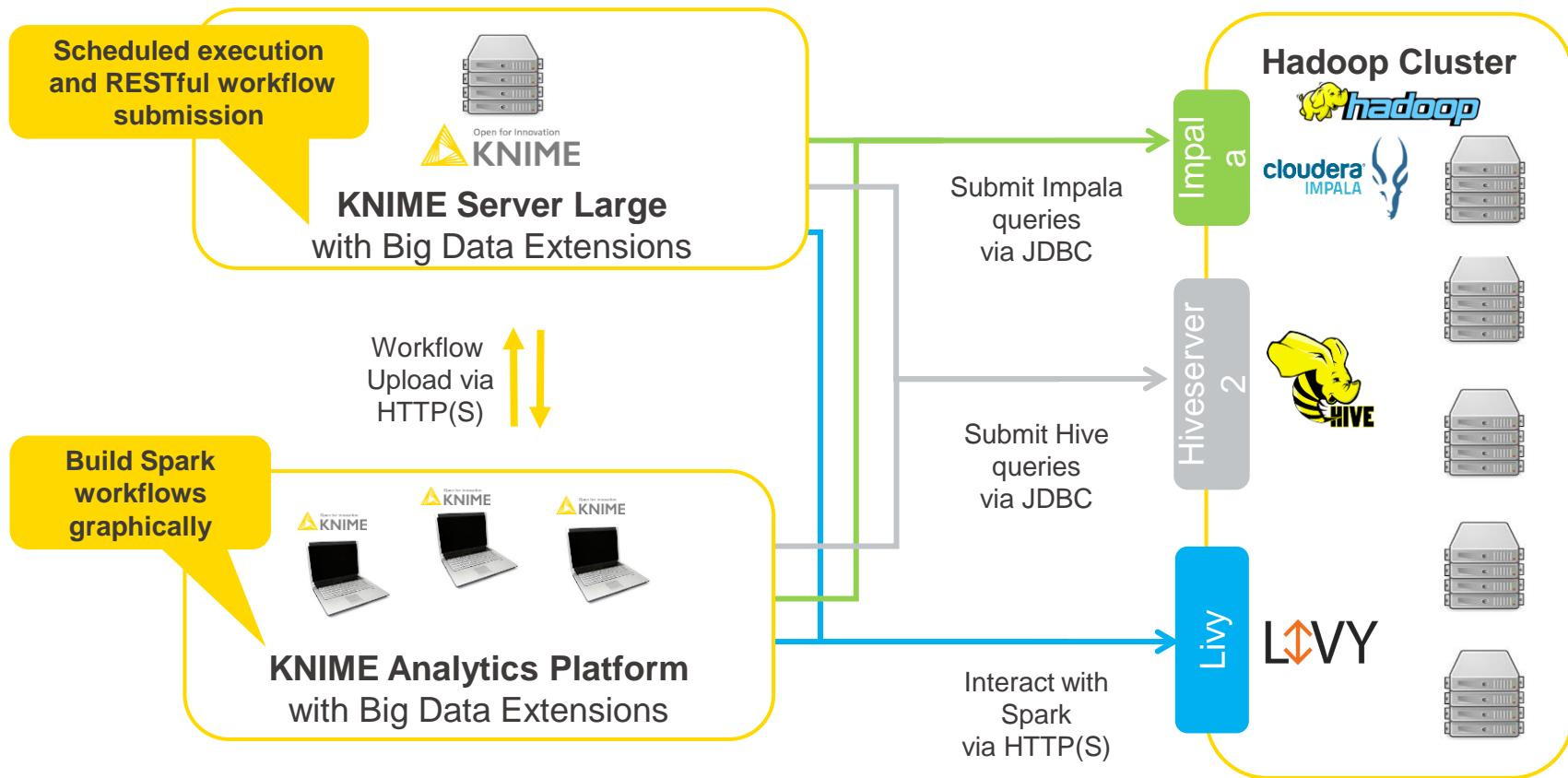


Spark Context

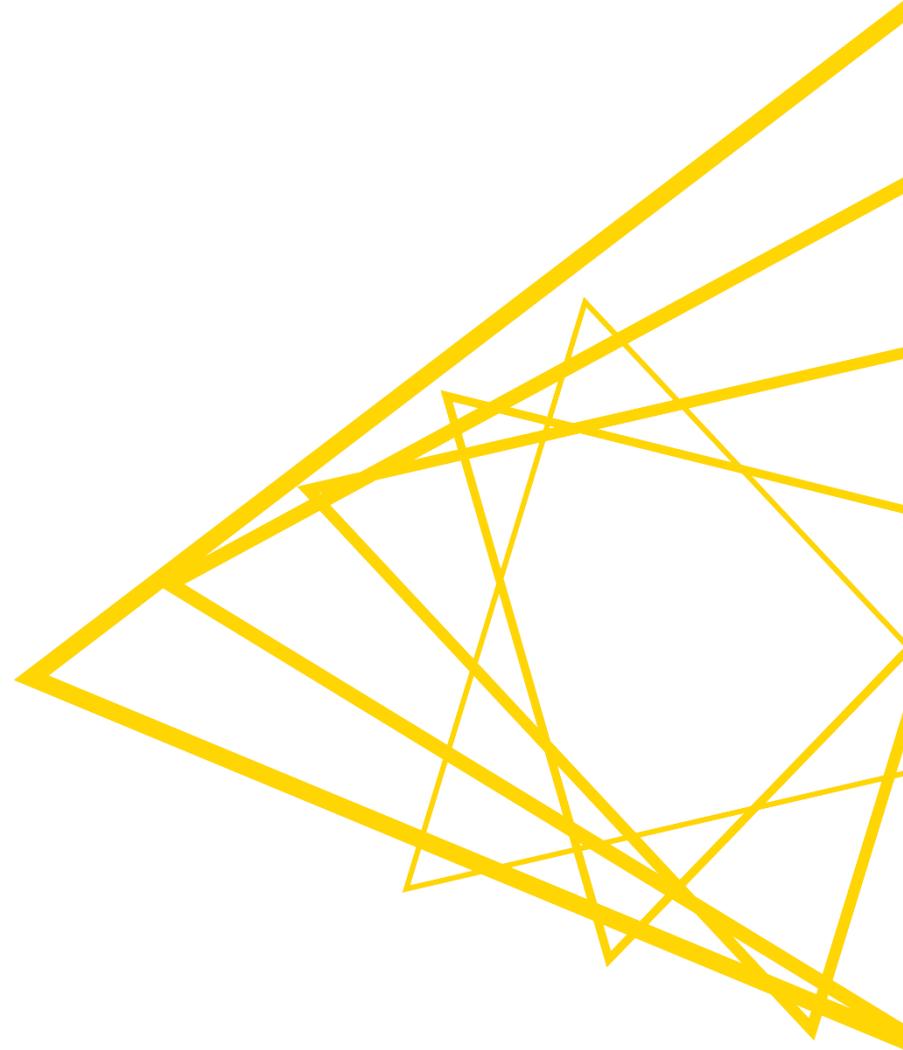
- Spark Context
 - Main entry point for Spark functionality
 - Represents connection to a Spark cluster
 - Allocates resources on the cluster



Big Data Architecture with KNIME

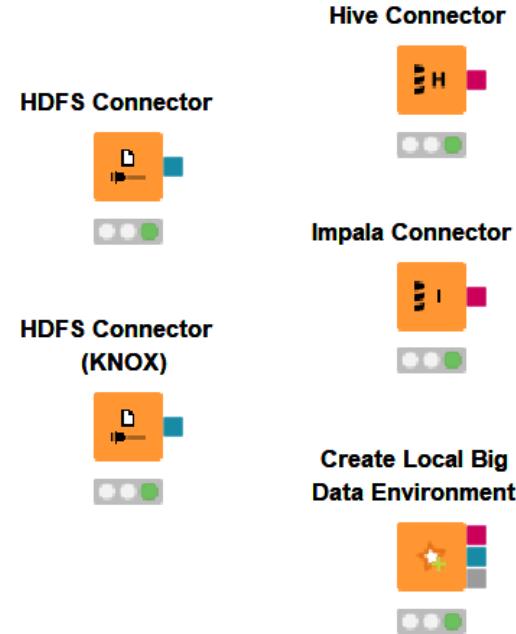


In-Database Processing on Hadoop



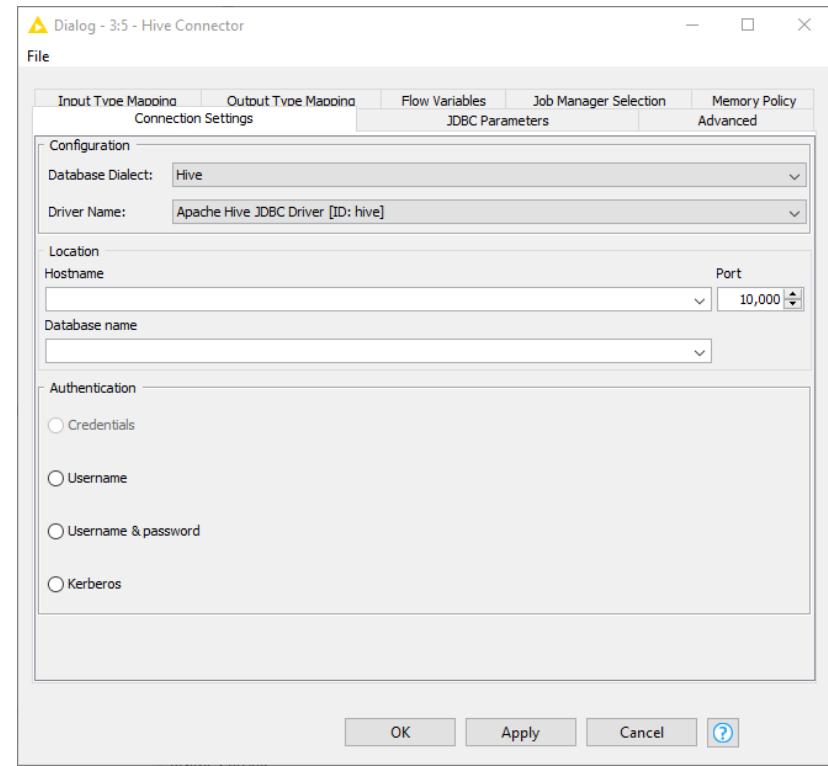
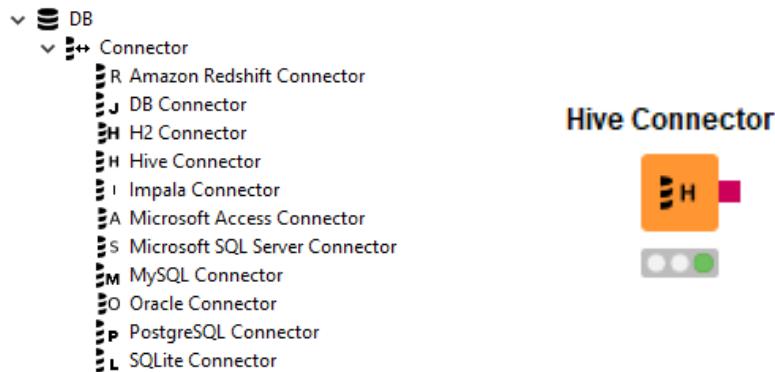
KNIME Big Data Connectors

- Package required drivers/libraries for HDFS, Hive, Impala access
- Preconfigured database connectors
 - Hive
 - Impala

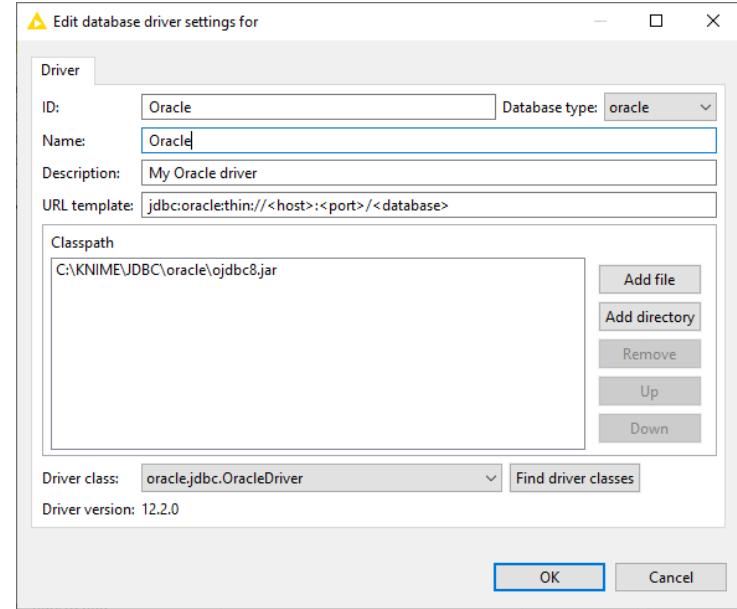
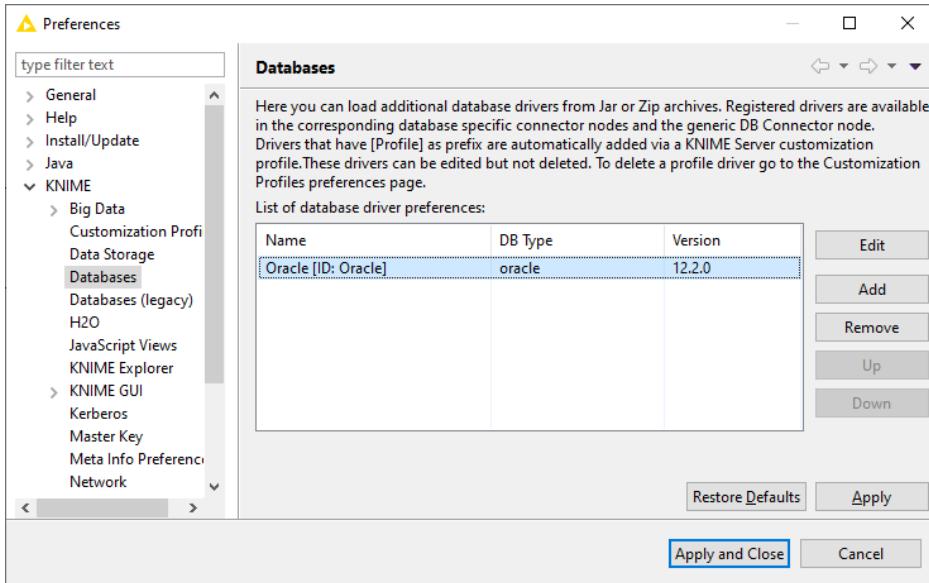


Hive Connector

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



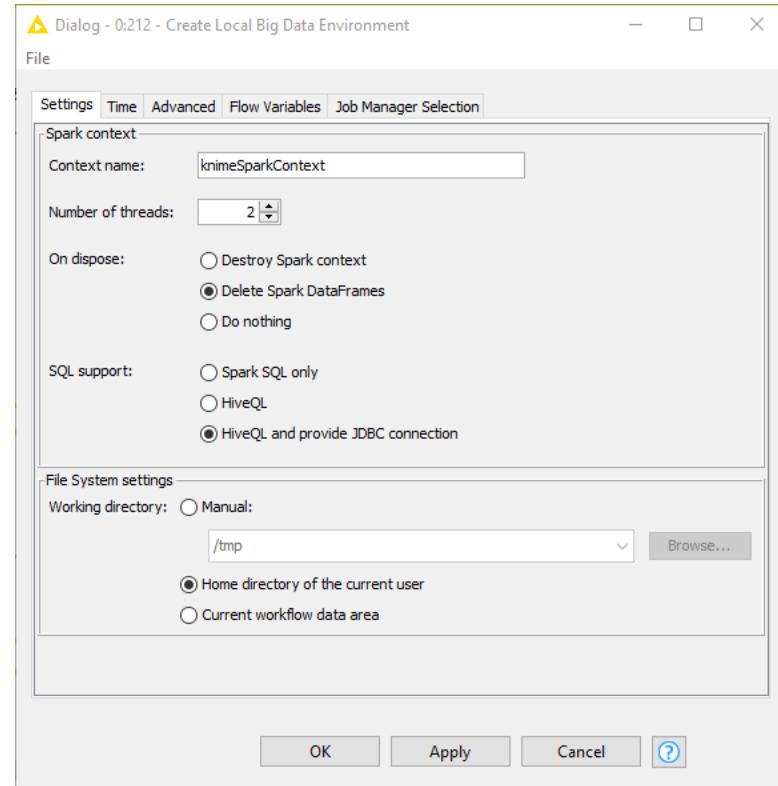
Preferences



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

Create Local Big Data Environment



Section Exercise – 01_Hive_Connect

Execute the workflow 00_Setup_Hive_Table to create a local big data environment with the data used in this training.

On the workflow implemented in the previous section to predict missing COW values, move execution from database to Hive. That means:

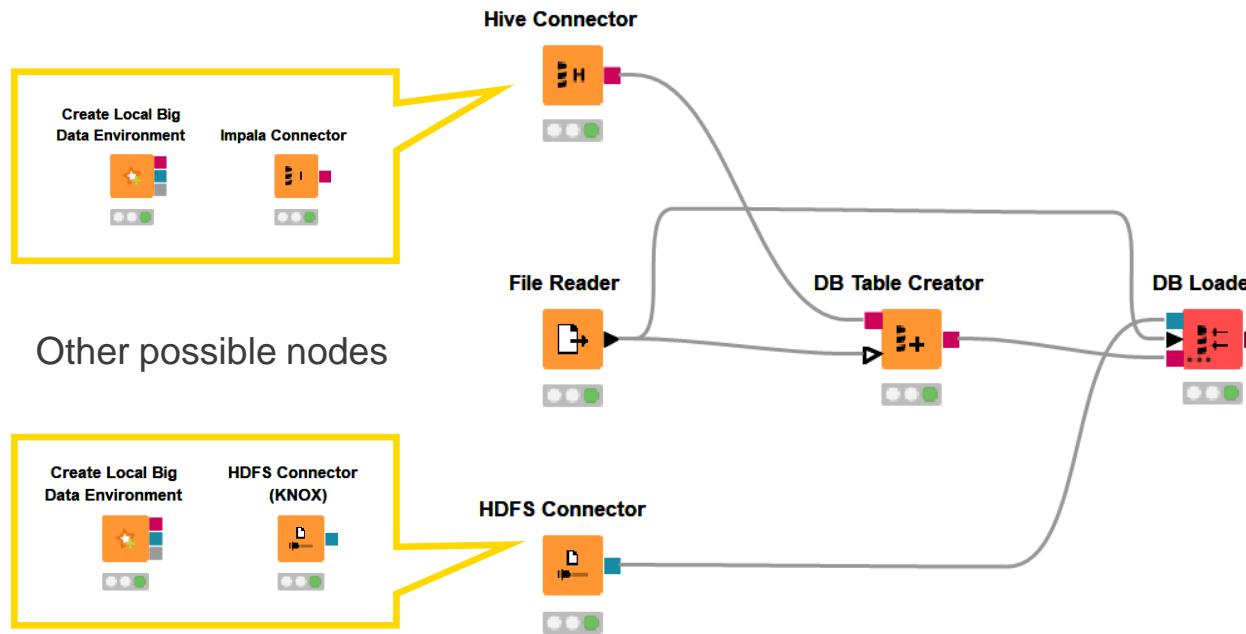
- change this workflow to run on the ss13pme table on the Hive database in your local big data environment.

Write/Load Data into Hadoop

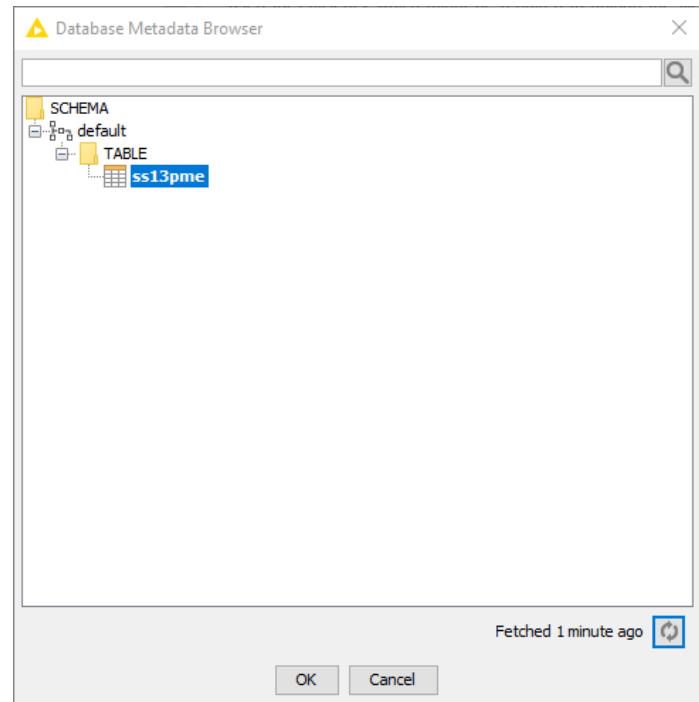
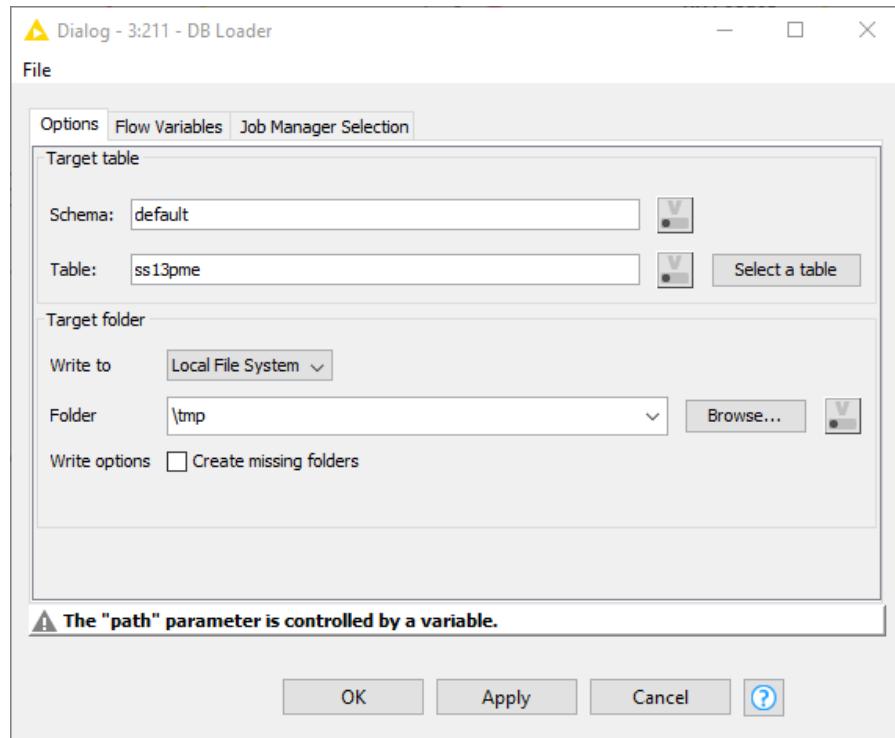


Loading Data into Hive/Impala

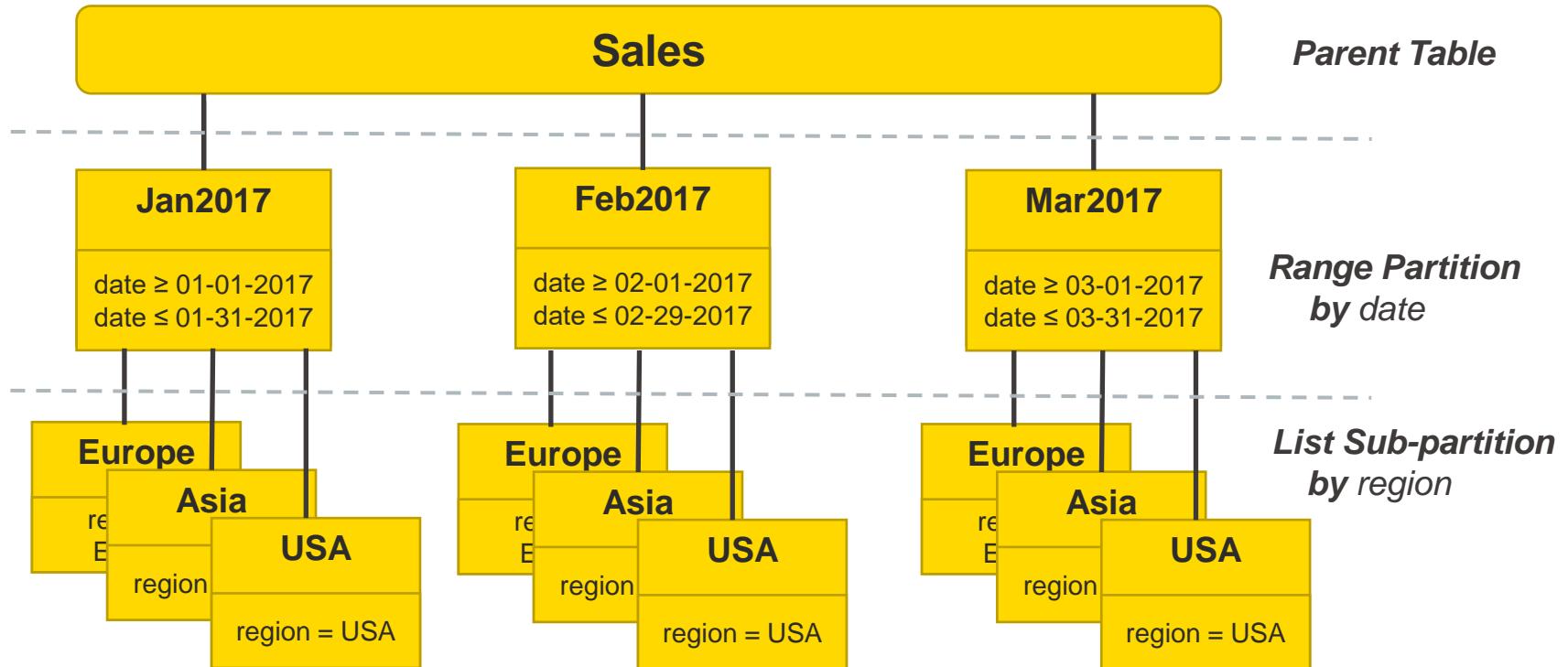
- Connectors are from KNIME Big Data Connectors Extension
- Use DB Table Creator and DB Loader from regular DB framework



DB Loader



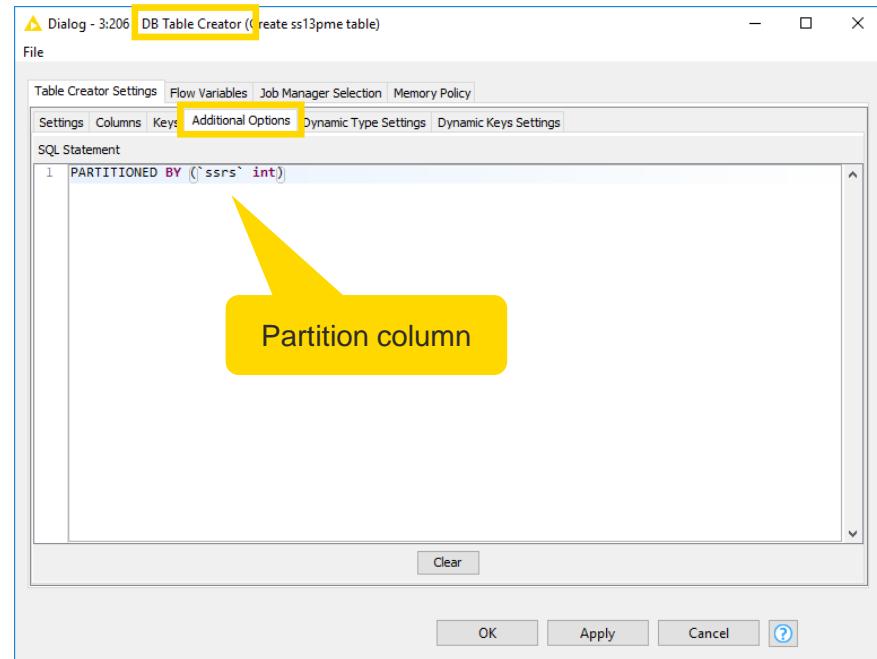
Hive Partitioning



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

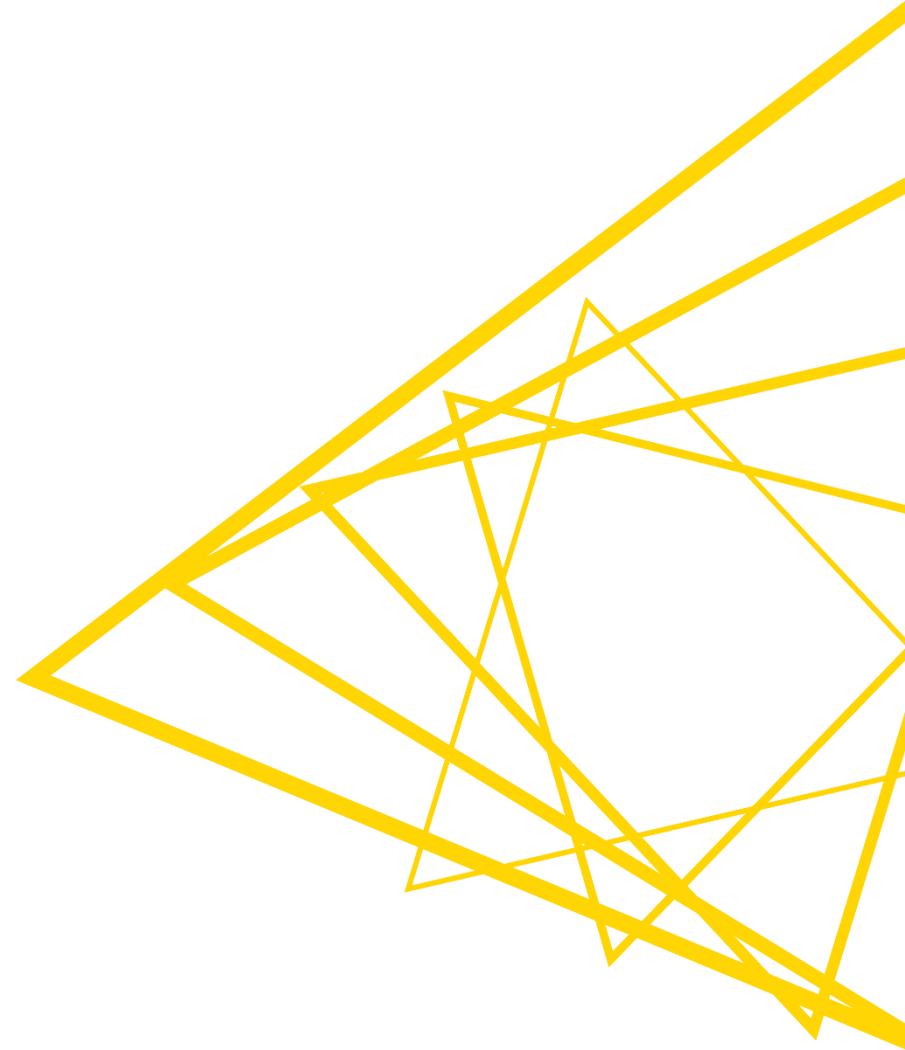


Section Exercise – 02_Hive_WritingToDB

Start from the workflow that implements the missing value strategy and write the results back into Hive. That is:

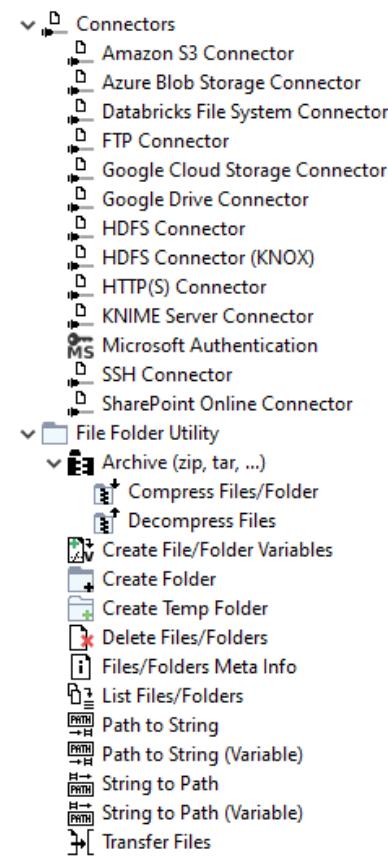
- write the results onto a new table named "newTable" in Hive using the HDFS Connection of the Local Big Data Environment along with both DB Table Creator and DB Loader nodes.

HDFS File Handling

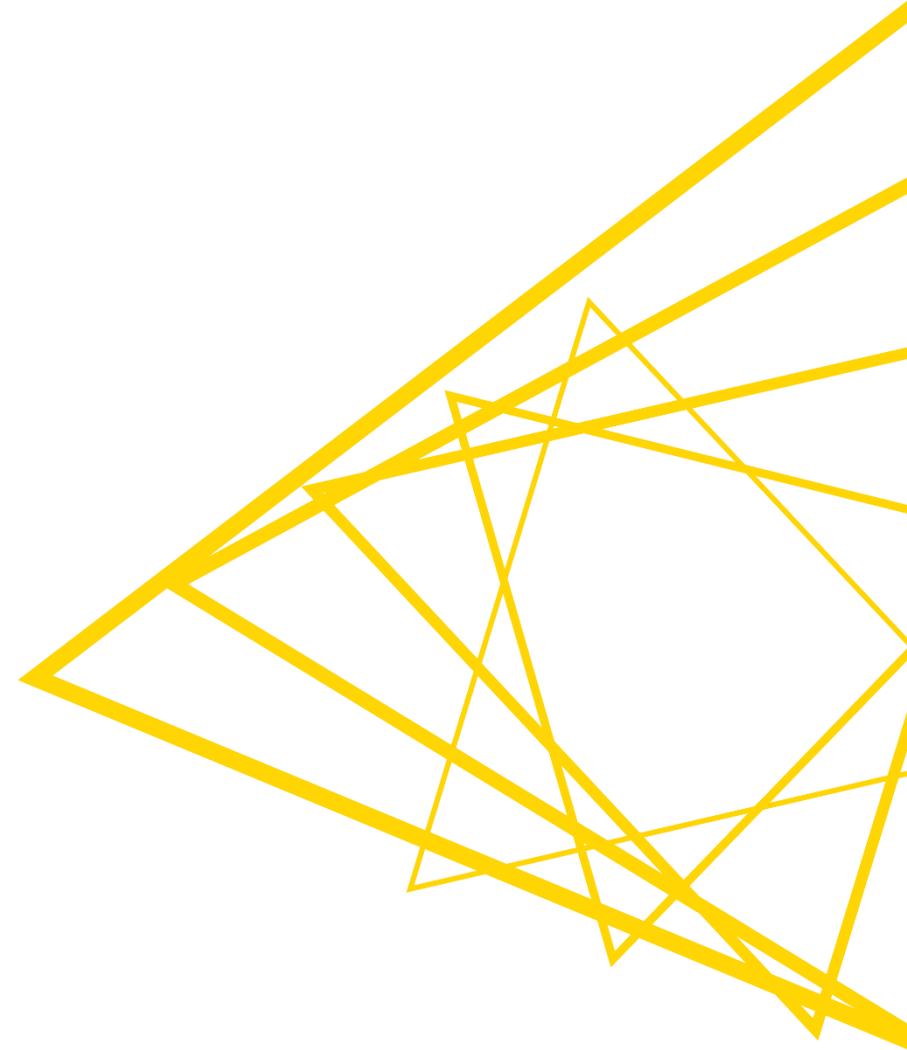


File Handling Nodes

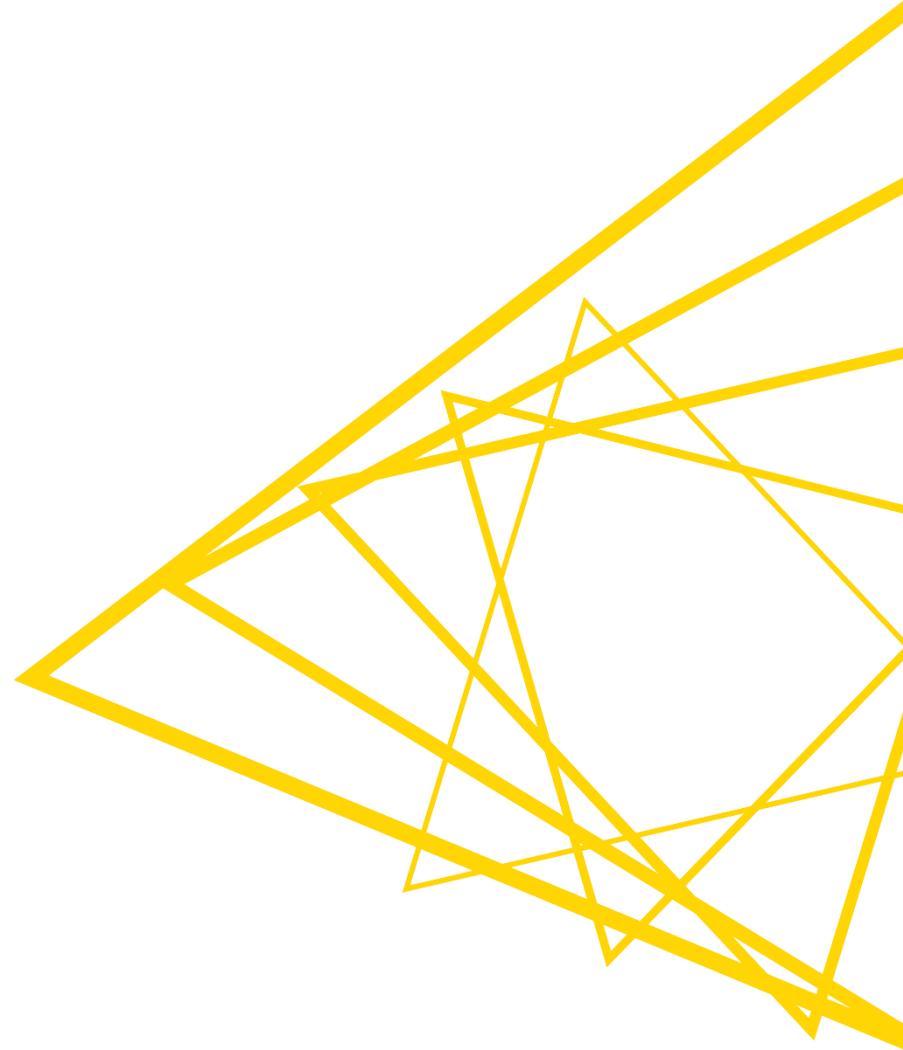
- Connection/Administration
 - HDFS Connection
 - HDFS File Permission
- Utilize the existing remote file handling nodes
 - Transfer Files
 - Create Folders
 - Delete Files/Folders
 - Compress and Decompress
 - [Full documentation available](#)



Ready for Spark?



KNIME Extension for Apache Spark

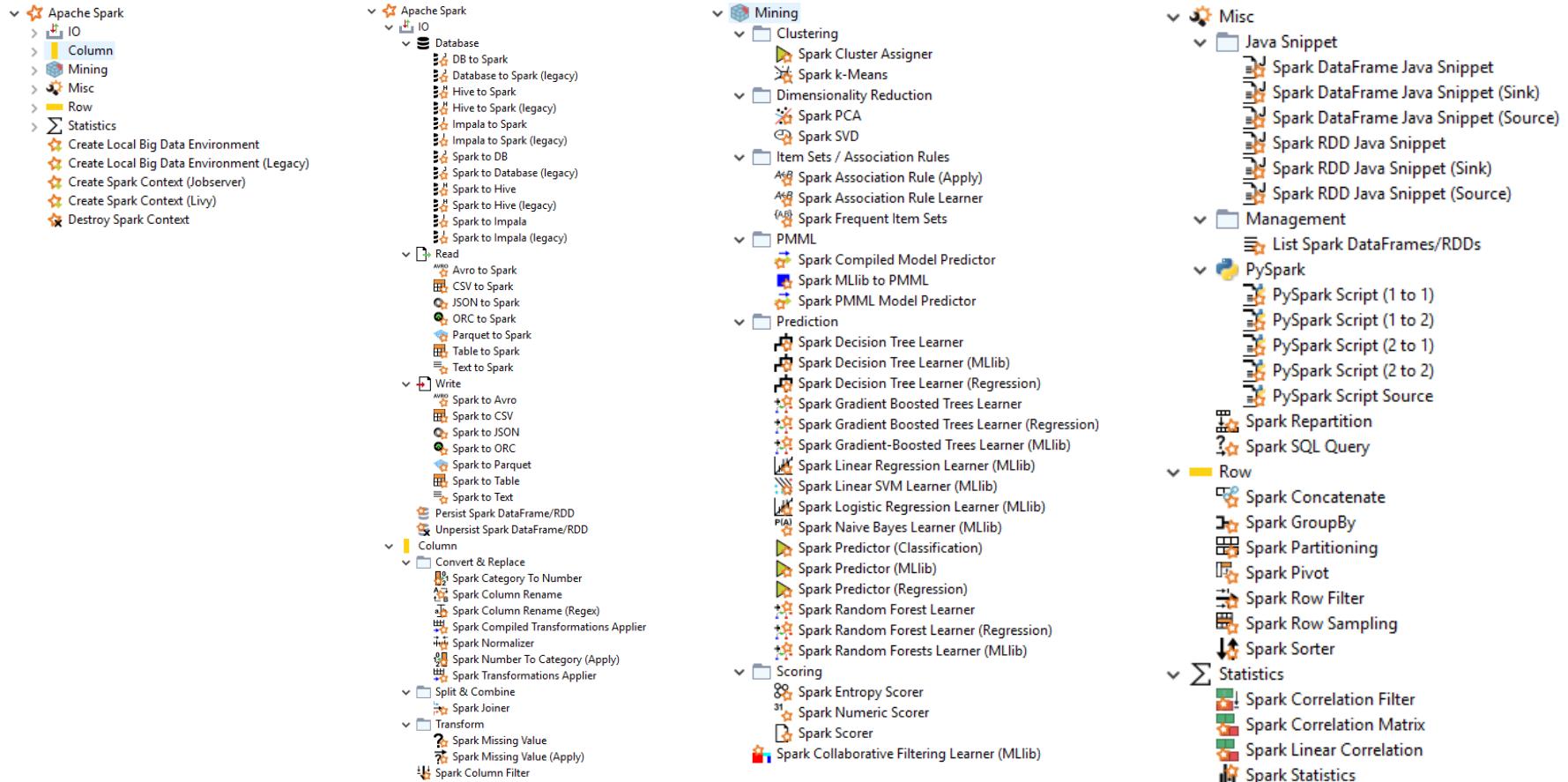


Spark: Machine Learning on Hadoop

- Runs on Hadoop
- Supported Spark Versions
 - 1.2, 1.3, 1.5, 1.6, 2.x
- One KNIME extension for all Spark versions
- 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 Integration in KNIME



Spark Contexts: 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
 - Good for production use
- Create Databricks Environment
 - Runs Spark on a remote Databricks cluster
 - Good for large-scale production use

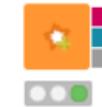
Create Local Big Data Environment



Create Spark Context (Livy)

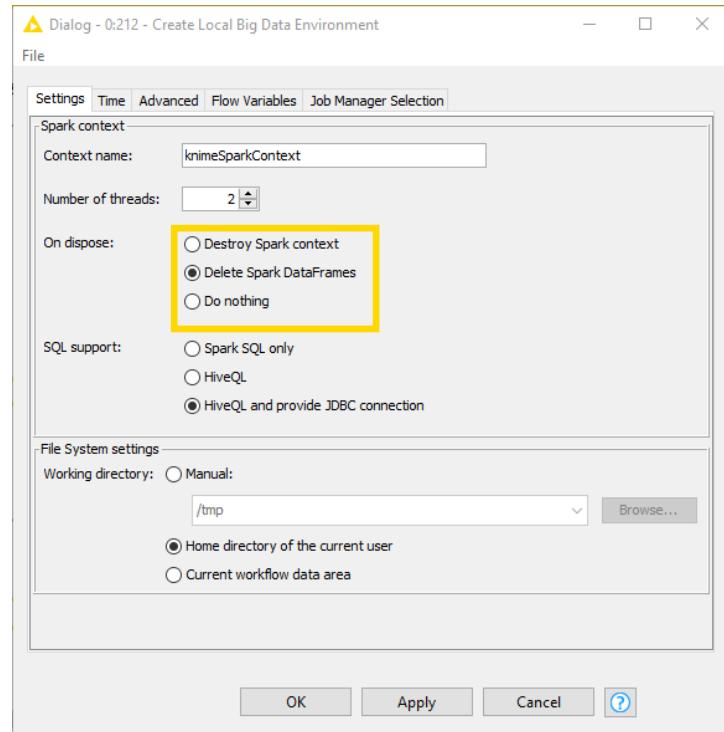
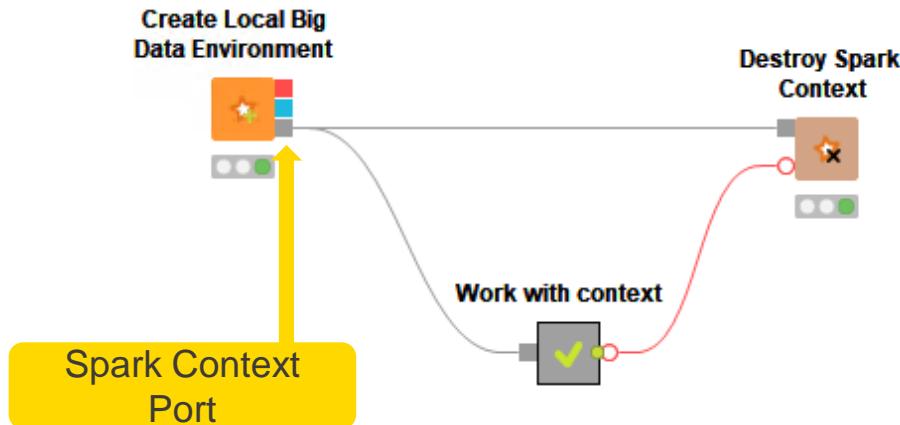


Create Databricks Environment



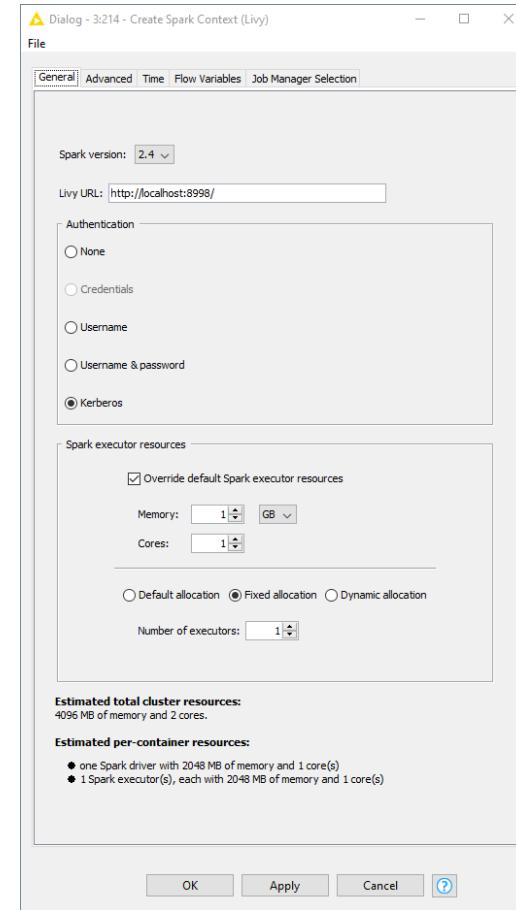
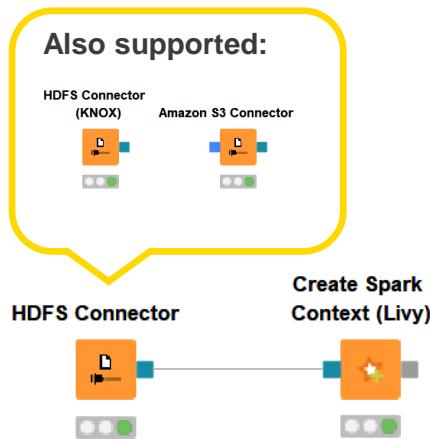
Spark Contexts: Using, Destroying

- Spark Context port is required by all Spark nodes
- Destroying a Spark Context destroys all Spark DataFrames within the context

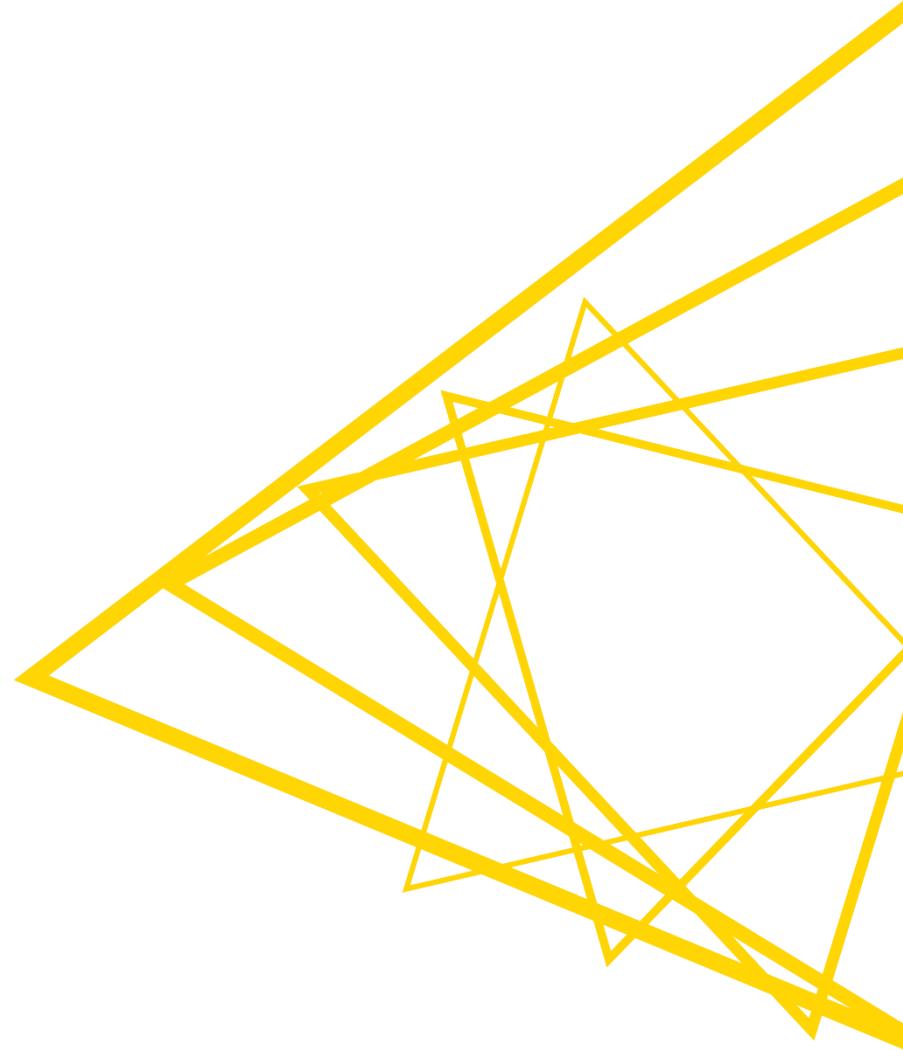


Create Spark Context (Livy)

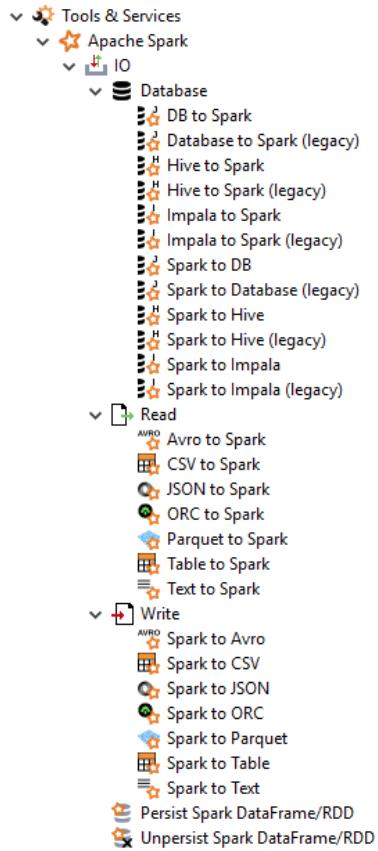
- Allows to use Spark nodes on clusters with Apache Livy
- Out-of-the-box compatibility with:
 - Hortonworks (v2.6.3 and higher)
 - Amazon EMR (v5.9.0 and higher)
 - Azure HDInsight (v3.6 and higher)



Import Data from KNIME or Hadoop



Import Data to Spark



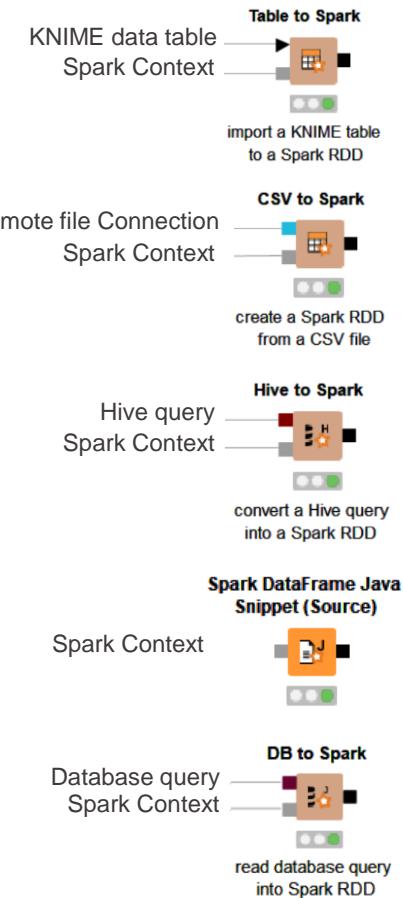
From KNIME

From CSV file in HDFS

From Hive

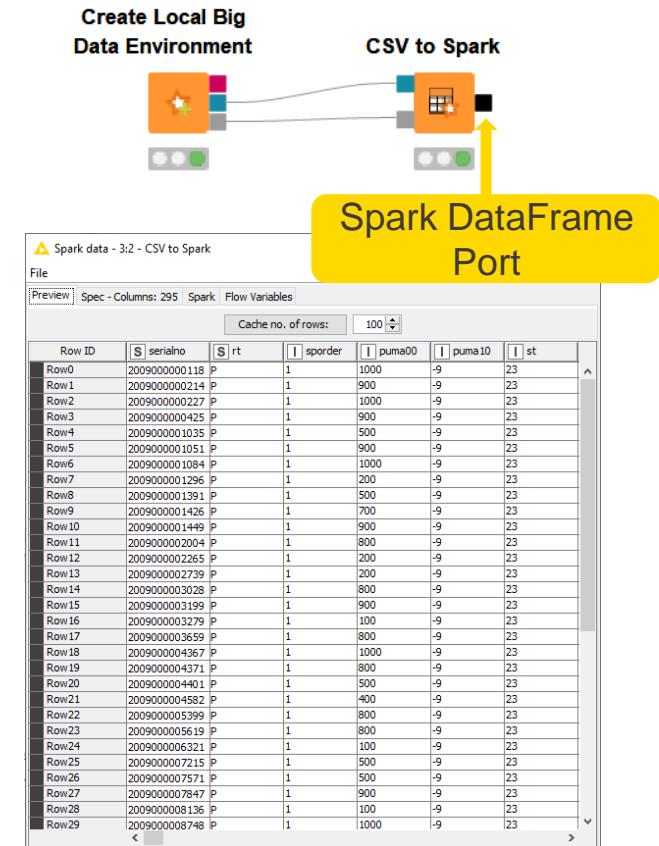
From other sources

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!

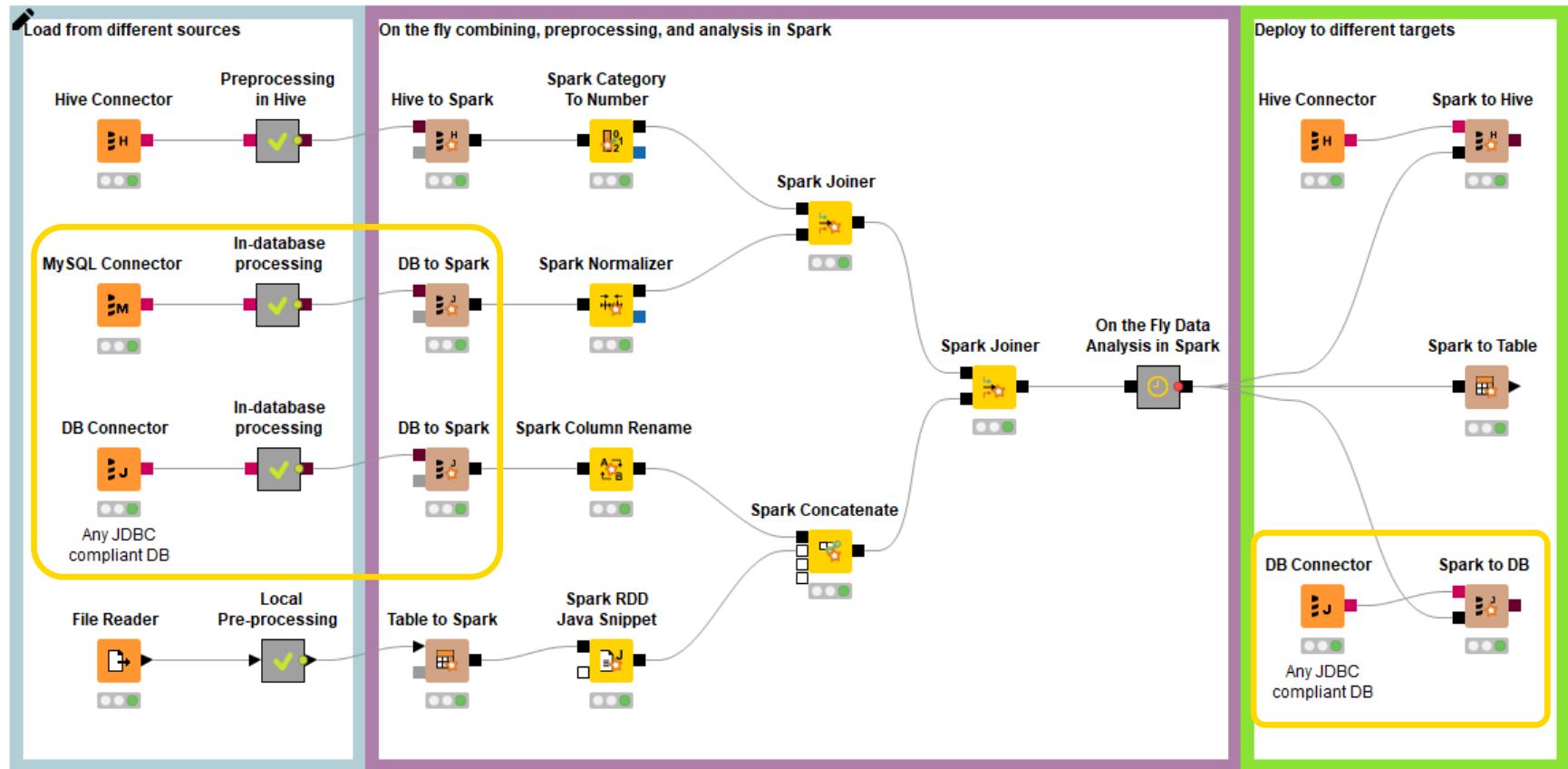


Section Exercise – 01_Spark_Connect

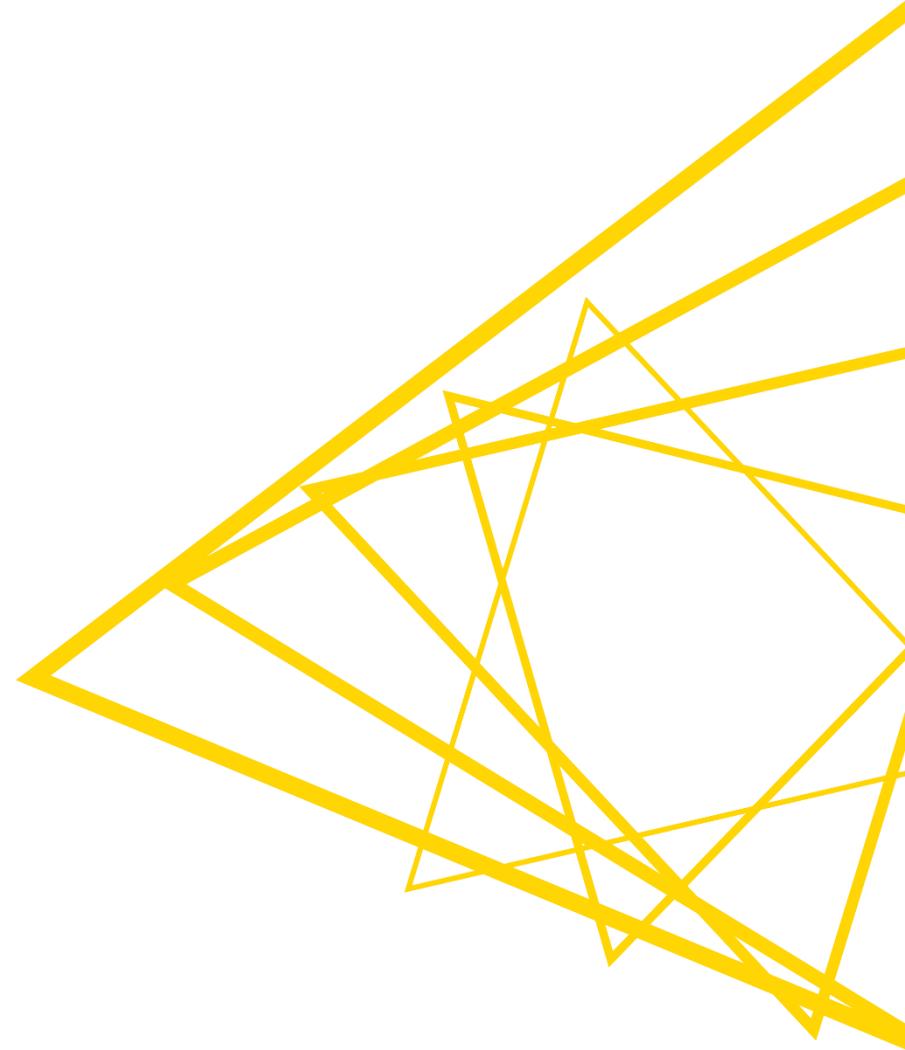
To do:

- Connect to Spark via the Create Local Big Data Environment node
- Import the ss13pme table from Hive into Spark

Virtual Data Warehouse

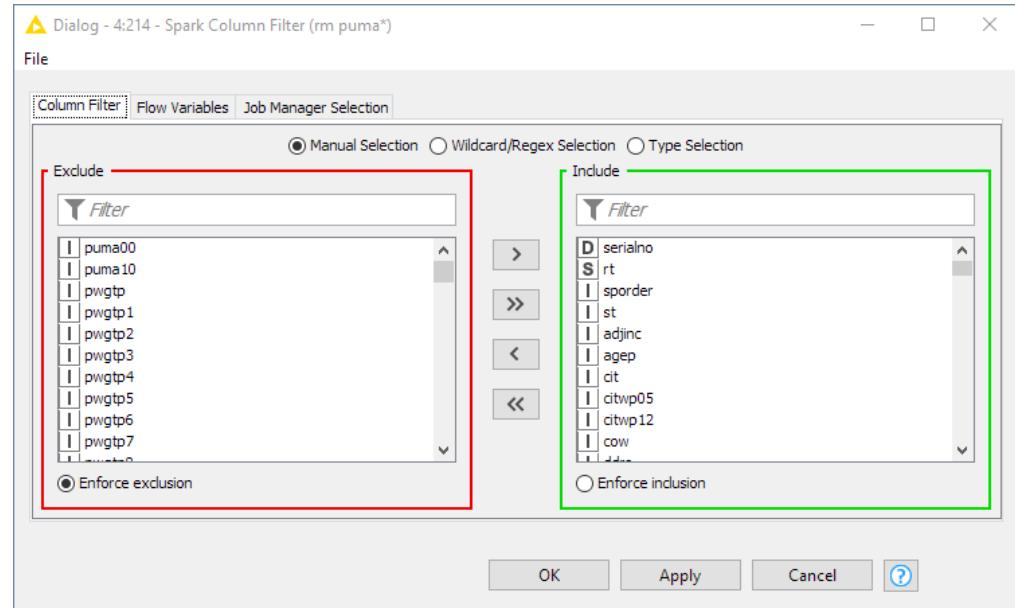


Pre-Processing with Spark



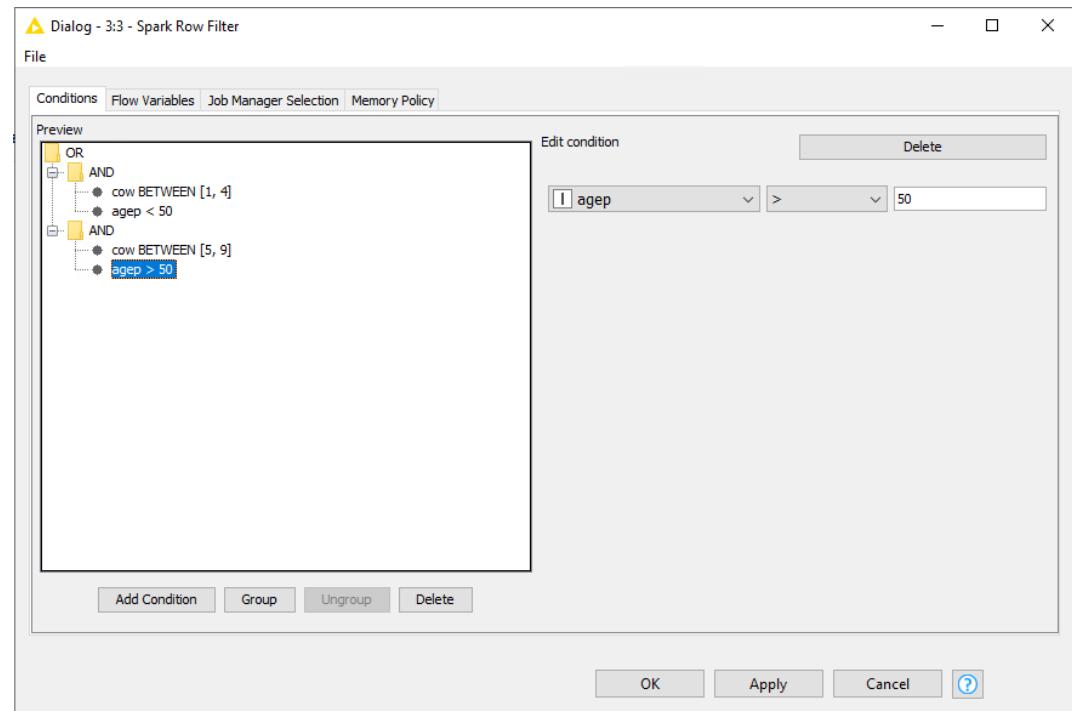
Spark Column Filter Node

- ▼ Tools & Services
- ▼ Apache Spark
 - > IO
 - ▼ Column
 - ▼ Convert & Replace
 - Spark Category To Number
 - Spark Column Rename
 - Spark Column Rename (Regex)
 - Spark Compiled Transformations Applier
 - Spark Normalizer
 - Spark Number To Category (Apply)
 - Spark Transformations Applier
 - ▼ Split & Combine
 - Spark Joiner
 - ▼ Transform
 - Spark Missing Value
 - Spark Missing Value (Apply)
 - Spark Column Filter
 - > Mining
 - > Misc
 - ▼ Row
 - Spark Concatenate
 - Spark GroupBy
 - Spark Partitioning
 - Spark Pivot
 - Spark Row Filter
 - Spark Row Sampling
 - Spark Sorter



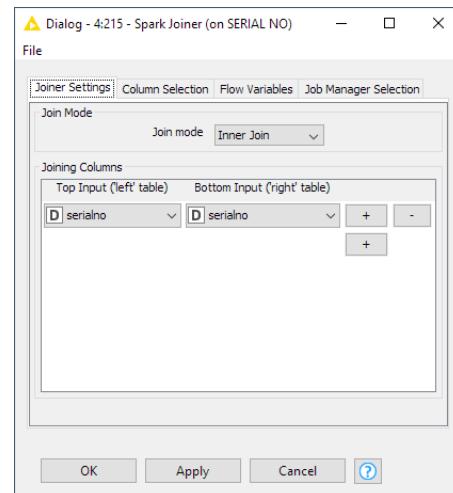
Spark Row Filter Node

- ▼  Tools & Services
- ▼  Apache Spark
 - >  IO
 - ▼  Column
 - ▼  Convert & Replace
 -  Spark Category To Number
 -  Spark Column Rename
 -  Spark Column Rename (Regex)
 -  Spark Compiled Transformations Applier
 -  Spark Normalizer
 -  Spark Number To Category (Apply)
 -  Spark Transformations Applier
 - ▼  Split & Combine
 -  Spark Joiner
 - ▼  Transform
 -  Spark Missing Value
 -  Spark Missing Value (Apply)
 -  Spark Column Filter
 - >  Mining
 - >  Misc
 - ▼  Row
 -  Spark Concatenate
 -  Spark GroupBy
 -  Spark Partitioning
 -  Spark Pivot
 -  **Spark Row Filter**
 -  Spark Row Sampling
 -  Spark Sorter



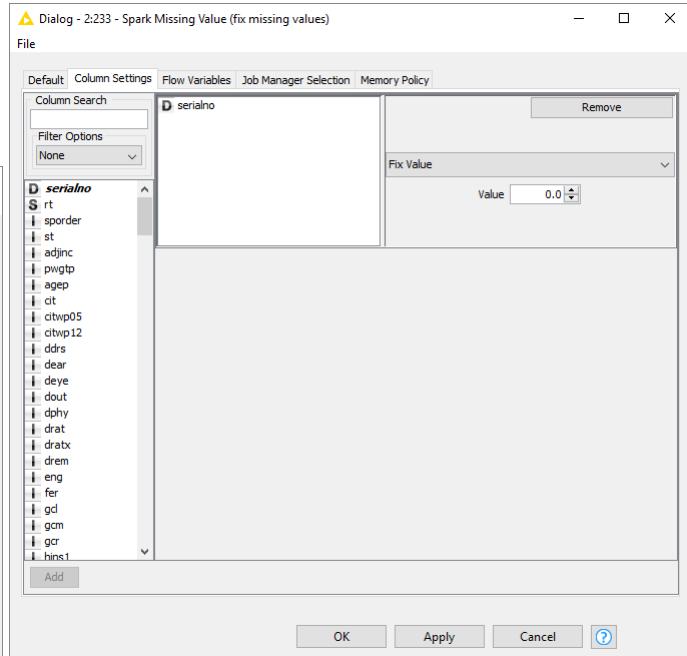
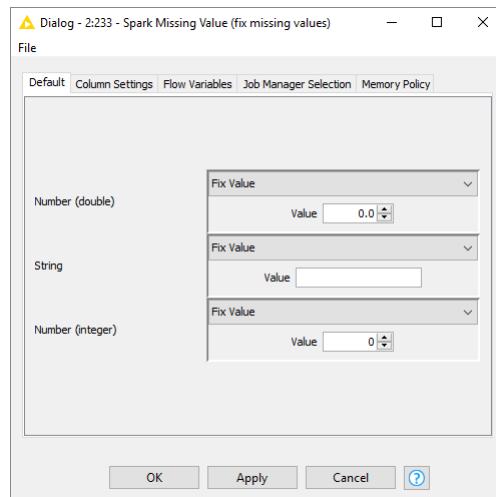
Spark Joiner Node

- ✓ Tools & Services
- ✓ Apache Spark
 - > IO
 - ✓ Column
 - Convert & Replace
 - Spark Category To Number
 - Spark Column Rename
 - Spark Column Rename (Regex)
 - Spark Compiled Transformations Applier
 - Spark Normalizer
 - Spark Number To Category (Apply)
 - Spark Transformations Applier
 - Split & Combine
 - SparkJoiner
 - Transform
 - Spark Missing Value
 - Spark Missing Value (Apply)
 - Spark Column Filter
 - > Mining
 - > Misc
 - ✓ Row
 - Spark Concatenate
 - Spark GroupBy
 - Spark Partitioning
 - Spark Pivot
 - Spark Row Filter
 - Spark Row Sampling
 - Spark Sorter



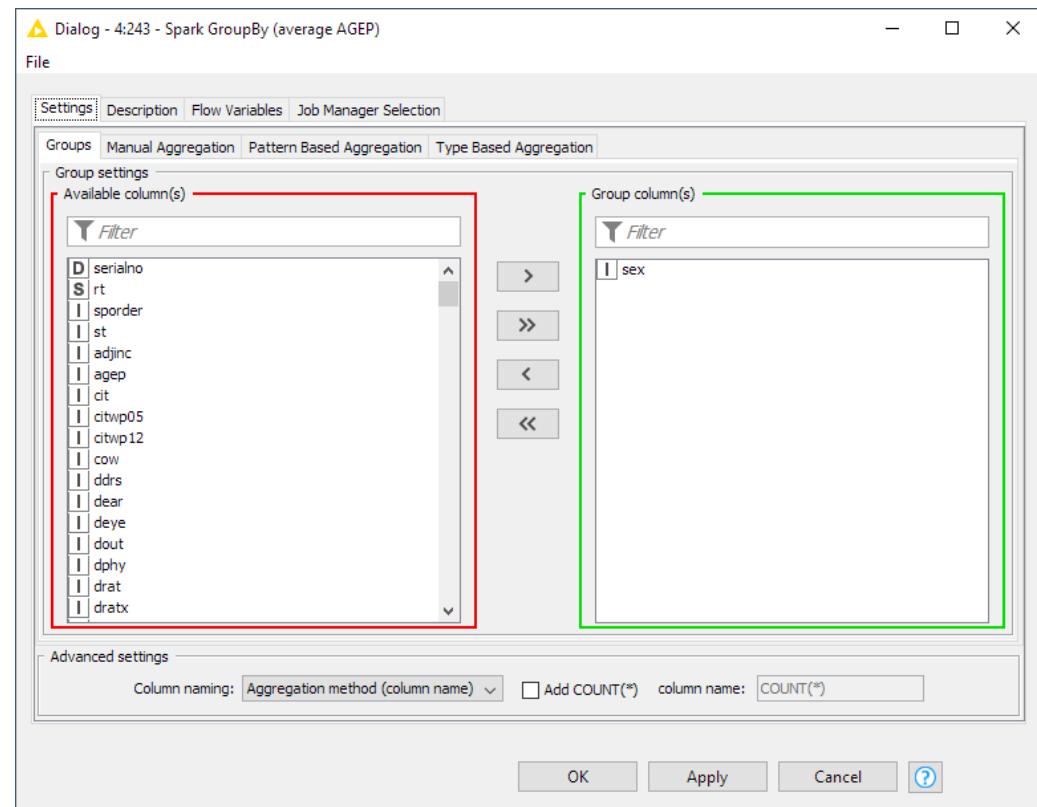
Spark Missing Value Node

- ❖ Tools & Services
 - ❖ Apache Spark
 - > IO
 - ❖ Column
 - Convert & Replace
 - ❖ Spark Category To Number
 - ❖ Spark Column Rename
 - ❖ Spark Column Rename (Regex)
 - ❖ Spark Compiled Transformations Applier
 - ❖ Spark Normalizer
 - ❖ Spark Number To Category (Apply)
 - ❖ Spark Transformations Applier
 - Split & Combine
 - ❖ Spark Joiner
 - Transform
 - ❖ Spark Missing Value
 - ❖ Spark Missing Value (Apply)
 - > Mining
 - > Misc
 - ❖ Row
 - ❖ Spark Concatenate
 - ❖ Spark GroupBy
 - ❖ Spark Partitioning
 - ❖ Spark Pivot
 - ❖ Spark Row Filter
 - ❖ Spark Row Sampling
 - ❖ Spark Sorter



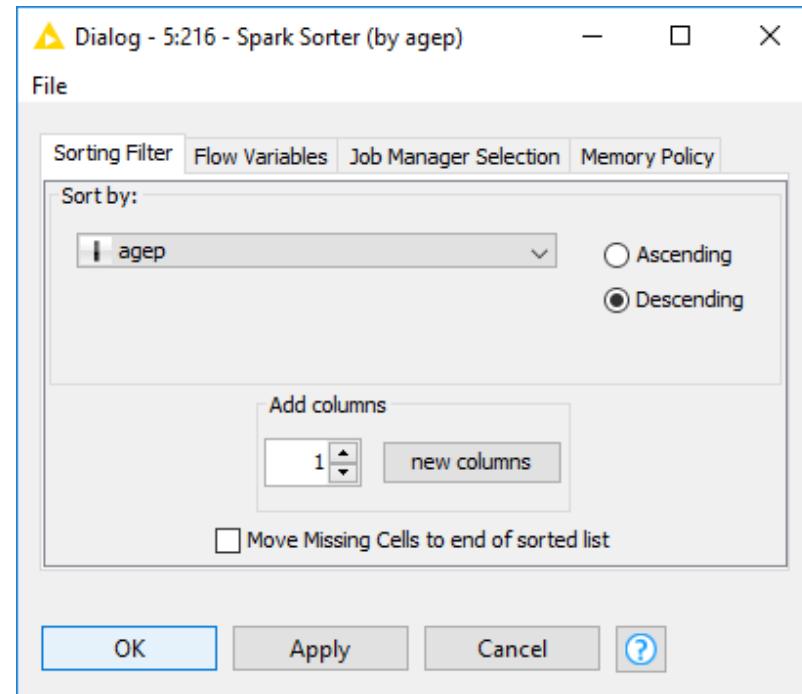
Spark GroupBy and Spark Pivot Nodes

- ❖ Tools & Services
- ❖ Apache Spark
 - > IO
 - ❖ Column
 - Convert & Replace
 - Spark Category To Number
 - Spark Column Rename
 - Spark Column Rename (Regex)
 - Spark Compiled Transformations Applier
 - Spark Normalizer
 - Spark Number To Category (Apply)
 - Spark Transformations Applier
 - Split & Combine
 - Spark Joiner
 - Transform
 - Spark Missing Value
 - Spark Missing Value (Apply)
 - Spark Column Filter
 - > Mining
 - > Misc
 - ❖ Row
 - Spark Concatenate
 - Spark GroupBy
 - Spark Partitioning
 - Spark Pivot
 - Spark Row Filter
 - Spark Row Sampling
 - Spark Sorter



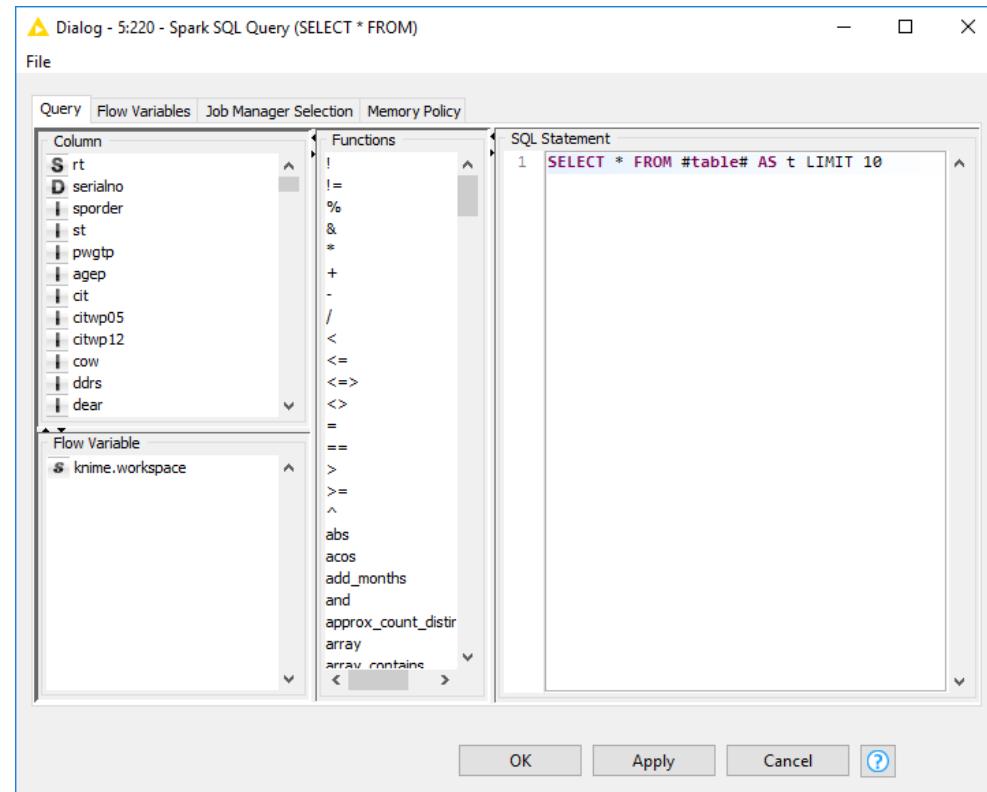
Spark Sorter Node

- ▼ Tools & Services
- ▼ Apache Spark
 - > IO
 - ▼ Column
 - ▼ Convert & Replace
 - Spark Category To Number
 - Spark Column Rename
 - Spark Column Rename (Regex)
 - Spark Compiled Transformations Applier
 - Spark Normalizer
 - Spark Number To Category (Apply)
 - Spark Transformations Applier
 - ▼ Split & Combine
 - Spark Joiner
 - ▼ Transform
 - Spark Missing Value
 - Spark Missing Value (Apply)
 - Spark Column Filter
 - > Mining
 - > Misc
 - ▼ Row
 - Spark Concatenate
 - Spark GroupBy
 - Spark Partitioning
 - Spark Pivot
 - Spark Row Filter
 - Spark Row Sampling
 - Spark Sorter



Spark SQL Query Node

- ▼ Tools & Services
 - ▼ Apache Spark
 - > IO
 - > Column
 - > Mining
 - ▼ Misc
 - > Java Snippet
 - > Management
 - ≡ List Spark DataFrames/RDDs
 - ?★ Spark SQL Query**
 - > Row
 - > Statistics



Section Exercise – 02_Spark_Preprocessing

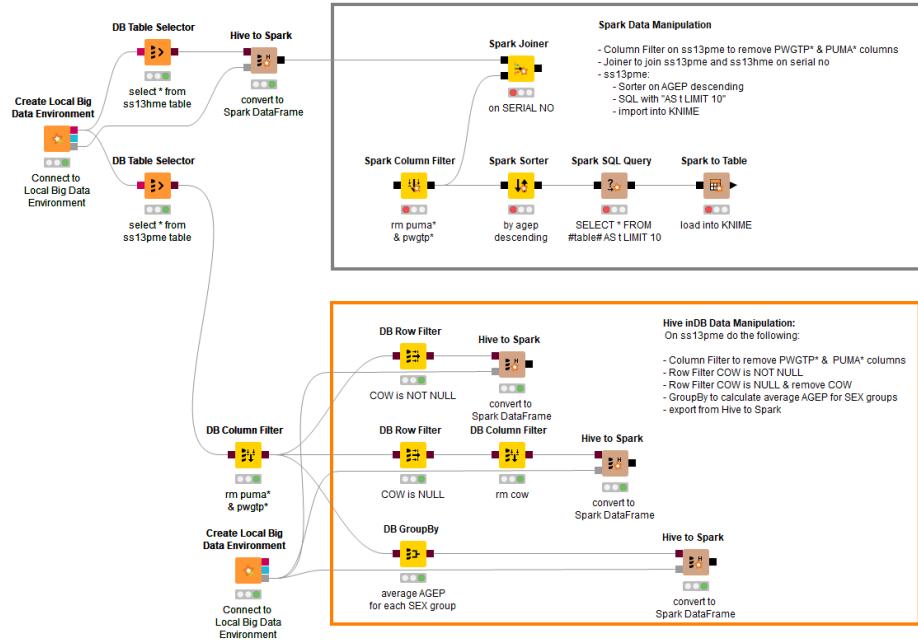
This exercise will demonstrate some data manipulation operations in Spark. It initially imports the ss13pme and ss13hme tables from Hive.

To do:

- Column Filter to remove PWGTP* and PUMA* columns
- Join with ss13hme on SERIAL NO
- Find rows with top ten AGEP in dataset and import them into KNIME
- Calculate average average AGEP per SEX group
- Split the dataset into two:
 - One where COW is null
 - One where COW is not null

Mix & Match

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



Modularize and Execute Your Own Spark Code: Java Snippets

The image shows the KNIME interface. On the left, there is a tree view under 'Tools & Services' with 'Apache Spark' expanded, showing 'Java Snippet' and several sub-snippets: 'Spark DataFrame Java Snippet', 'Spark DataFrame Java Snippet (Sink)', 'Spark DataFrame Java Snippet (Source)', 'Spark RDD Java Snippet', 'Spark RDD Java Snippet (Sink)', and 'Spark RDD Java Snippet (Source)'. Below this, a workflow diagram is displayed with four nodes connected sequentially: 'Spark DataFrame Java Snippet (Source)', 'Spark DataFrame Java Snippet', 'Spark DataFrame Java Snippet', and 'Spark DataFrame Java Snippet (Sink)'. The second and third nodes have small yellow icons next to them.

On the right, a dialog window titled 'Dialog - 3:5 - Spark DataFrame Java Snippet' is open. It has tabs for 'Java Snippet', 'Additional Libraries', 'Templates', 'Flow Variables', 'Job Manager Selection', and 'Memory Policy'. The 'Java Snippet' tab is selected. It contains a code editor with the following Java code:

```
// system imports
// Your custom imports:
// system variables
// Your custom variables:
// expression start
// *****
// Specify the row length
// *****
final int minLength = 10;

// Filters all rows from the Dataset that do not have the given length
Dataset<Row> result = dataFrame1.filter(new FilterFunction<Row>() {
    private static final long serialVersionUID = 1L;

    @Override
    public boolean call(Row row) throws Exception {
        // filter by length of value in first column
        return row.getString(0).length() > minLength;
    }
});
return result;
// expression end
```

Below the code editor is an 'Input' table with columns 'Name', 'Java Type', and 'Java Field'. There are three rows in the table:

Name	Java Type	Java Field

At the bottom of the dialog are buttons for 'OK', 'Apply', 'Cancel', and a help icon.

Modularize and Execute Your Own Spark Code: PySpark Script

The screenshot shows the KNIME interface with a focus on PySpark modularization.

Left Panel (Tools & Services):

- Tools & Services
- Apache Spark
- Misc
- PySpark
 - PySpark Script (1 to 1)
 - PySpark Script (1 to 2)
 - PySpark Script (2 to 1)
 - PySpark Script (2 to 2)
 - PySpark Script Source

Process Flow:

```
graph LR; A[PySpark Script Source] --> B[PySpark Script (1 to 1)];
```

PySpark Script (1 to 1) Dialog:

Script Tab:

```
1 # System imports
2 # Your custom imports:
3
4 # Flowvariables
5 # Your custom global variables:
6
7 # Initialization of Spark environment
8
9 # Custom pySpark code
10 # SparkSession can be used with variable spark
11 # The input DataFrame(s): [dataFrame1]
12 # The output DataFrame(s) must be: [resultDataFrame1]
13 resultDataFrame1 = dataFrame1.filter(dataFrame1.cow.isNull())
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
```

Flow Variables Tab:

- knime.workspace

Buttons at the bottom:

- Validate on Cluster
- Number of rows to validate on: 50

Data Preview:

```
resultDataFrame1(10 of 50 rows):
+---+---+---+---+---+---+---+---+---+---+---+
| serialno | rtisporde | puma00 | puma10 | stl | adjinc | pwgtp | agep | cit | citwp05 | citwp12 | cow | ddris | dear |
+---+---+---+---+---+---+---+---+---+---+---+
| 2009000000425 | P | 1 | 900 | -9 | 23 | 1085467 | 32 | 22 | 1 | null | null | 4 | 2 | 2 |
| 2009000001035 | P | 1 | 500 | -9 | 23 | 1085467 | 23 | 57 | 1 | null | null | 6 | 2 | 2 |
| 2009000001051 | P | 1 | 900 | -9 | 23 | 1085467 | 23 | 43 | 1 | null | null | 4 | 2 | 2 |
| 2009000001084 | P | 1 | 1000 | -9 | 23 | 1085467 | 51 | 27 | 1 | null | null | 11 | 2 | 2 |
| 2009000001426 | P | 1 | 700 | -9 | 23 | 1085467 | 17 | 47 | 1 | null | null | 1 | 2 | 2 |
```

Dialog Buttons:

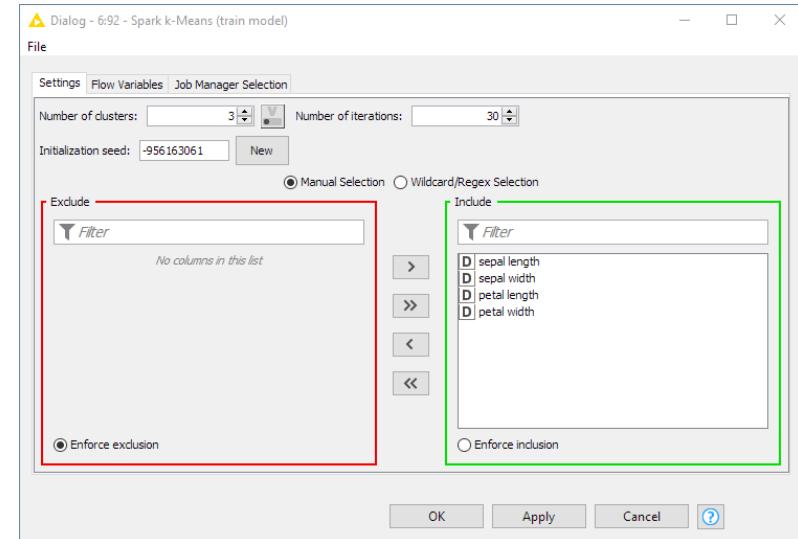
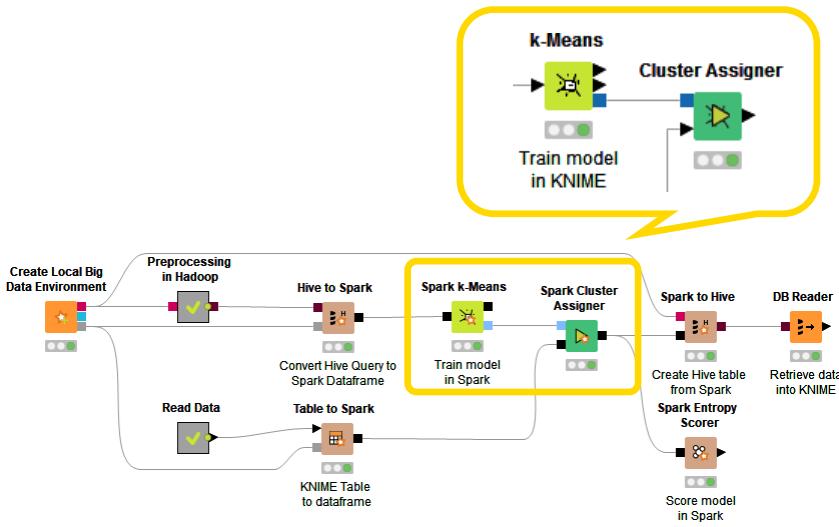
- OK
- Apply
- Cancel
- ?

Machine Learning with Spark



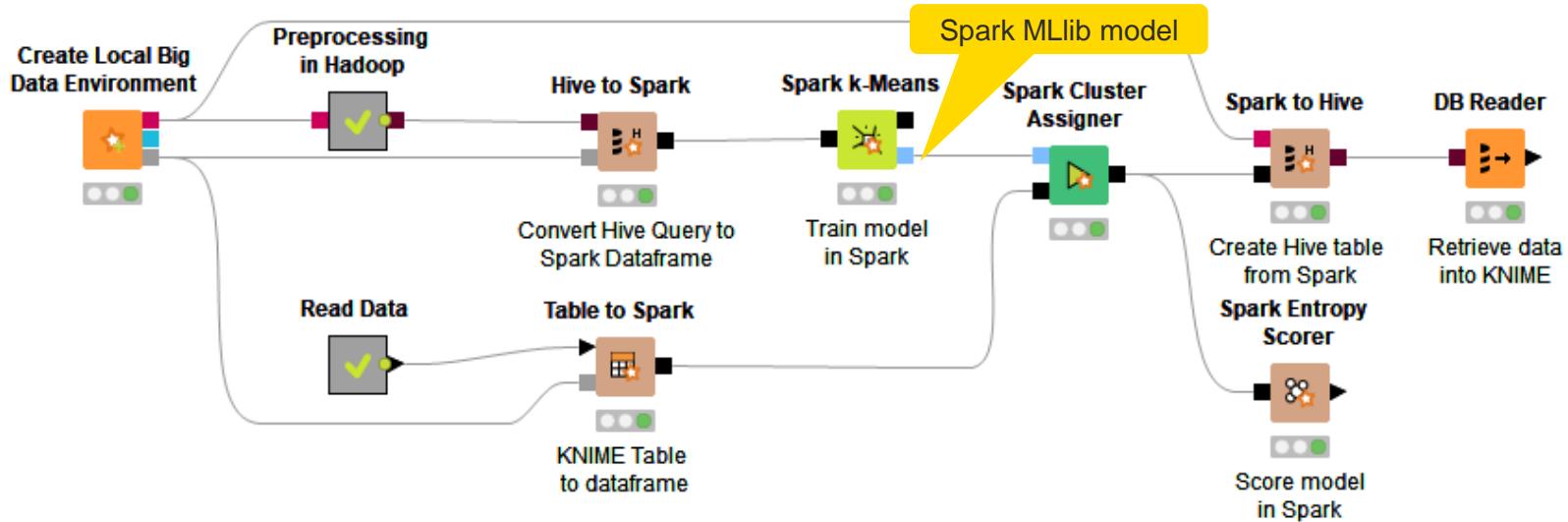
MLlib Integration: Familiar Usage Model

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



MLlib Integration: Spark MLlib Model Port

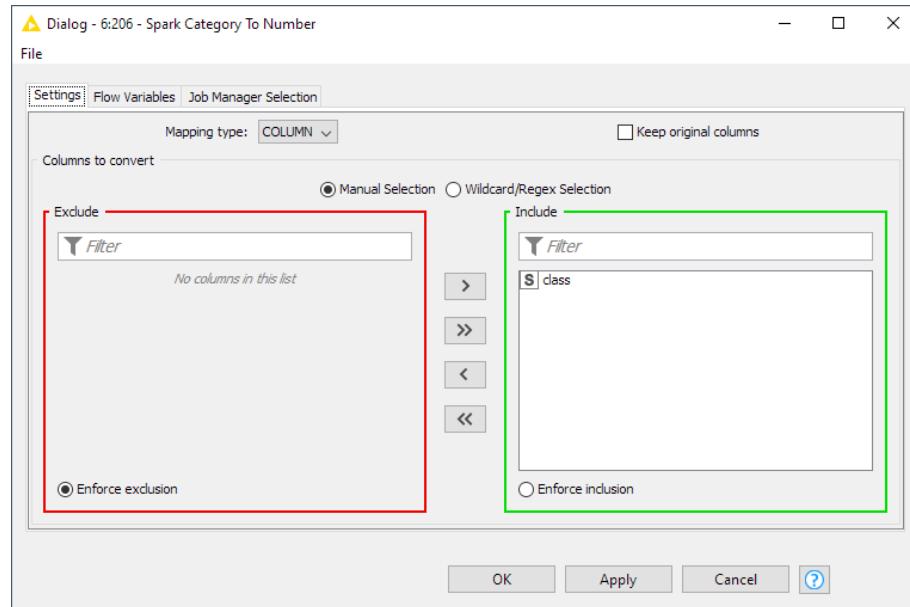
- MLlib model ports for model transfer
- Model ports provide more information about the model itself



MLlib Integration: Categorical features

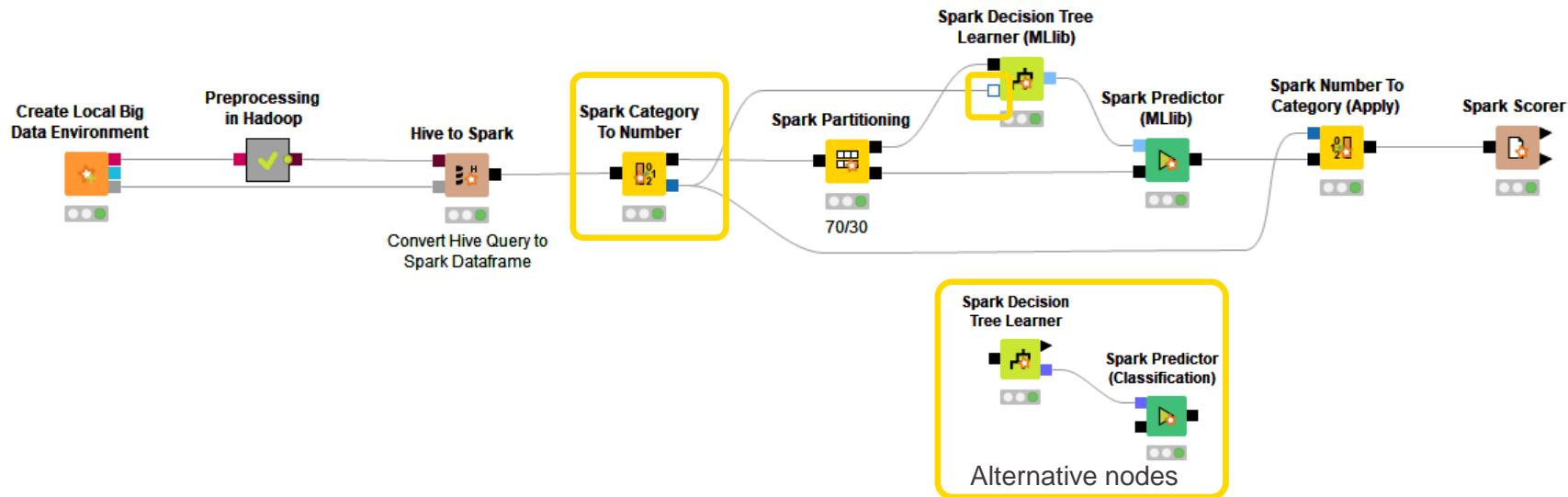
- ▼ Tools & Services
 - ▼ Apache Spark
 - ▶ IO
 - ▼ Column
 - ▼ Convert & Replace
 - Spark Category To Number
 - Spark Column Rename
 - Spark Column Rename (Regex)
 - Spark Compiled Transformations Applier
 - Spark Normalizer
 - Spark Number To Category (Apply)
 - Spark Transformations Applier
 - ▼ Split & Combine
 - Spark Joiner
 - ▼ Transform
 - Spark Missing Value
 - Spark Missing Value (Apply)
 - Spark Column Filter
 - ▶ Mining
 - ▶ Misc
 - ▼ Row
 - Spark Concatenate
 - Spark GroupBy
 - Spark Partitioning
 - Spark Pivot
 - Spark Row Sampling
 - Spark Sorter

- MLLib learner nodes only support numeric features and labels
- String columns (with categorical values) need to be mapped to numeric first



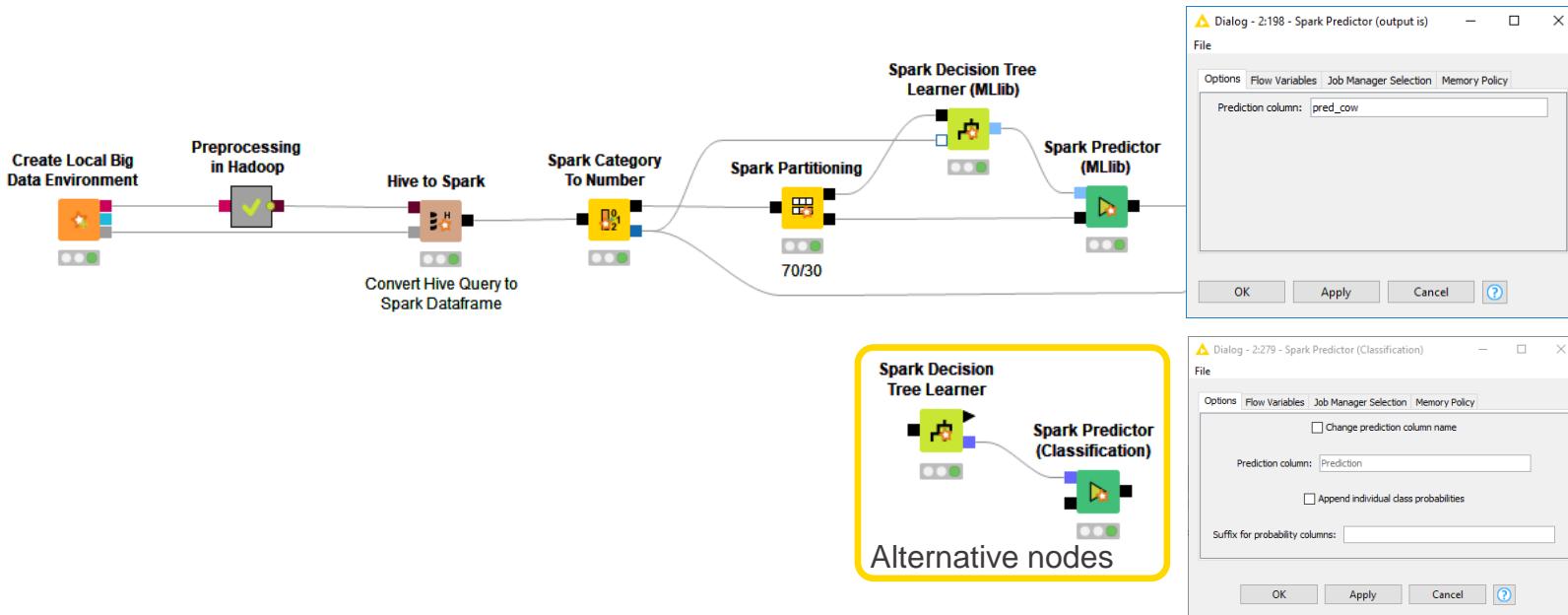
MLlib Integration: Categorical Values for Decision Tree Algorithms

- MLlib tree algorithms have optional PMML input port
 - If connected, hints to Decision Tree algorithm which numeric columns are categorical in nature
 - Improves performance in some cases



Spark Predictor Node

- Spark Predictor (MLlib) assigns labels based on an MLlib model
- Supports all **supervised** classification & regression MLlib models
- Spark.ml models have a separate learner/predictor



Section Exercise – 03_Spark_Modelling

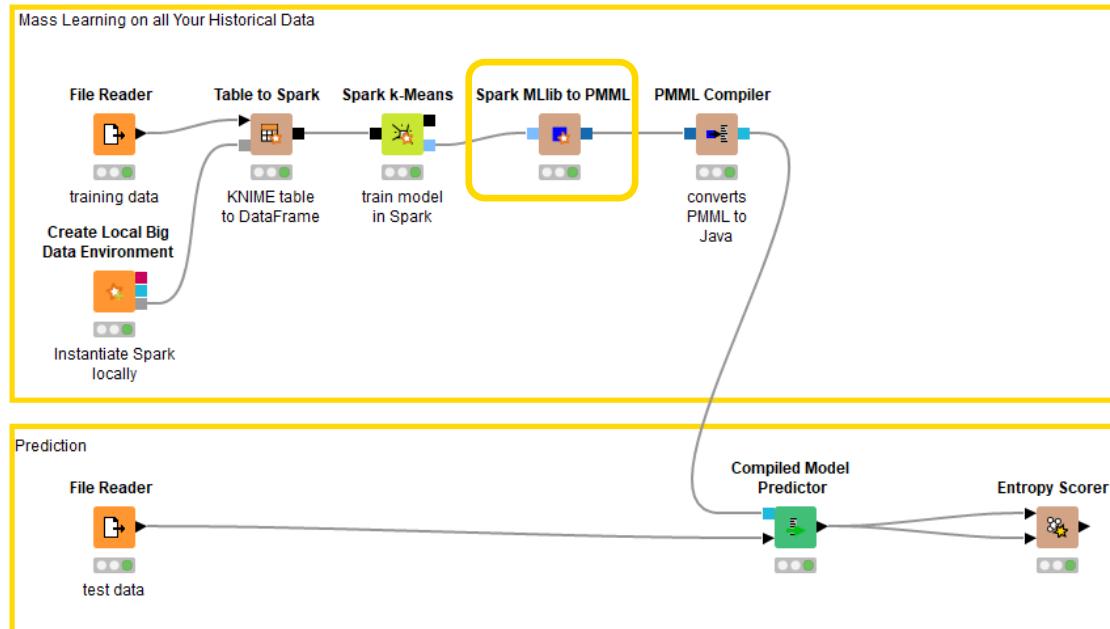
On the ss13pme table, the current workflow separates the rows where COW is null, from those where COW is not null, and then modifies COW to be zero-based.

To do:

- Where COW is not null:
 - Fix missing values in feature columns
 - Train a decision tree on COW on data rows
- Where COW is null:
 - Remove COW column
 - Apply the decision tree model to predict COW

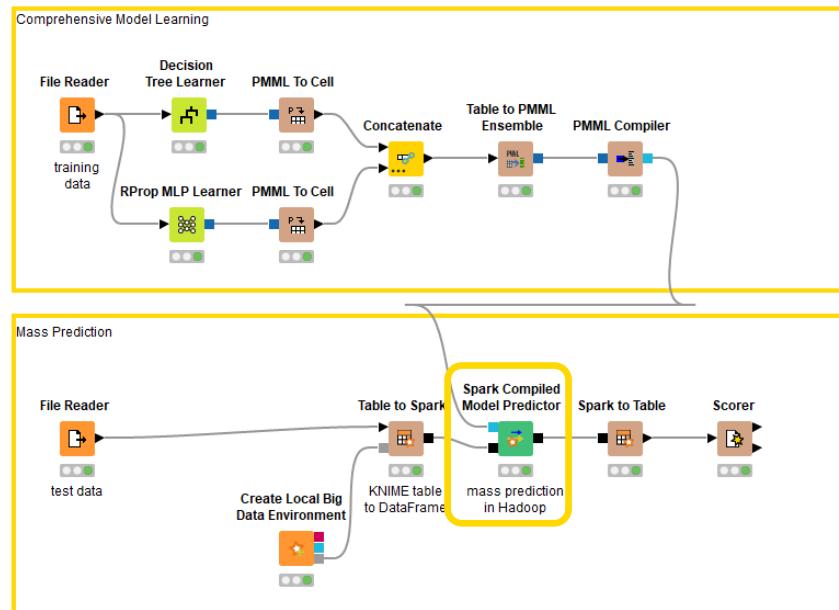
Mass Learning in Spark – Conversion to PMML

- Mass learning on Hadoop
- Convert supported MLlib models to PMML

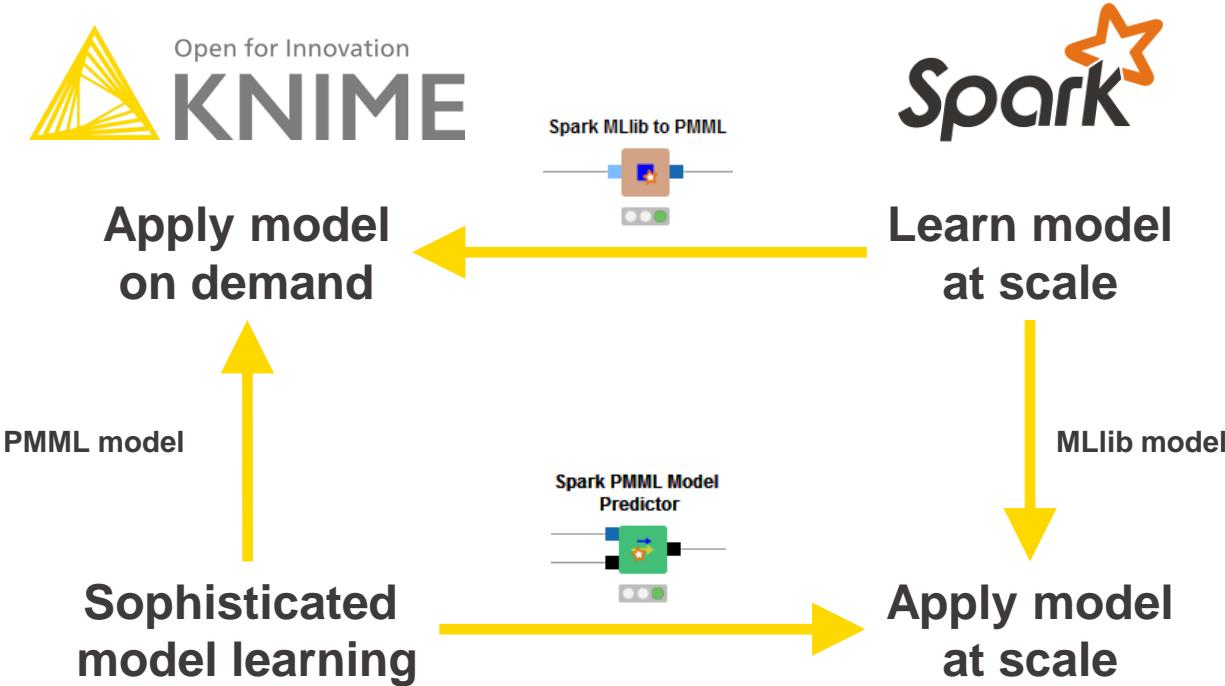


Sophisticated Model Learning in KNIME - Mass Prediction in Spark

- Supports KNIME models and pre-processing steps
- Sophisticated model learning in KNIME
- Mass prediction in Spark using Spark PMML Model Predictor

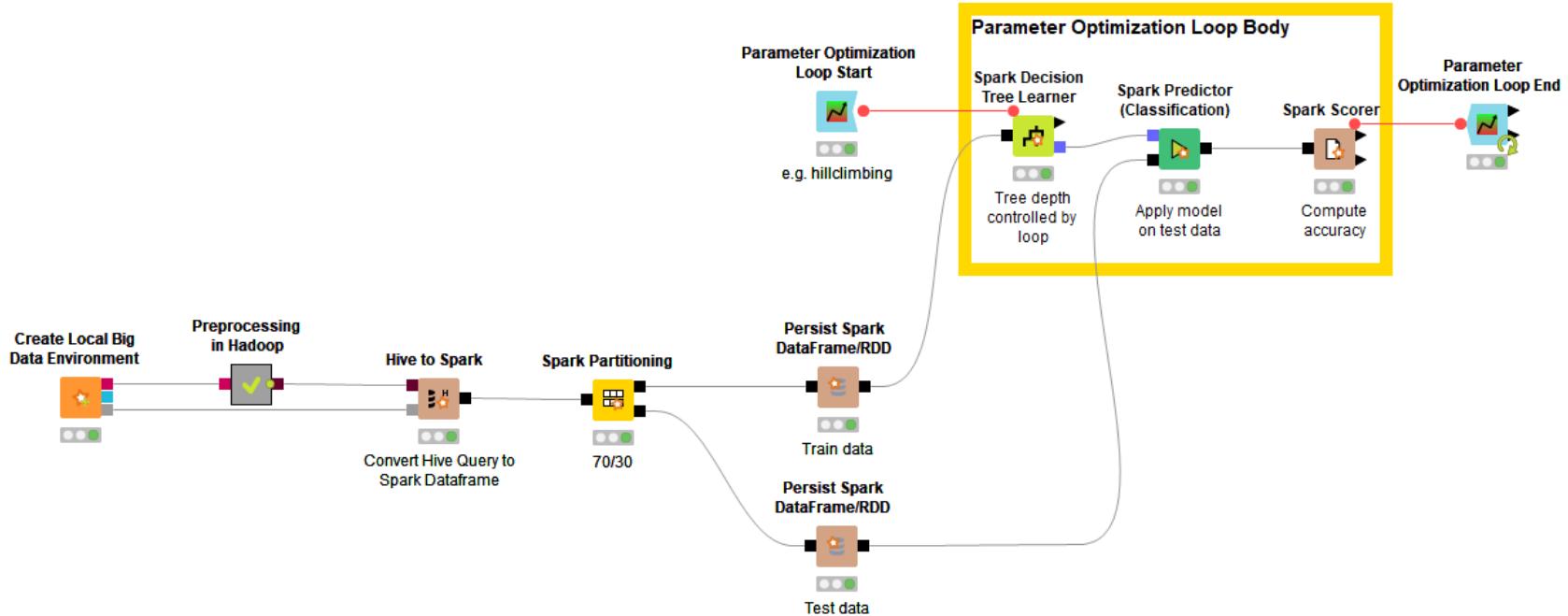


Closing the Loop

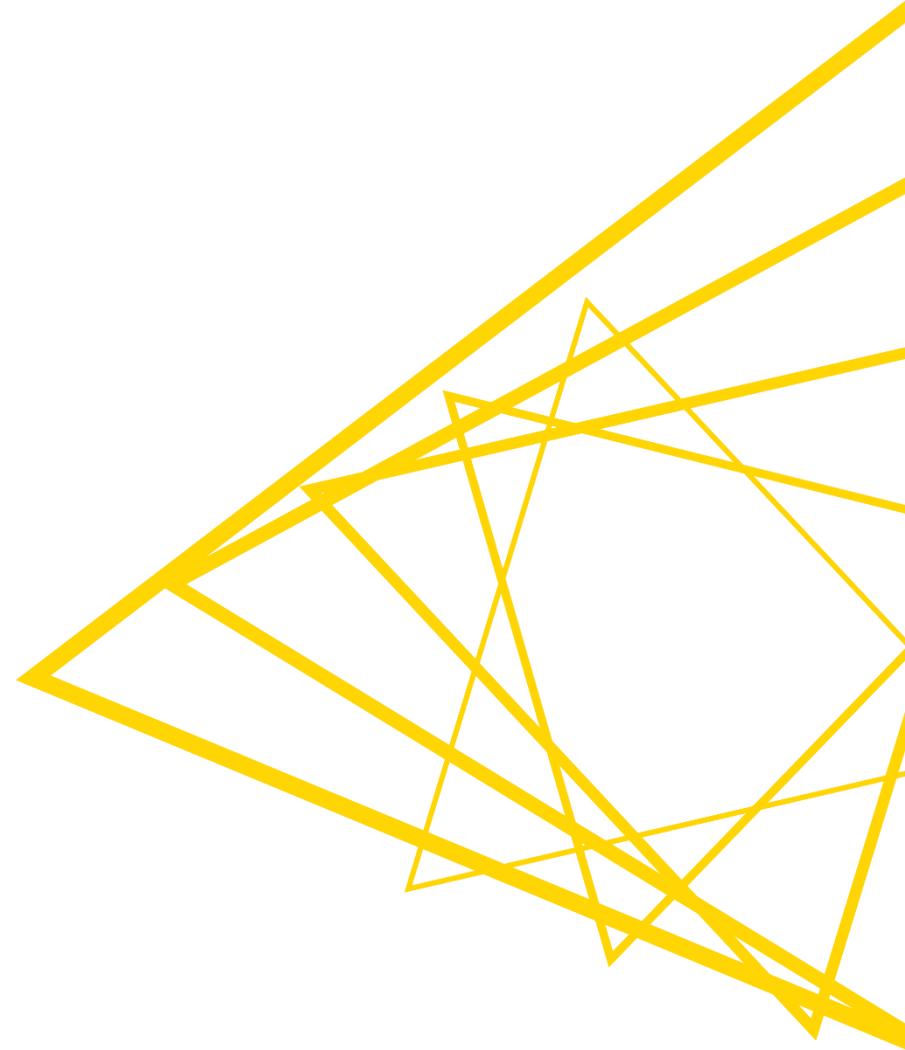


Mix and Match

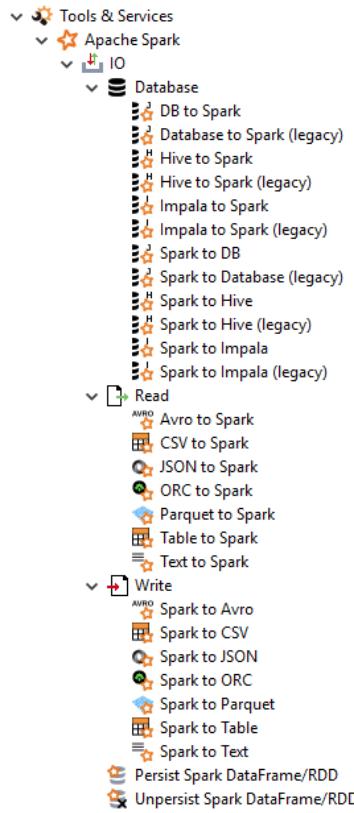
- KNIME <-> Hive <-> Spark



Export Data back into KNIME/Hadoop



Export Data from Spark



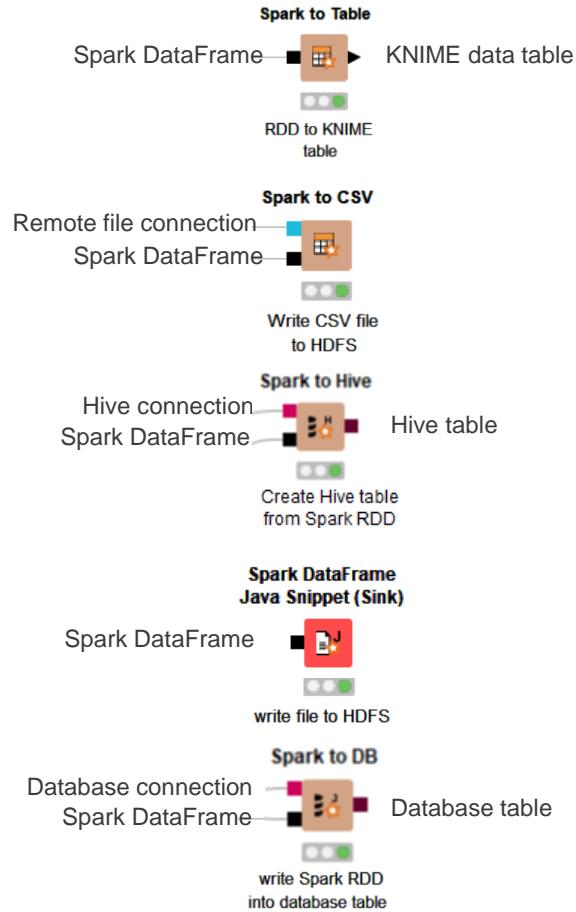
■ To KNIME

■ To CSV file in HDFS

■ To Hive

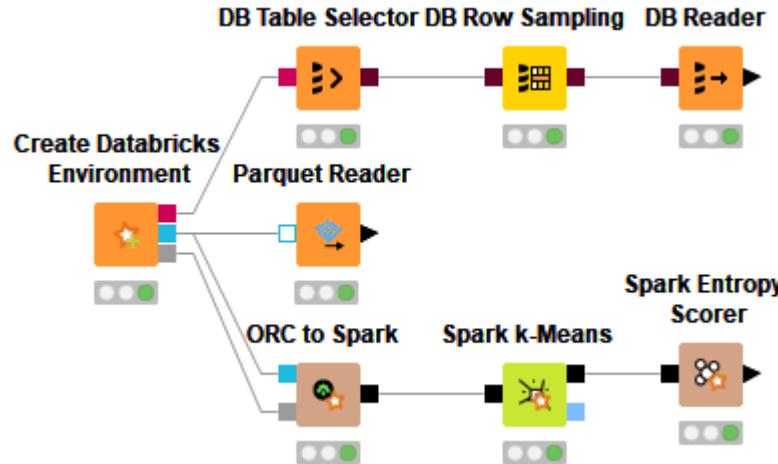
■ To Other Storages

■ To Database



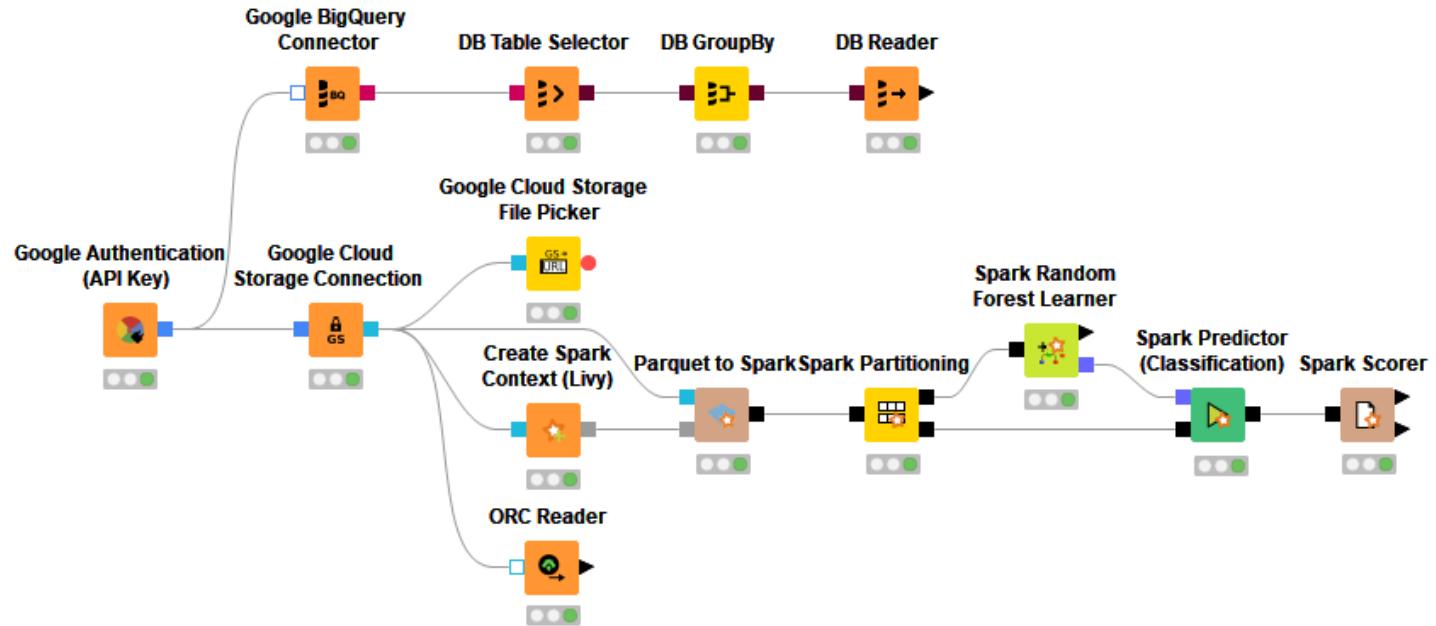
Cloud & Big Data Connectivity: Databricks

- Create Databricks Environment: connect to your Databricks cluster
 - Azure or AWS
- Databricks Delta, Databricks File System, or Apache Spark



Cloud & Big Data Connectivity: Google

- Connectivity to
 - Google Cloud Storage
 - Google Big Query (via DB Nodes)
 - Google Cloud Dataproc



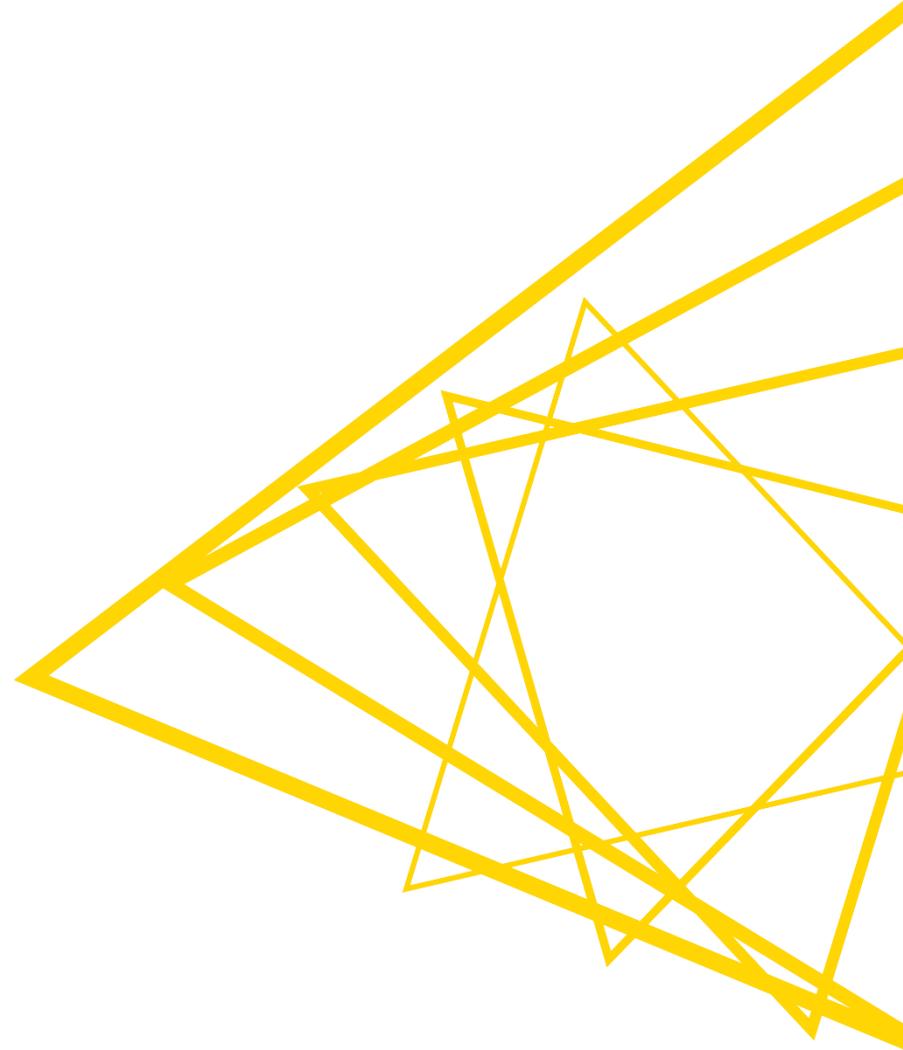
Section Exercise – 04_Spark_WritingToDB

This workflow provides a Spark predictor to predict COW values for the ss13pme data set. The model is applied to predict COW values where they are missing.

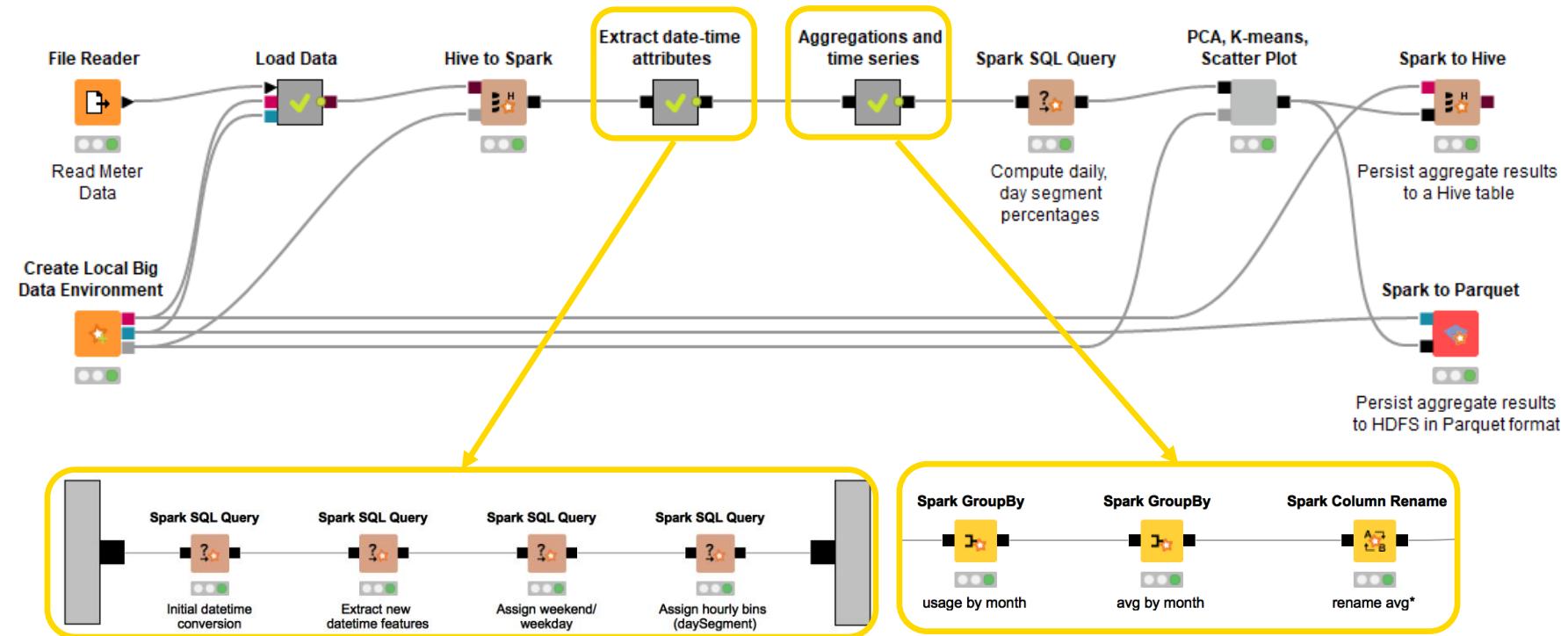
Now export the new data set without missing values to:

- A KNIME table
- A Parquet file in HDFS
- A Hive table

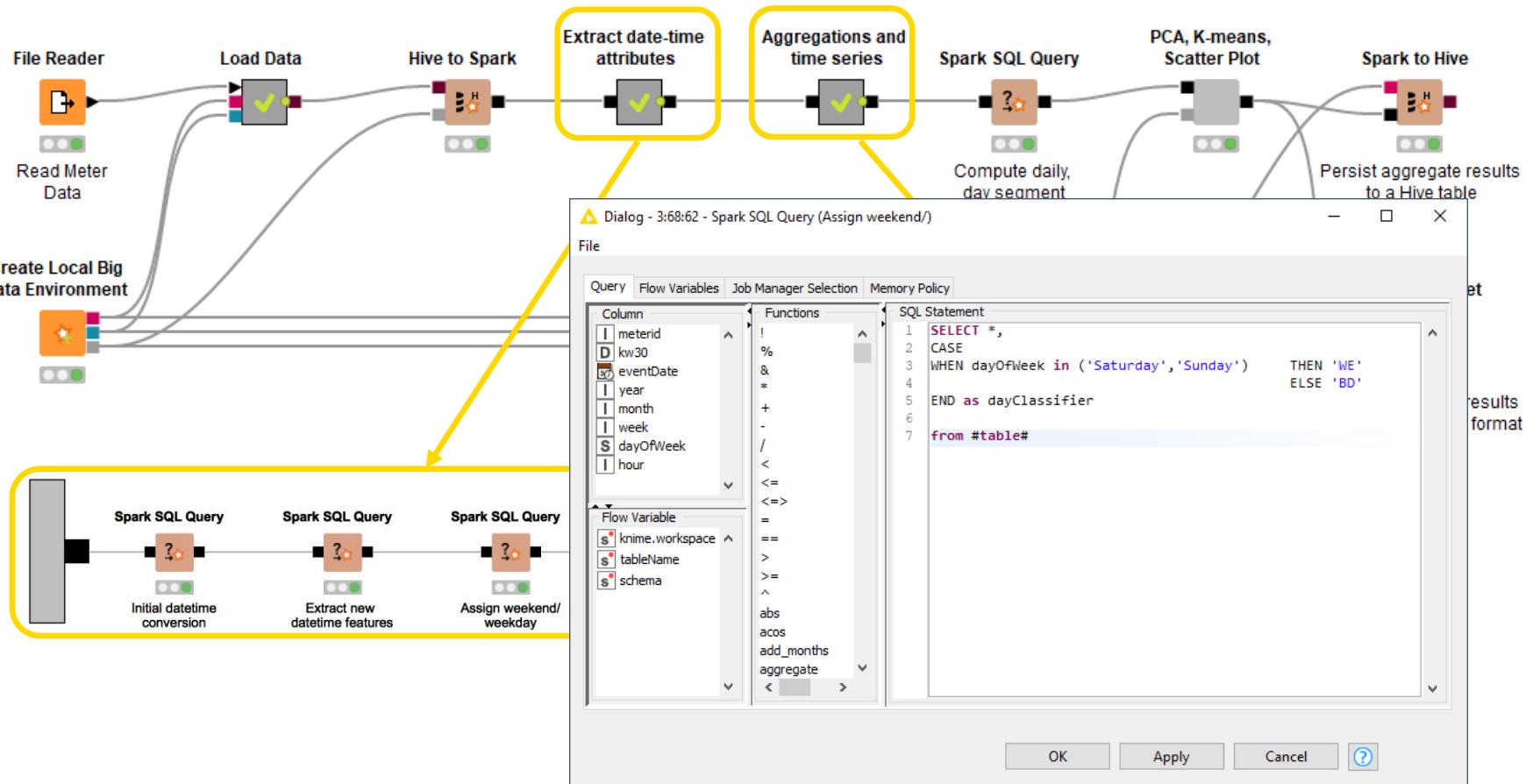
Examples



Analyzing the Irish Meter Dataset Using Spark SQL



Analyzing the Irish Meter Dataset Using Spark SQL



Columnar File Formats

- Available in KNIME Analytics Platform: ORC and Parquet
- 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.
- Improves performance when Hive is reading, writing, and processing data in HDFS

Example: Columnar File Formats

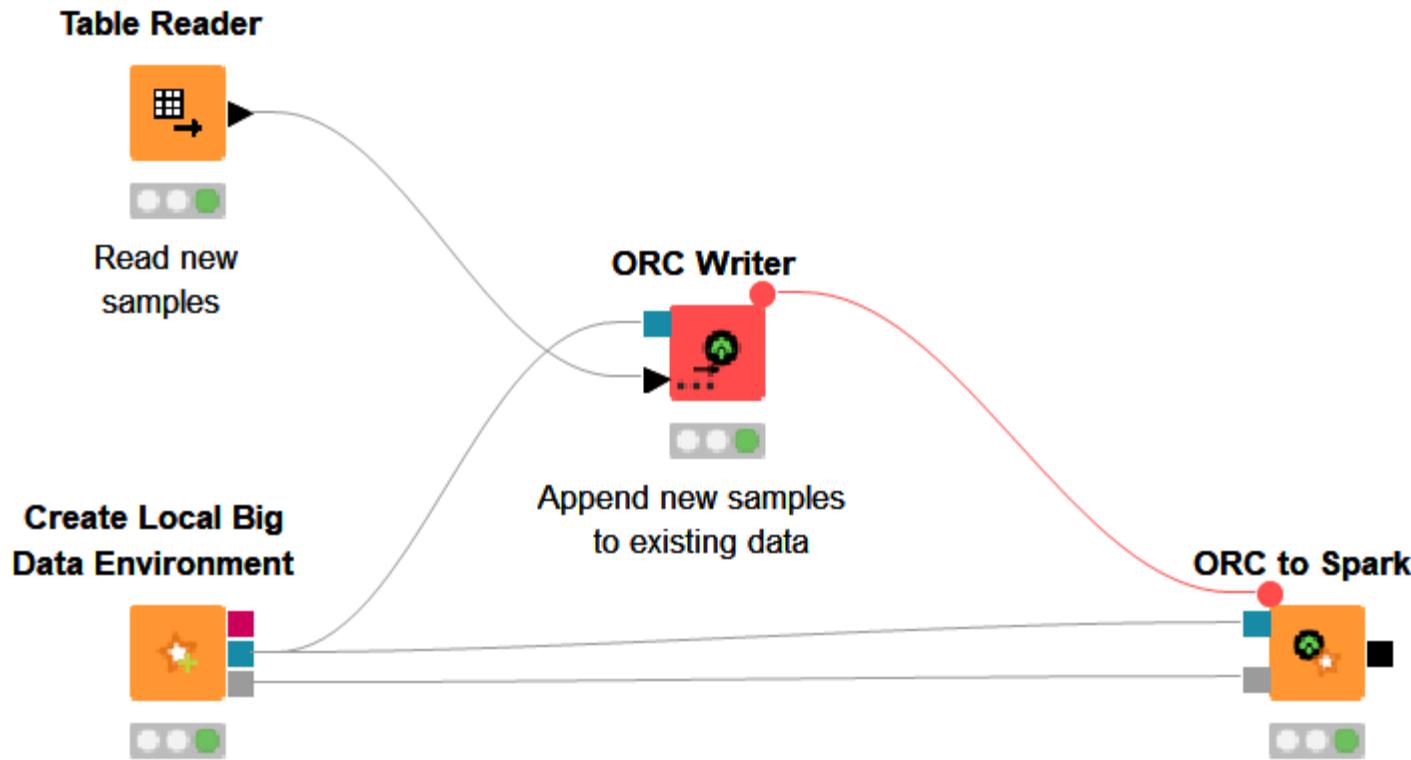
ID	Gender	Age
1	female	45
2	male	20
:	:	:
3333	male	42
Metainformation ID: min = 1 ; max = 3333 Gender: female; male Age: min = 20; max = 45		
ID	Gender	Age
3334	female	45
2	male	20
:	:	:
6666	male	42
Metainformation ID: min = 3334 ; max = 6666 Gender: female; male Age: min = 5; max = 24		
ID	Gender	Age
6667	female	45
2	male	20
:	:	:
10000	male	42
Metainformation ID: min = 6667 ; max = 10000 Gender: female; male Age: min = 45; max = 87		

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	5
2	male	24
:	:	:
6666	male	17
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 = 2 ; max = 10000 Gender: male Age: min = 45; max = 87		

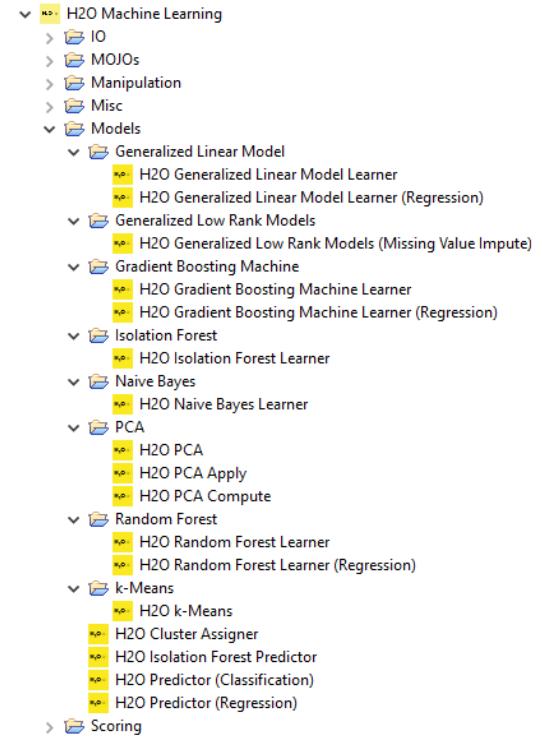
Select ID from table where
Age > 30 and Gender = female

Example: Write an ORC File

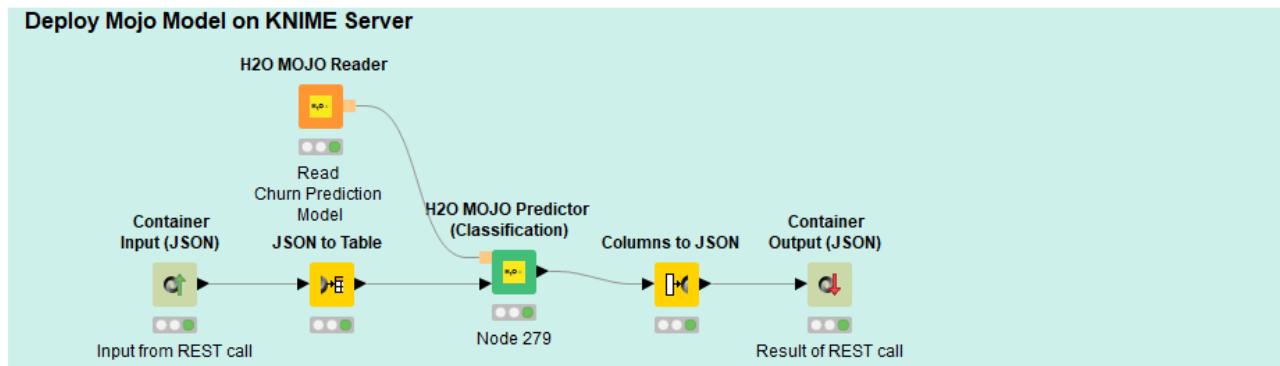
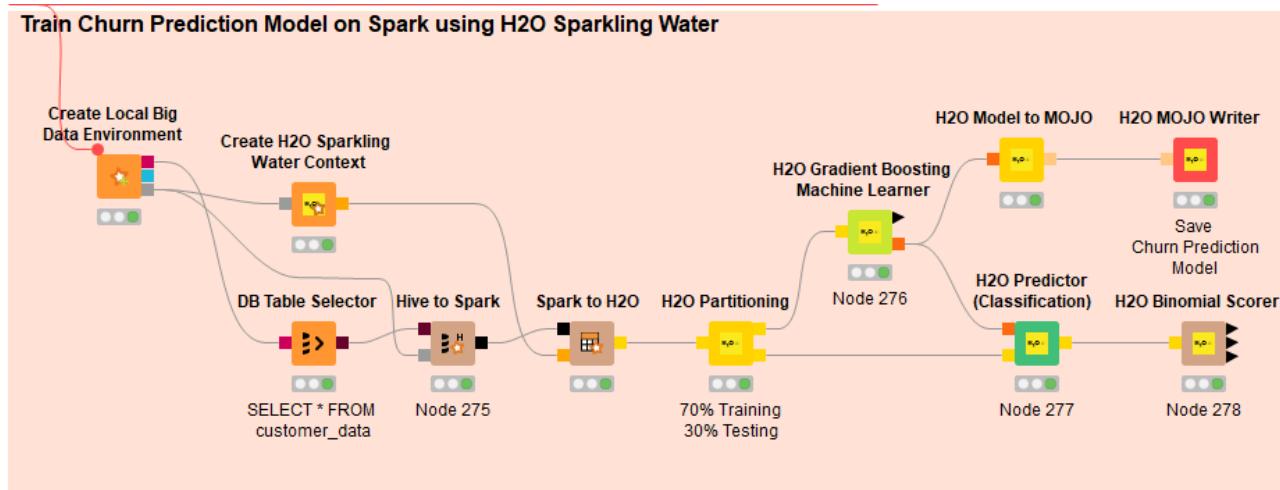


H2O Integration

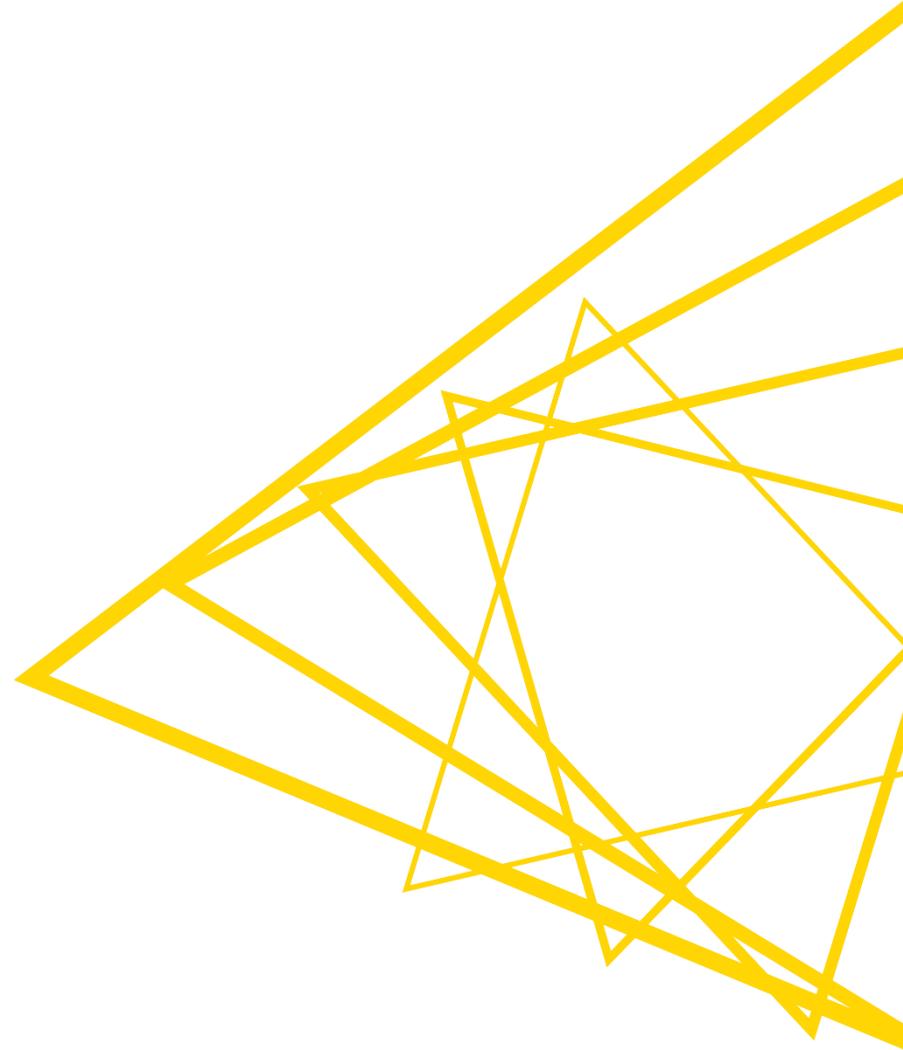
- KNIME integrates the H2O machine learning library
- H2O: Open source, focus on scalability and performance
- Supports many different models
 - Generalized Linear Model
 - Gradient Boosting Machine
 - Random Forest
 - k-Means, PCA, Naive Bayes, etc. and more to come!
- Includes support for MOJO model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



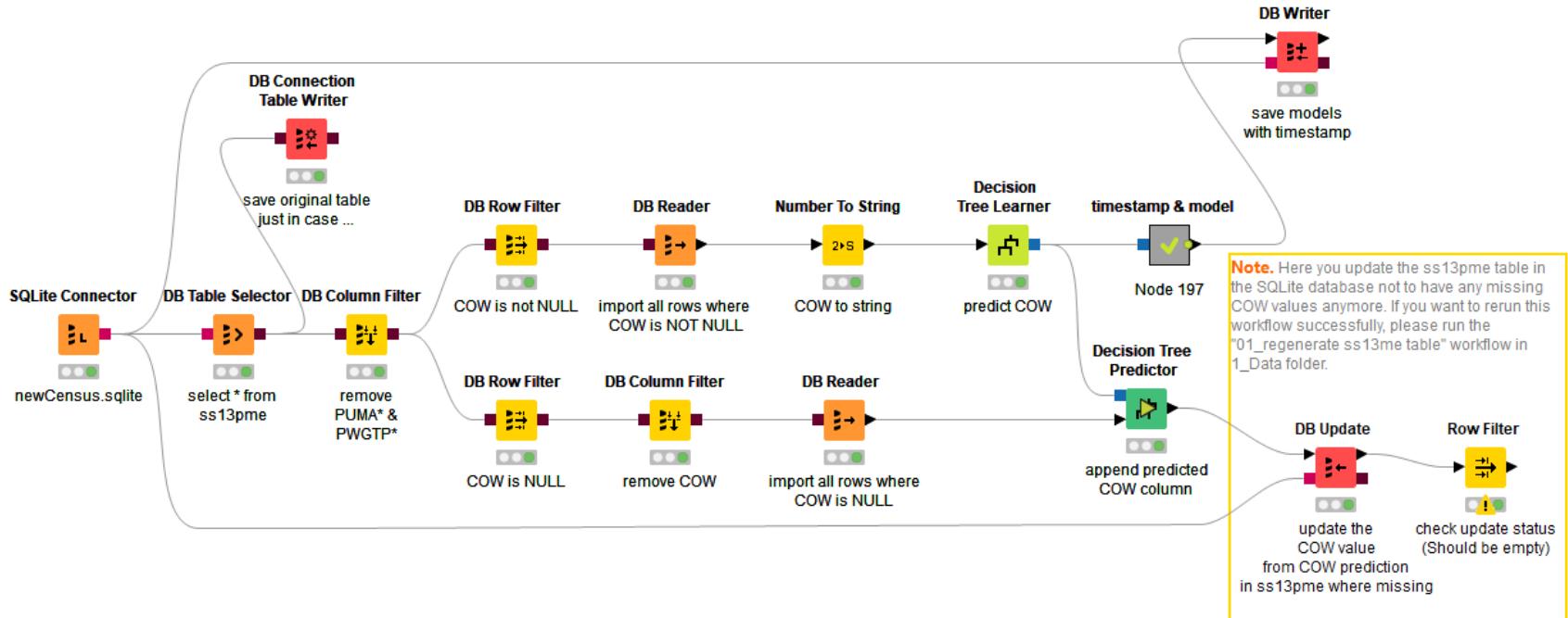
The H2O Sparkling Water Integration



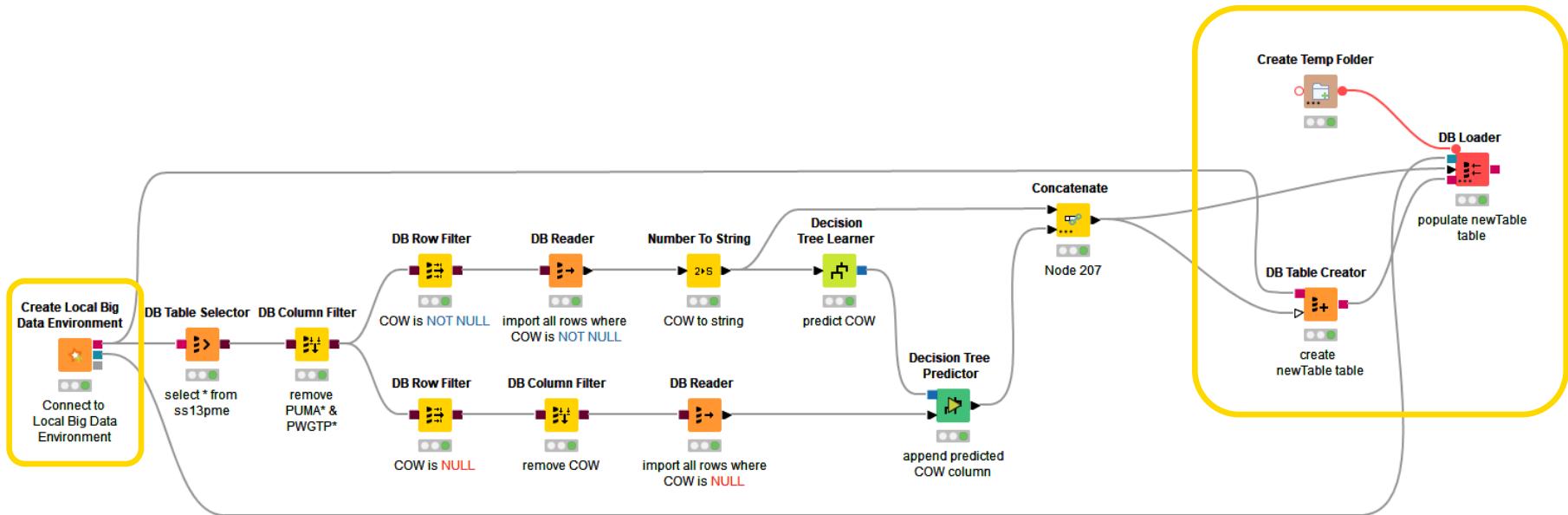
Conclusions



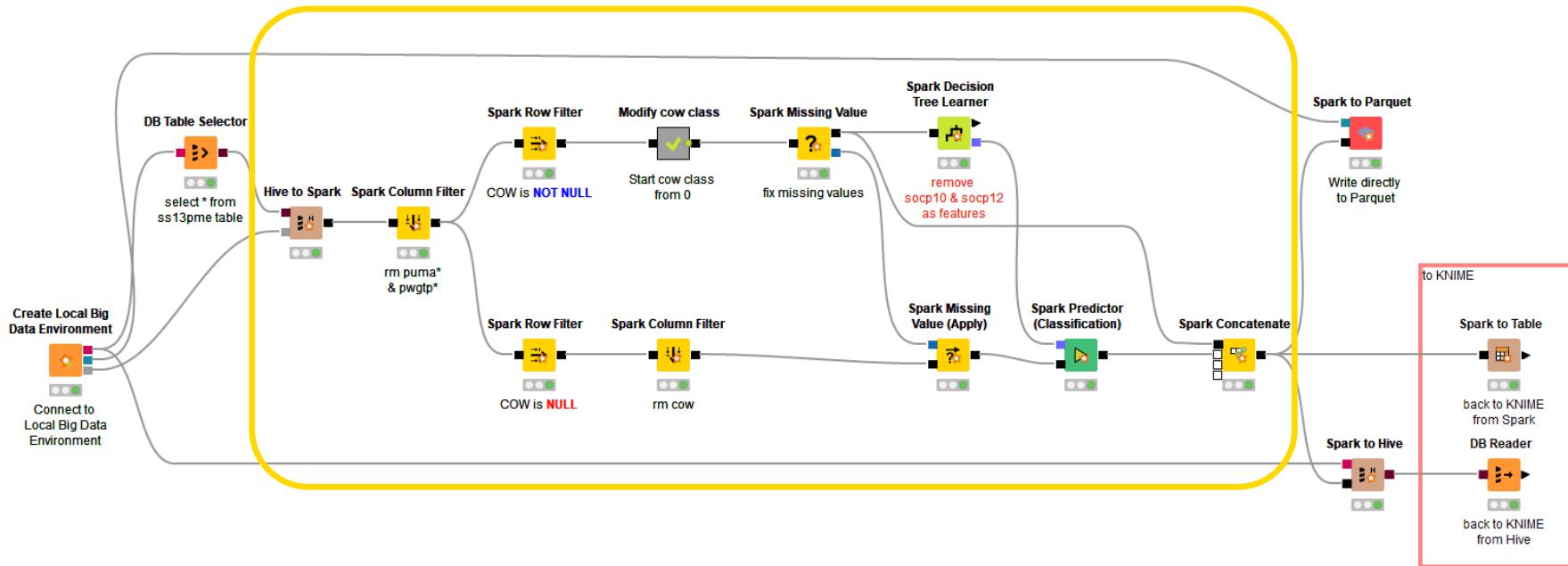
SQLite



Hadoop Hive

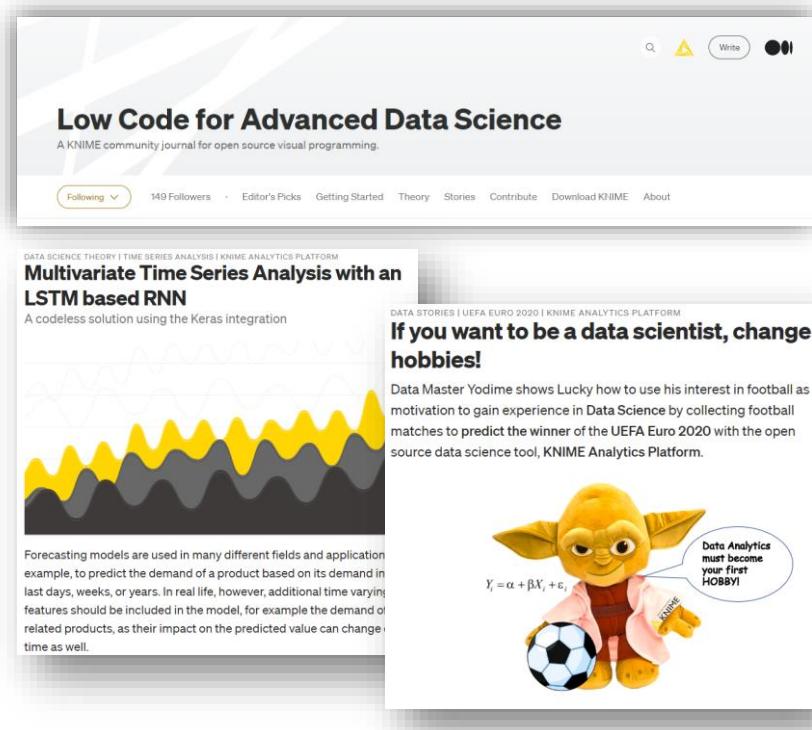


Spark



Stay Up-To-Date and Contribute

- Follow the KNIME Community Journal on Medium
[Low Code for Advanced Data Science](#)
- Daily content on data stories, data science theory, getting started with KNIME and more for the community by the community
- Would you like to share your data story with the KNIME community?



[Contributions](#) are always welcome!



The End

education@knime.com
Twitter: @KNIME

