

Teradata KNIME Components

Release 2.0.0

March 2024

Copyright © 2024 Teradata

Component Information

Plugin name	Teradata
Author	Teradata
Version	2.0.0
License	https://www.knime.com/terms-forum-hub
Components on KNIME Hub	https://hub.knime.com/teradata%20team/spaces/Teradata/~aQ-yIBPDOq9mudEL/
Reporting issues	https://support.teradata.com

Software and Compatibility Information

KNIME Components for Teradata Version	Date	Compatible Vantage Version	Compatible Vantage Platforms	Compatible Operating Systems
1.0	October 2023	1.0 or later (SQL Engine 16.20 or later)	All platforms that support the compatible VantageCore and VantageCloud Enterprise versions	SLES11 SP3 SLES12 SP3
2.0	March 2024	1.0 or later (SQL Engine 16.20 or later)	All platforms that support the compatible VantageCore and VantageCloud Enterprise versions	SLES11 SP3 SLES12 SP3

Short Description

The Teradata KNIME Components enhances KNIME's built-in interaction capabilities with the Teradata Vantage™ Data and Analytics Platform. The plugin provides visual components for scaled, in-Database Analytics with data that you keep in the Teradata Vantage Analytics Database.

Long Description

With these components, you will be able to:

- Access and execute analytic functions that reside in the Teradata Vantage

Analytics Database. The functions are comprised by the following sets:

- A select subset of the Analytics Database analytic functions.
- The entire Vantage Analytics Library (VAL) functions set.
- The Unbounded Array Framework (UAF) time series functions.
- Run R and Python scripts natively in the Analytics Database for scaled and performant in-Vantage execution.
 - Execute APPLY Table Operator on VantageCloud Lake systems.
 - Execute SCRIPT Table Operator on VantageCloud Enterprise and Vantage Core systems.
- Run scaled scoring tasks in the Analytics Database via the Bring Your Own Model(BYOM) Analytics Database software with models you have previously stored in Analytics Database tables. The present node supports scoring with models in PMML, ONNX, native Dataiku, DataRobot and H2O MOJO formats.
- (For administrators) Collect telemetry information about the KNIME-based session on a target Vantage system.

Note that to use the plugin:

- The present version of the plugin supports VantageCloud Enterprise, Vantage Core systems and Vantage Cloud Lake systems.
- You will need Teradata Vantage access credentials to establish a connection to a target Analytics Database.
- To connect to an Analytics Database, the Teradata JDBC Driver 16.20 or later is required. First, the driver can be downloaded from the website:
<https://downloads.teradata.com/download/connectivity/jdbc-driver>
- To use the "In-Vantage Scripting" nodes
 - a. with VantageCloud Lake systems:
 - Your target system must be equipped with a compute cluster component which has at least one analytic compute group so the Open Analytics Framework can be used. Furthermore, your Analytics Database Administrator must grant your Analytics Database *user* account with access privileges to an analytic compute group.
 - If the analytic compute group name you wish to use and have access to on your target system is not set by default, it can be specified in your workflow using a 'DB SQL Executor' node, after establishing a database connection. The statement to run after the is created connection should be specified in this node as:

SET SESSION COMPUTE GROUP *group_name*;

where *group_name* is the name of the analytic compute group you have access to on your target system.

b. with VantageCloud Enterprise or Vantage Core systems:

- the corresponding language bundles need to be installed directly on each node of the target Analytics Database per the following table:

PID	Product name
9687-2000-0120	R Interpreter and Add-on Pkg on Analytics Database
9687-2000-0122	Python Interpreter and Add-on Pkg on Analytics Database

- To use the "BYOM Scoring" recipe, the Teradata Vantage BYOM software package must be installed on the target Analytics Database, and you need to have execute privilege on the installation database; see the related BYOM documentation at the following link:

https://docs.teradata.com/r/Enterprise_IntelliFlex_Lake_VMware/Teradata-VantageTM-Bring-Your-Own-Model-User-Guide

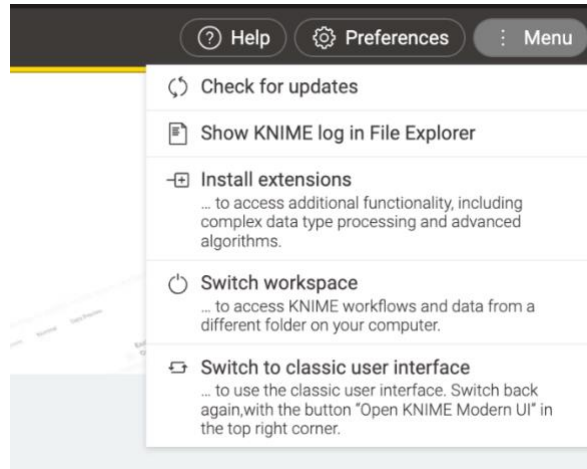
In the Teradata Vantage Analytics Database, the default installation location of the BYOM software is a database called *mldb*. In line with the preceding, your Analytics Database Administrator needs to grant your Database *user* account in advance the privilege

GRANT EXECUTE FUNCTION ON *mldb* TO *user* WITH GRANT OPTION;

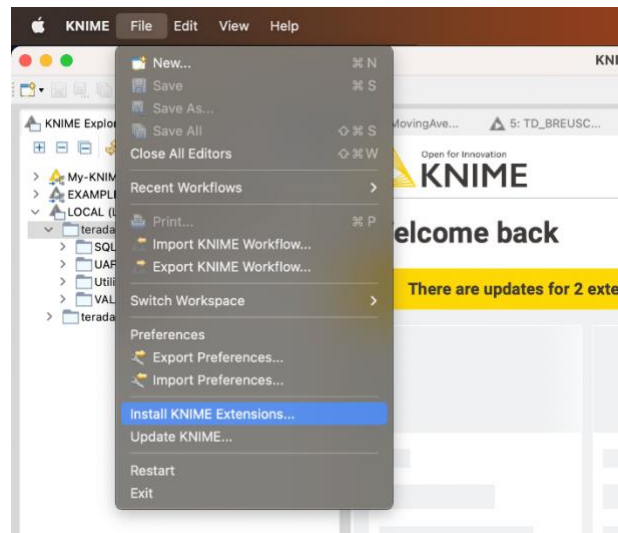
on *mldb*. If BYOM is installed in a different database on your system, then your Analytics Database Administrator needs to grant you the above privilege on the corresponding database, instead.

Install Dependencies

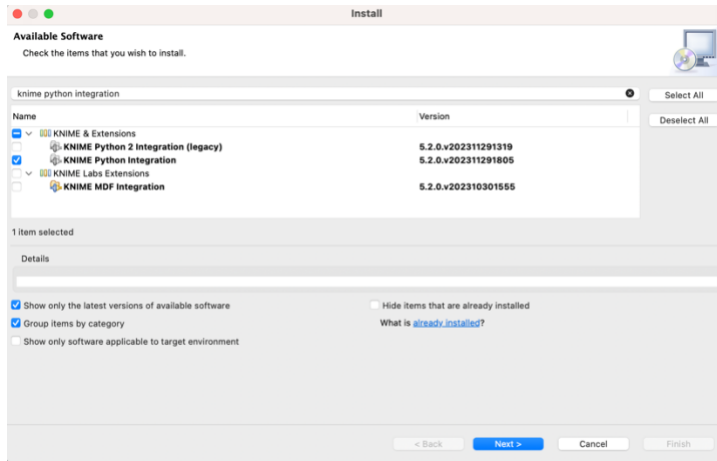
Make sure to be in “Classic UI”, if you are in “Modern UI”, switch to classic by clicking Menu in top right and click “Switch to classic user interface”.



Click File → “Install KNIME Extensions...”



Type “KNIME Python Integration”, choose the option under “KNIME & Extensions” and click through and finally “Finish”.

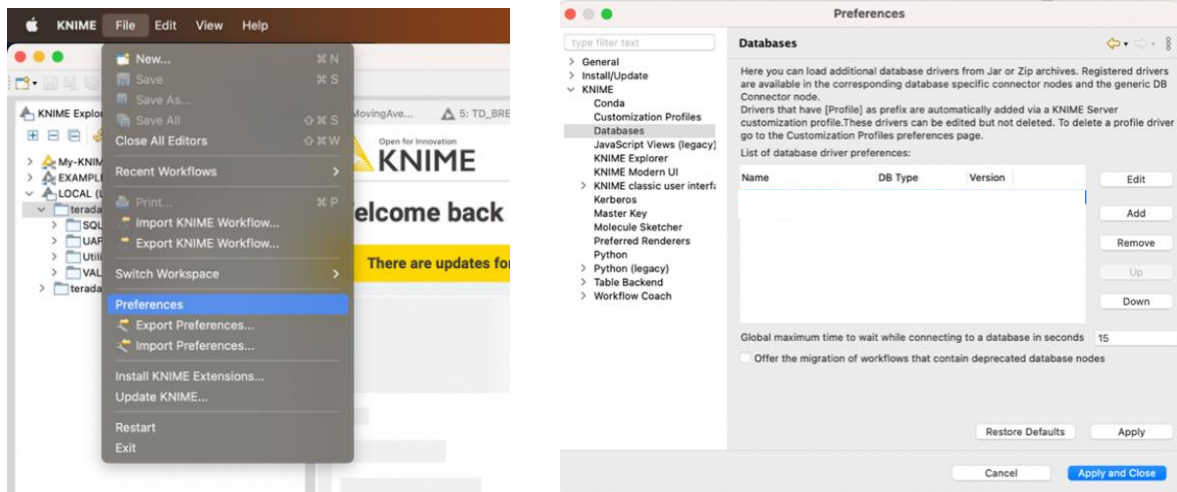


Finally, if KNIME asks you to “Restart Now”, then click “Restart Now” for the software update.

Register Teradata Database Driver

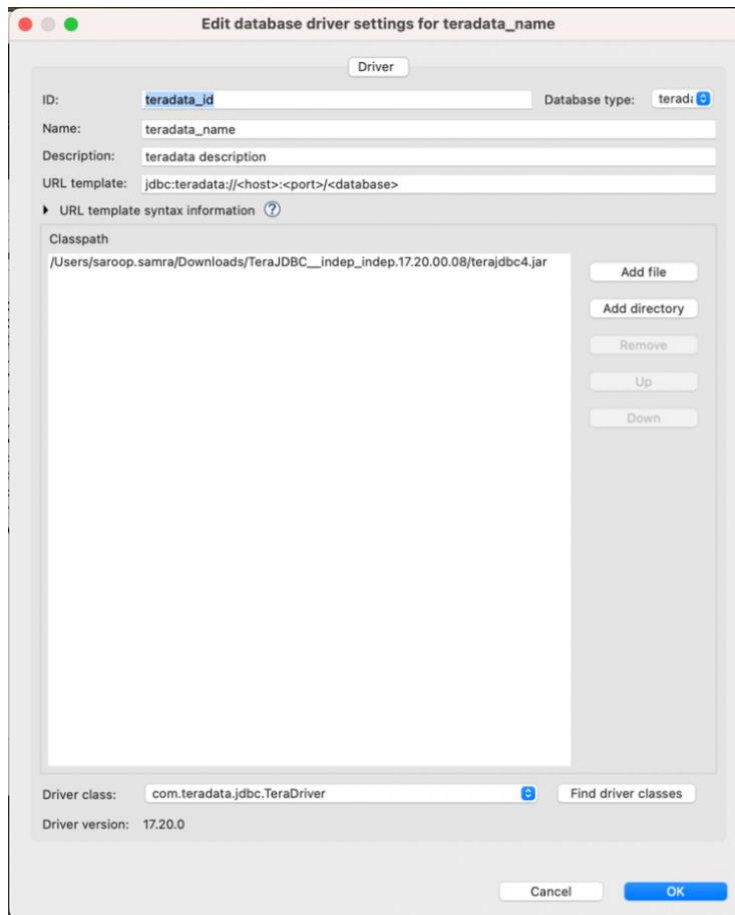
These instructions are for version KNIME 5.2 and above. If you are using 5.1, the only difference is that instead of selecting “teradata” in the Database Type, select “default”

1. Click on File → “Preferences”. Then go to “KNIME” → “Databases”. Click “Add”.



2. Type information for ID, Name, Description. In screenshot below, we use “teradata” for all three. If using KNIME 5.2+ then use choose “teradata” as Database Type.

3. Click “Add File” to direct to the filepath to where the Teradata JDBC is downloaded on the local machine.

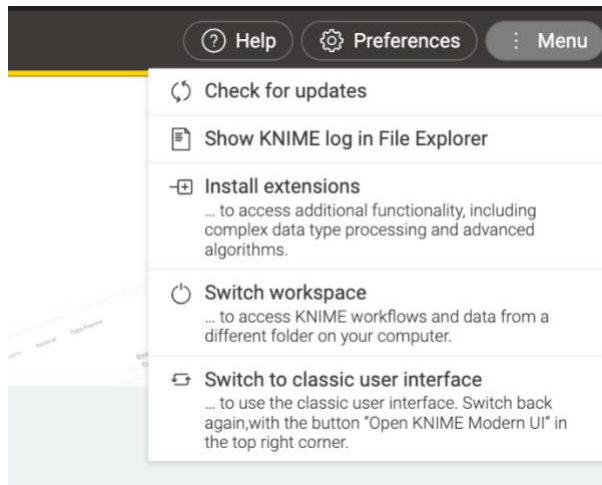


4. In URL Template replace “<protocol>” with the ID name chosen, (in this case “teradata” so it would be jdbc:teradata://<host>:<port>/<database>)
5. Click “Find driver classes” so that Driver class gets populated. Finally click “OK”.

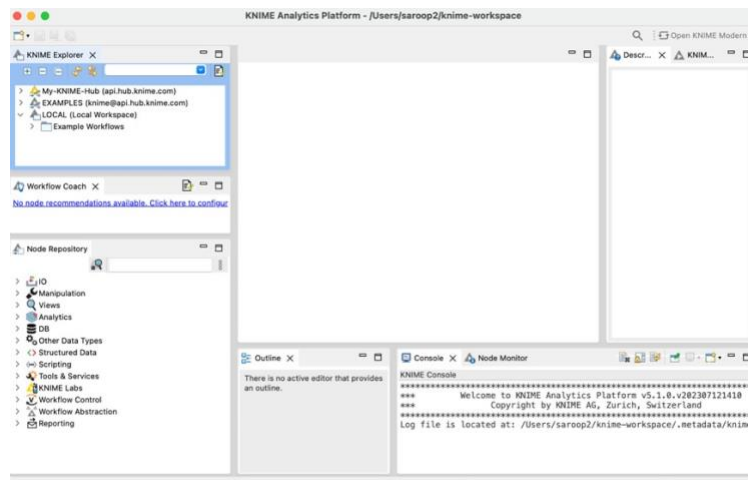
Installing Teradata Components

NOTE: to get the Teradata components (i.e. drag and drop into the workflow), KNIME needs to be in classic user interface style.

Make sure to be in “Classic UI”, if you are in “Modern UI”, switch to classic by clicking Menu in top right and click “Switch to classic user interface”.

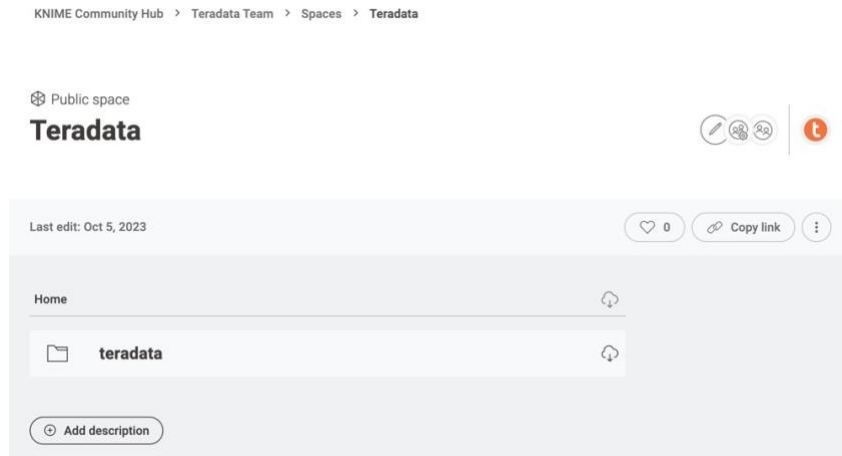


This is KNIME Classic UI. In order to get Teradata components, go to KNIME Community Hub to Teradata’s page (<https://hub.knime.com/teradata%20team>).

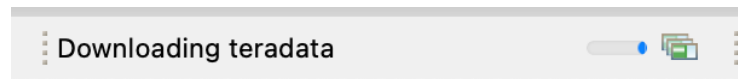
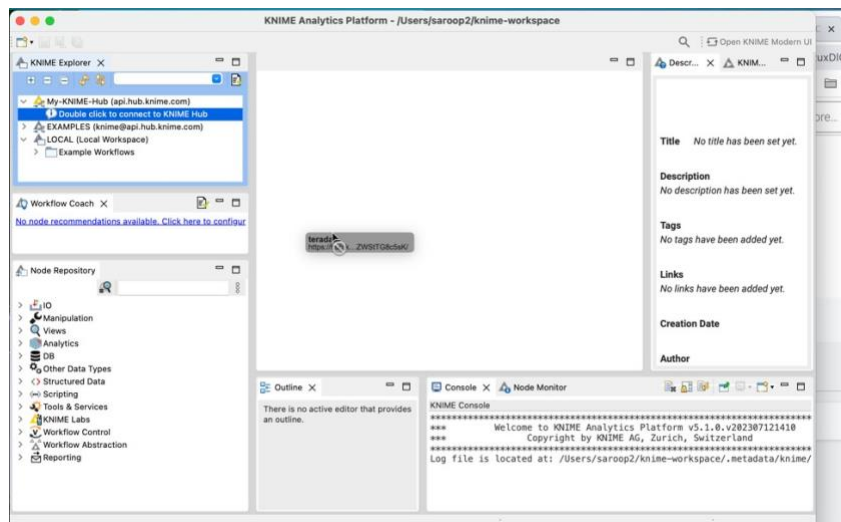


Go to the KNIME Hub link for the Teradata Components (<https://hub.knime.com/teradata%20team/spaces/Teradata/~aQ-yIBPDOq9mudEL/>)

Teradata KNIME Components, Release 2.0

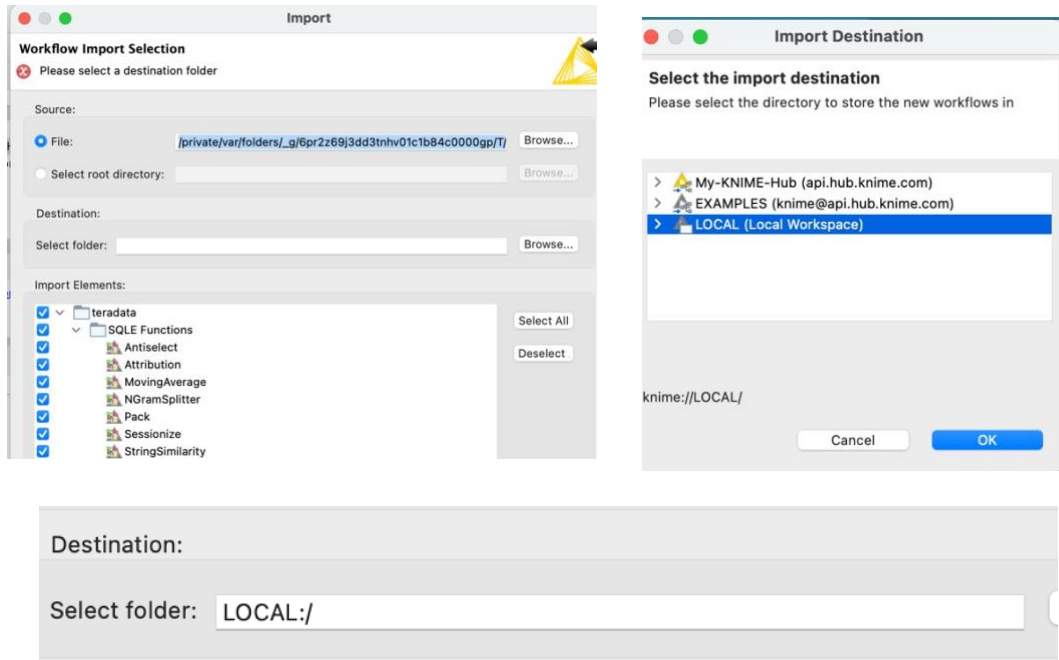


1. Drag the “teradata” folder from the webpage directly into the KNIME application into your workspace.

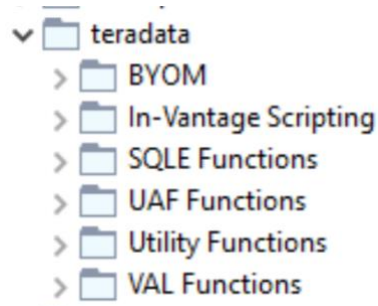


NOTE: This may take some time, do not exit out, the bottom right will show it is downloading.

2. A pop-up window will open for the “Workflow Import Selection.” Choose the destination folder by clicking “Browse” and selecting a folder under Destination section. In our example we will put into “LOCAL (Local Workspace).” Click “OK” then after click “Finish”.









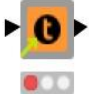
The Teradata folder is now available with all of the analytic functions, BYOM scoring, In-Vantage Scripting and utility functions.



Teradata Utility Functions

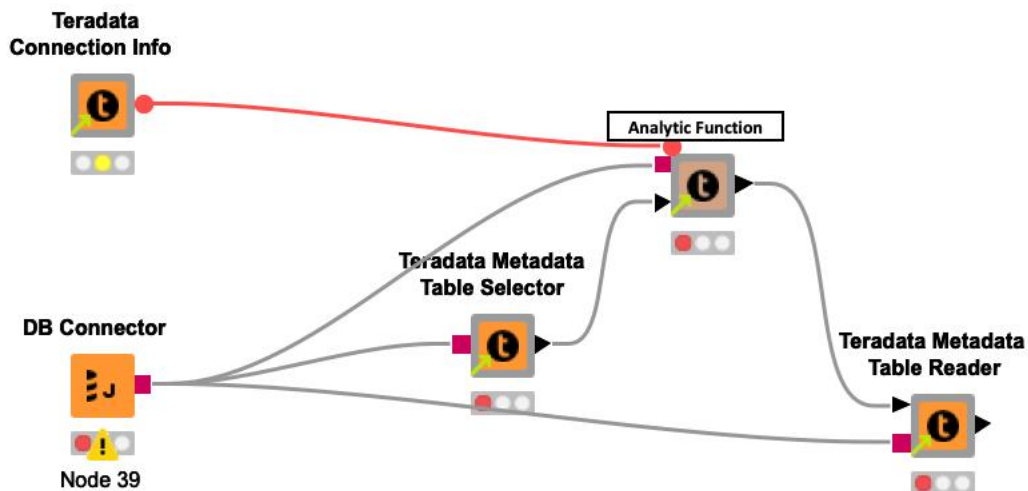
The following is a summary of the Teradata utility components.

Utility Function Node	Description
<p data-bbox="266 512 418 558">Teradata Connection Info</p> 	<p data-bbox="516 464 1487 533">Set up the “VAL Location” and the “Default Database”. This goes as a flow variable input to the <i>analytic function</i>.</p>
<p data-bbox="266 770 418 816">Teradata Data Table Partition</p> 	<p data-bbox="516 722 1422 905">Use if Partition Type is desired for the function. Choose “PARTITION BY”, “HASH BY”, “PARTITION BY ANY”, or “DIMENSION for the Partition Type. There are “Partition By” and “Order By” placeholders for choice to partition by or order by column(s).</p>
<p data-bbox="266 1037 418 1083">Teradata Metadata Table Reader</p> 	<p data-bbox="516 989 1451 1094">Used to read the metadata table as a KNIME table, which can also be seen in Node Monitor. This node will query the database and download the table as a KNIME table.</p>
<p data-bbox="290 1274 418 1320">Teradata Empty Table</p> 	<p data-bbox="516 1226 1463 1331">Several functions have the option for multiple inputs, but not all inputs are required. If you do not want to set a table for the optional table, use Teradata Empty Table as a placeholder input.</p>
<p data-bbox="266 1541 418 1587">Teradata Metadata Table Selector</p> 	<p data-bbox="516 1493 1495 1598">This is the node for selecting the “Schema” and the “Table” name. It is not the input table itself but rather the metadata of the table that feeds into the <i>analytic function</i>.</p>

<p>Teradata UAF Art</p>  <p>The icon for the Teradata UAF Art node, featuring a central 't' logo with a yellow background, flanked by black arrows, and a red status indicator below.</p>	<p>For UAF functions that are classified as ART, use this node to set up the parameters that will go into the function.</p>
<p>Teradata UAF Matrix</p>  <p>The icon for the Teradata UAF Matrix node, featuring a central 't' logo with a yellow background, flanked by black arrows, and a red status indicator below.</p>	<p>For UAF functions that are classified as Matrix, use this node to set up the parameters that will go into the function.</p>
<p>Teradata UAF Series</p>  <p>The icon for the Teradata UAF Series node, featuring a central 't' logo with a yellow background, flanked by black arrows, and a red status indicator below.</p>	<p>For UAF functions that are classified as Series, use this node to set up the parameters that will go into the function.</p>

Basic Node Setup

To get a function to run, a flow structure is needed. The details of setting up each of these nodes are in the following sections.



Overall, there needs to be a DB Connector, this is a KNIME node that is used to connect to Teradata connection. This goes into the red square ports of the Teradata Metadata Table Selector, the *analytic function* of choice, and the Teradata Metadata Table Reader.

There is an additional node, Teradata Connection Info, which is used if one needs to specify the Val Location or Default Database. This goes as an input variable on top of the *analytic function* (drag and drop connection in general area above the node).

The Teradata Metadata Table Selector is the node for choosing the Table name and Schema. This node is the metadata of the table selected. This goes in as the black triangle input to the *analytic function* of choice.

The *analytic* can be a SQLE, VAL, or UAF node*. This node has the parameters specific to this function inside. The analytic function will perform the query on the database and the node outputs a Teradata Metadata Table which further analytic functions can be applied to, or if you want to download the table to KNIME then it can be connected to a Teradata Metadata Table Reader, which reads the metadata and queries the database to generate a KNIME table.

*Note: UAF requires the use of either Teradata UAF Art, Teradata UAF Matrix, or Teradata UAF Series node after the Teradata Metadata Table Selector and before the function (see Workflow Example 3 below for example)

DB Connector Setup (KNIME 5.2 and above)

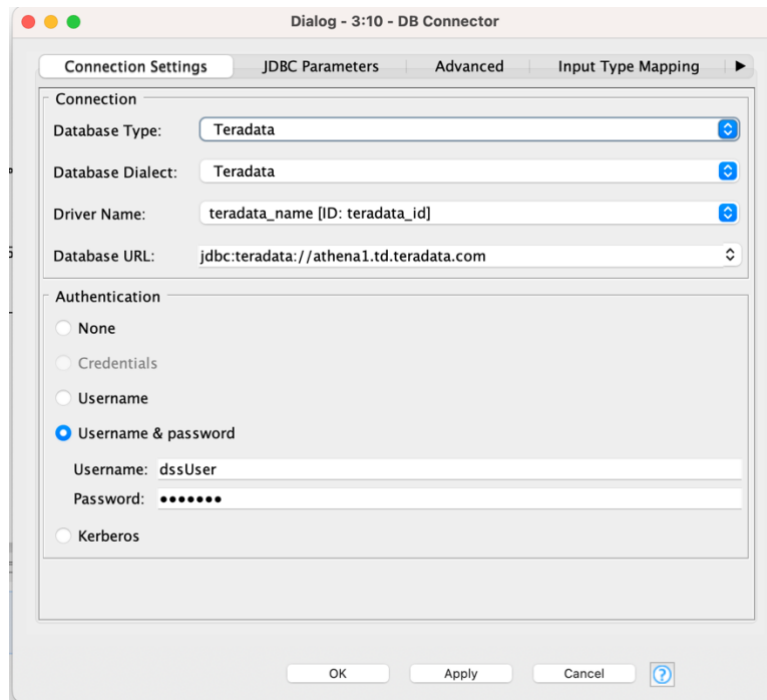
These instructions are for version KNIME 5.2 and above. If you are using 5.1, see Appendix for DB Connector Setup prior to version 5.2.

The UI for DB Connector will ask to specify the “Database Type”, choose “Teradata”.

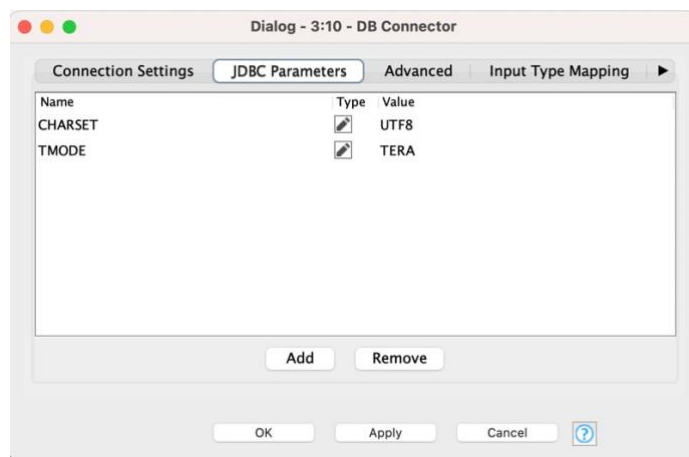
For “Database Dialect”, choose “Teradata”.

The “Driver Name” will be “teradata [ID: teradata]” (name based on what you specified for database driver step before).

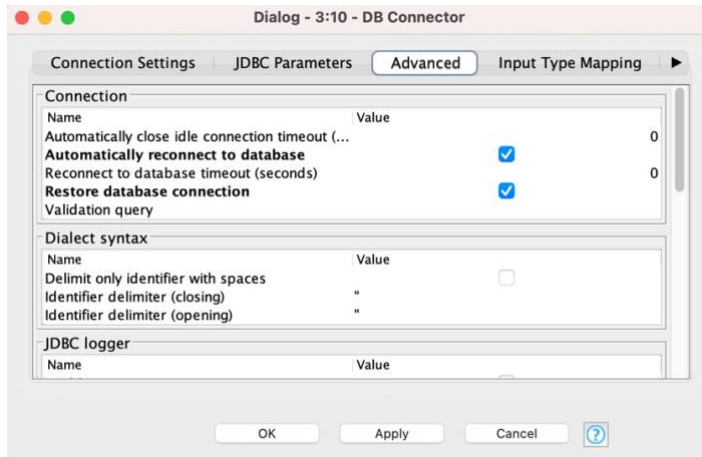
Enter the Database URL and the Username and Password for your Teradata system.



For the JDBC Parameters set “CHARSET” to the value “UTF8” and set “TMODE” to the value “TERA”.



In the Advanced tab, click the checkbox on for “Automatically reconnect to database” and “Restore database connection”, which will ensure that the database settings will be saved after exiting out of the application.



Teradata Metadata Table Selector Setup

The DB Connector connects to the function component as well as “Teradata Metadata Table Selector” component. The UI for the “Teradata Metadata Table Selector” has a place to specify the “Schema” and “Table” names.



Teradata Metadata Table Reader Setup

The “DB Connector” and the function component node are connected to the “Teradata Metadata Table Reader” node. There is no UI for this node as after executing you will see the result of the query in the “Node Monitor” after clicking the “Teradata Metadata Table Reader” node.

Node: df_reader_1 (3:37)
 State: EXECUTED

Port Output: Port 0 Rows: 25, C

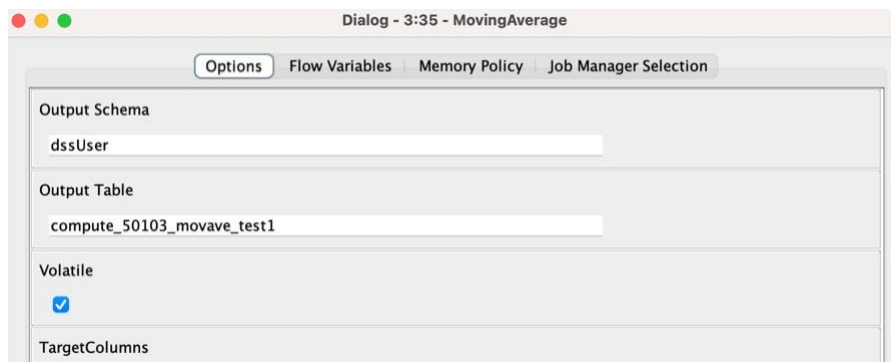
ID	id	name	period	stockprice	stockprice_mavg
Row0	11	Company1	1961-06-01T00:00	492.0	492.0
Row1	14	Company1	1961-06-06T00:00	497.0	497.0
Row2	6	Company1	1961-05-24T00:00	459.0	459.0
Row3	24	Company1	1961-06-20T00:00	478.0	478.0
Row4	12	Company1	1961-06-02T00:00	498.0	498.0
Row5	13	Company1	1961-06-05T00:00	499.0	499.0
Row6	16	Company1	1961-06-08T00:00	490.0	490.0
Row7	17	Company1	1961-06-09T00:00	489.0	489.0
Row8	15	Company1	1961-06-07T00:00	496.0	496.0
Row9	22	Company1	1961-06-16T00:00	482.0	482.0
Row10	7	Company1	1961-05-25T00:00	463.0	463.0
Row11	4	Company1	1961-05-22T00:00	459.0	459.0
Row12	19	Company1	1961-06-13T00:00	487.0	487.0
Row13	25	Company1	1961-06-21T00:00	479.0	479.0

Analytic Function Component Setup

The Function component will have the different parameters specific to that function, however each of them share the specification of the “Output Schema”, “Output Table” name you want to assign, and a checkbox of whether you want the table to be “Volatile” which by default is clicked on.

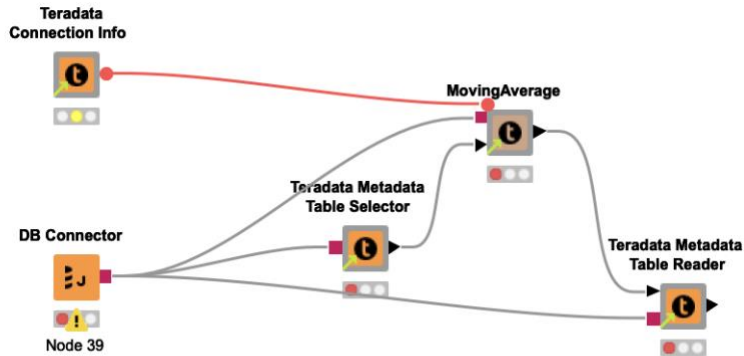
Note: The Output Table parameter is mandatory, but the output schema can be empty in which case the default database will be the schema that is used.

See the Moving Average Example for a step-by-step guide to how to setup a full KNIME flow with an analytic function.

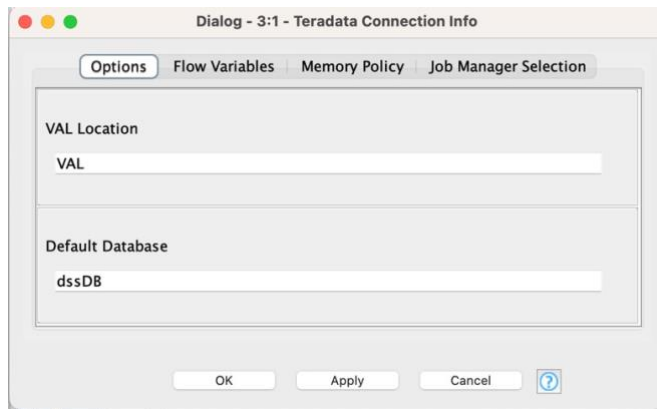


Workflow Example 1 – Moving Average

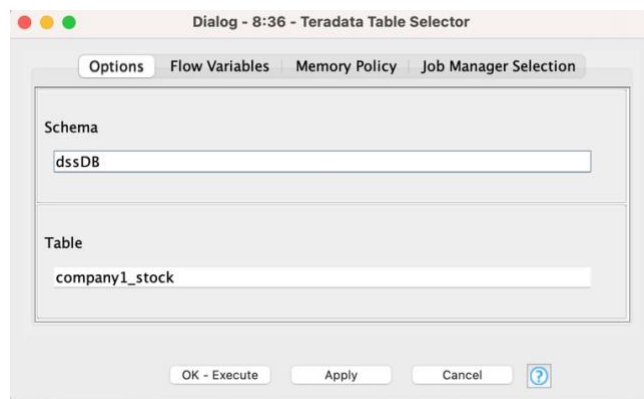
The first example is Moving Average, a SQLE function. The *analytic function* needs a DB Connector node, this is a KNIME node. The remaining nodes are Teradata specific.



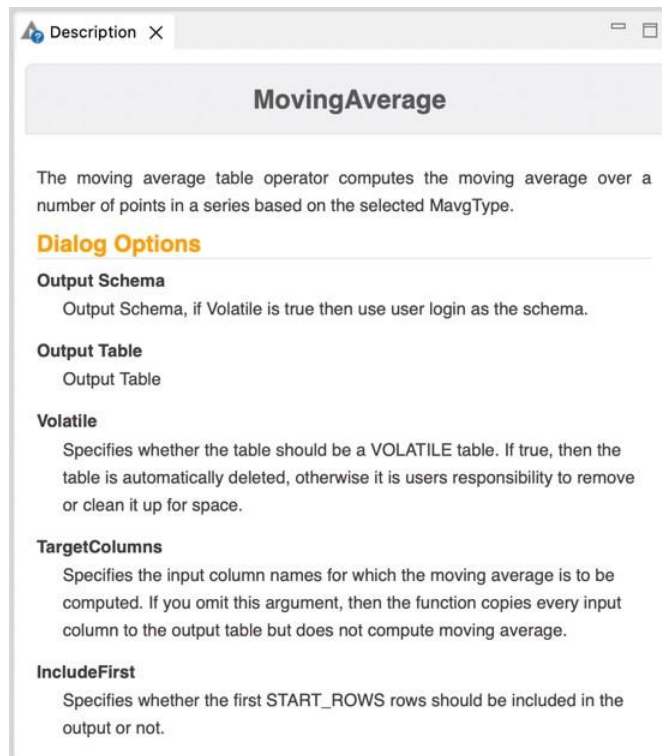
In this example, in the “Teradata Connection Info” node, we set VAL Location to “VAL” and Default Database to “dssDB”.



For the “Teradata Metadata Table Selector” node we set Schema as “dssDB” and Table to “company1_stock”.

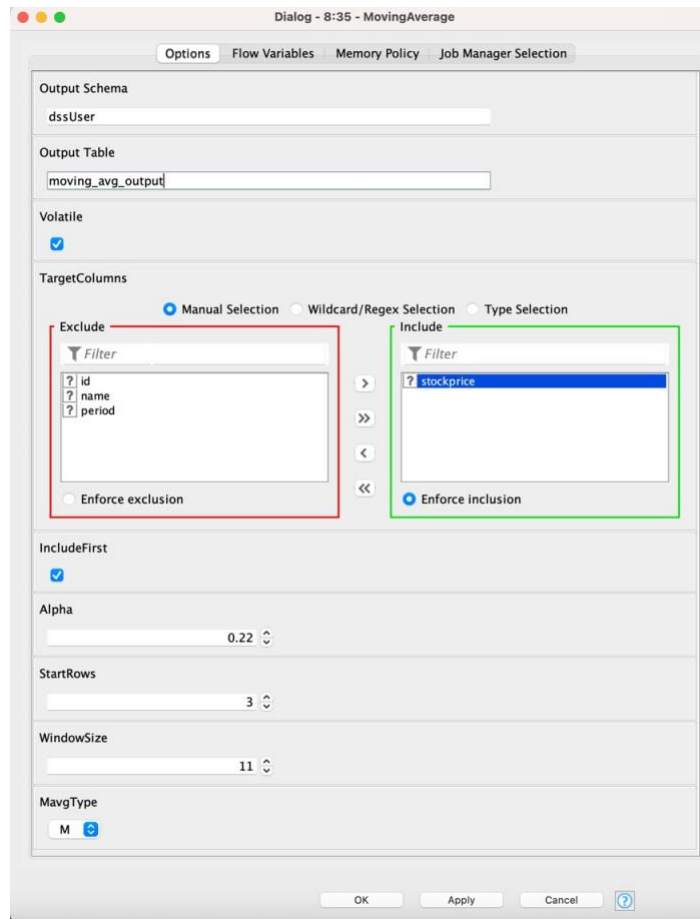


Each of the function nodes have a description of the parameters, which can be found on the right after clicking on the *analytic function*.



For the *analytic function* "MovingAverage" the example has Output Schema as "dssUser" and Output Table as "moving_avg_output". The parameters of the function are set as such: TargetColumns is set to "stockprice", IncludeFirst is True, "Alpha" is 0.22, "StartRows" is 3, "WindowSize" is 11, and MavgType is "M".

Teradata KNIME Components, Release 2.0



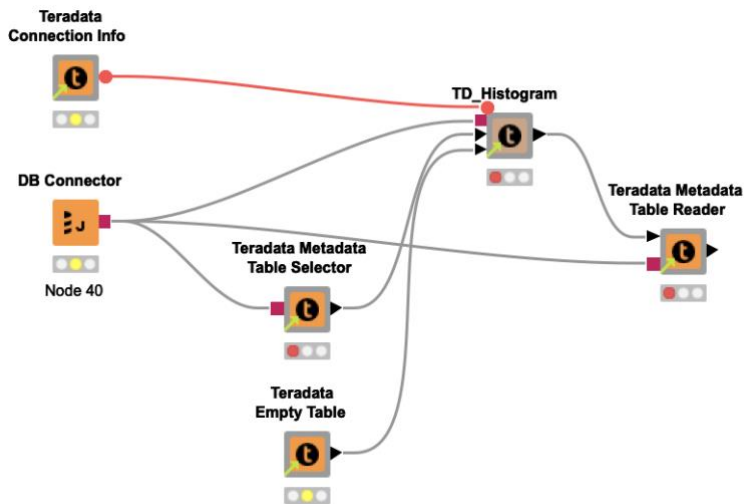
NOTE: for any parameters that are columns, the node must be run first and then the columns will be populated.

After executing and clicking the “Teradata Metadata Table Reader” node, the output table is shown.

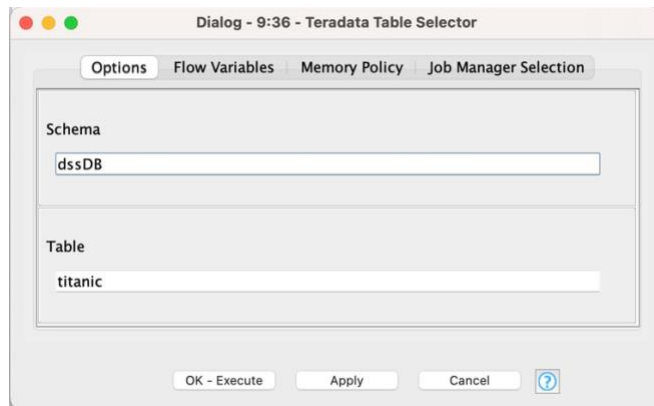
ID	id	name	period	stockprice	stockprice_mmavg
Row0	11	Company1	1961-06-01T00:00	492.0	492.0
Row1	14	Company1	1961-06-06T00:00	497.0	497.0
Row2	6	Company1	1961-05-24T00:00	459.0	459.0
Row3	24	Company1	1961-06-20T00:00	478.0	478.0
Row4	12	Company1	1961-06-02T00:00	498.0	498.0
Row5	13	Company1	1961-06-05T00:00	499.0	499.0
Row6	16	Company1	1961-06-08T00:00	490.0	490.0
Row7	17	Company1	1961-06-09T00:00	489.0	489.0
Row8	15	Company1	1961-06-07T00:00	496.0	496.0
Row9	22	Company1	1961-06-16T00:00	482.0	482.0
Row10	7	Company1	1961-05-25T00:00	463.0	463.0
Row11	4	Company1	1961-05-22T00:00	459.0	459.0
Row12	19	Company1	1961-06-13T00:00	487.0	487.0
Row13	25	Company1	1961-06-21T00:00	479.0	479.0
Row14	2	Company1	1961-05-18T00:00	457.0	457.0
Row15	23	Company1	1961-06-19T00:00	479.0	479.0
Row16	8	Company1	1961-05-26T00:00	479.0	479.0
Row17	18	Company1	1961-06-12T00:00	478.0	478.0
Row18	9	Company1	1961-05-29T00:00	493.0	493.0
Row19	3	Company1	1961-05-19T00:00	452.0	452.0
Row20	5	Company1	1961-05-23T00:00	462.0	462.0
Row21	10	Company1	1961-05-31T00:00	490.0	490.0
Row22	1	Company1	1961-05-17T00:00	460.0	460.0
Row23	21	Company1	1961-06-15T00:00	487.0	487.0
Row24	20	Company1	1961-06-14T00:00	491.0	498.272727272725

Workflow Example 2 – TD_Histogram

The second example is TD_Histogram, a SQLE function. This function shows the use of the “Teradata Empty Table” node. TD_Histogram takes in two input tables, but the second input is optional and so we use the “Teradata Empty Table” as the second input.



For the “Teradata Metadata Table Selector”, we have Schema set to “dssDB” and Table to “titanic”.



For the specific parameters of TD_Histogram, we have TargetColumn set to “age”, MethodType set to “STURGES”, NBins set to 1, Inclusion as “LEFT”, and GroupbyColumns as none of the columns.

Teradata KNIME Components, Release 2.0

TargetColumn

Manual Selection Wildcard/Regex Selection Type Selection

Exclude

Filter

- ? PASSENGER
- ? SURVIVED
- ? pclass
- ? name
- ? sex
- ? sibsp
- ? parch

Enforce exclusion

Include

Filter

- ? age

Enforce inclusion

MethodType

STURGES

NBins (integers separated by new line)

1

Inclusion

LEFT RIGHT

GroupbyColumns

Manual Selection Wildcard/Regex Selection Type Selection

Exclude

Filter

- ? age
- ? sibsp
- ? parch
- ? ticket
- ? fare
- ? cabin
- ? embarked

Enforce exclusion

Include

Filter

No columns in this list

Enforce inclusion

After executing and clicking the “Teradata Metadata Table Reader” node, the output table is shown.

Console Node Monitor

Node: df_reader_1 (9:38)

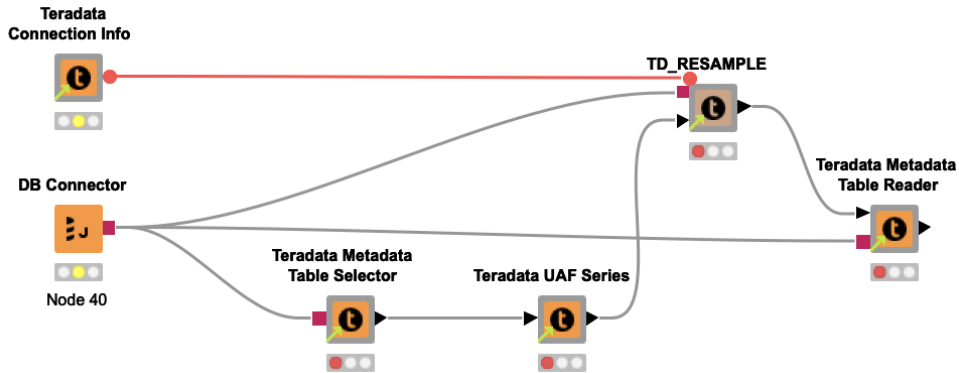
State: EXECUTED

Port Output Port 0 Load data Rows: 8, Columns: 6

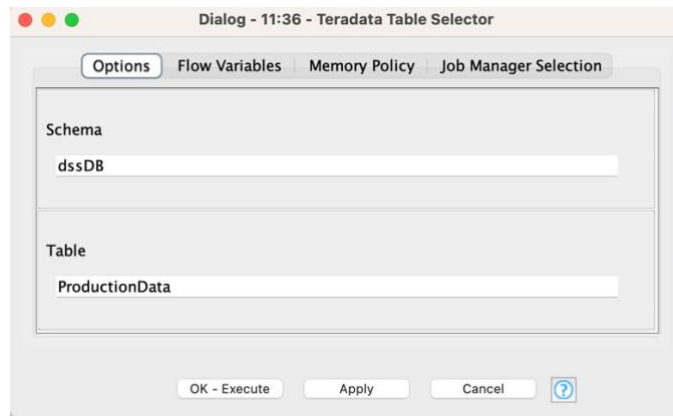
ID	ColumnName	Label	MinValue	MaxValue	CountOfValues	Bin_Percent
Row0	age	1	10.0	20.0	102	14.285714285714285
Row1	age	6	60.0	70.0	19	2.661064425770308
Row2	age	4	40.0	50.0	89	12.46498599439776
Row3	age	7	70.0	80.0	7	0.9803921568627451
Row4	age	2	20.0	30.0	220	30.81232492997199
Row5	age	0	0.0	10.0	62	8.683473389355742
Row6	age	5	50.0	60.0	48	6.722689075630252
Row7	age	3	30.0	40.0	167	23.389355742296917

Workflow Example 3 – TD_RESAMPLE

The third example is TD_RESAMPLE, a UAF function. This function shows the use of the “Teradata UAF Series” node.



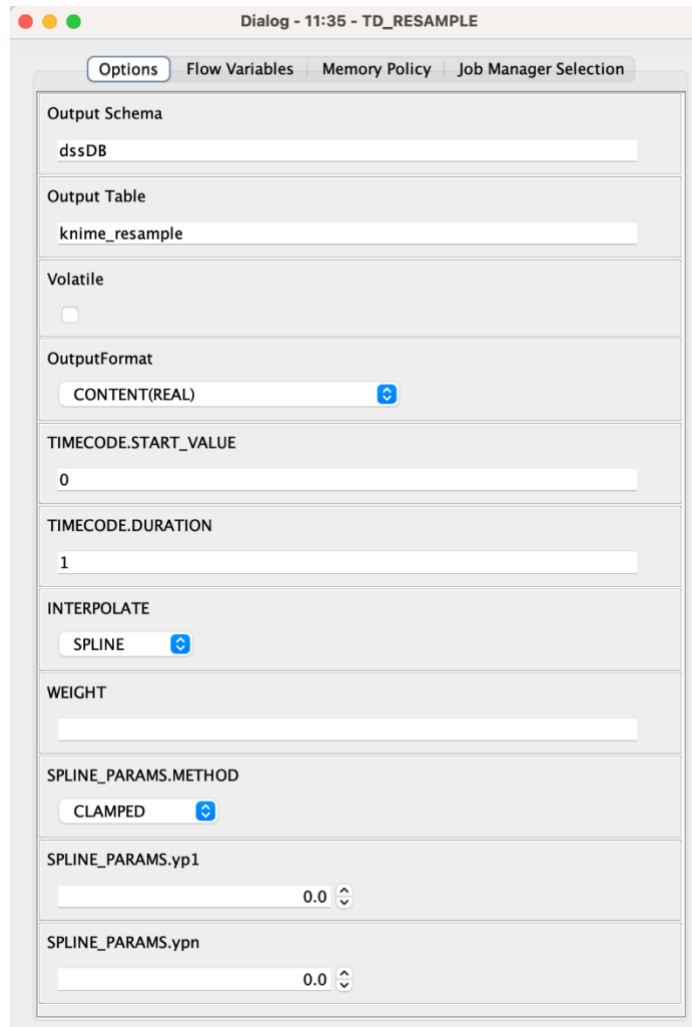
For the “Teradata Metadata Table Selector”, we have Schema as “dssDB” and Table as “ProductionData”.



The ID is set to “ProductID”, Row is “SEQUENCE”, Row Field is “TD_SEQNO”, Payload Field is “BEER_SALES”, Payload Content is “REAL”. Interval, Interval Type, and Zero are set to None. ID Sequence, Sequence Zero, and WHERE are left empty.

ID
ProductID
Row
SEQUENCE
ID Sequence
Row Field
TD_SEQNO
Payload Field
BEER_SALES
Payload Content
REAL
Layer
Interval
None
Interval Type
None
Zero
None
Sequence Zero

For the parameters inside TD_RESSAMPLE, the OutputFormat is "CONTENT(REAL)", TIMECODE.START_VALUE is 0, TIMECODE.DURATION is 1, INTERPOLATE is "SPLINE", WEIGHT is kept empty, SPLINE_PARAMS.METHOD is set to "CLAMPED", SPLINE_PARAMS.y₁ is 0.0, and SPLINE_PARAMS.y_n is 0.0.



After executing and clicking the “Teradata Metadata Table Reader” node, the output table is shown.

Console Node Monitor X

Node: df_reader_1 (11:38)

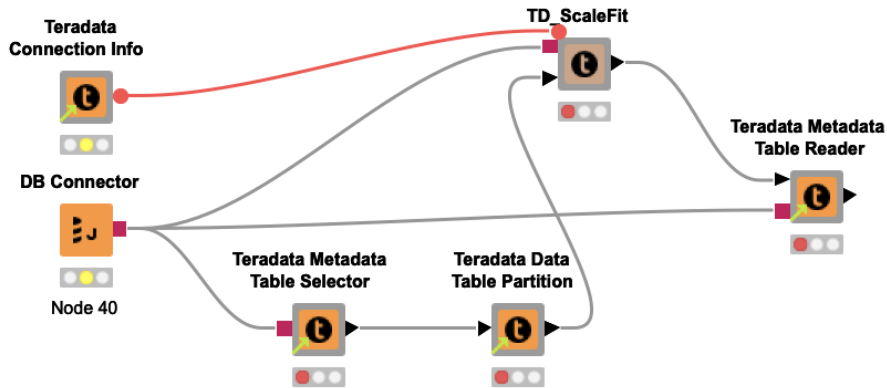
State: EXECUTED

Port Output Port 0 Load data Rows: 17, Columns: 3

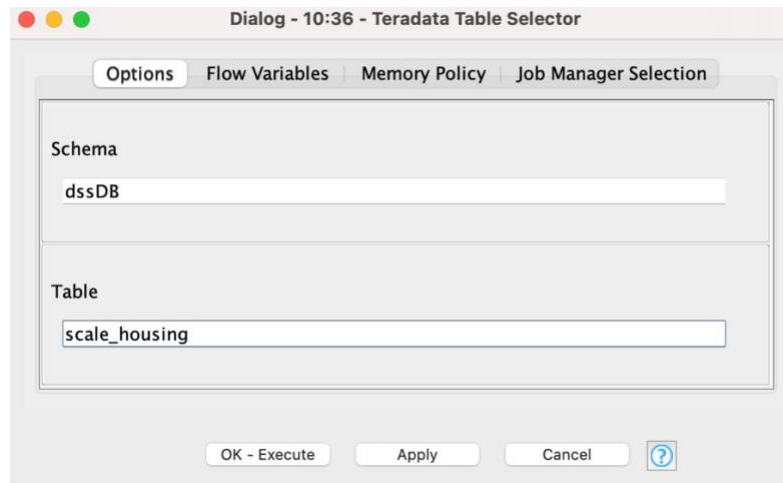
ID	ProductID	ROW_I	BEER_SALES
Row0	33	1	6.9
Row1	33	2	6.9
Row2	33	3	7.9
Row3	33	4	6.8
Row4	33	5	7.400246917071391
Row5	33	6	8.1
Row6	33	7	7.0
Row7	33	8	8.0
Row8	33	9	7.1
Row9	33	10	8.1
Row10	33	11	92.7
Row11	33	12	102.1
Row12	33	13	83.2

Workflow Example 4 – TD_ScaleFit

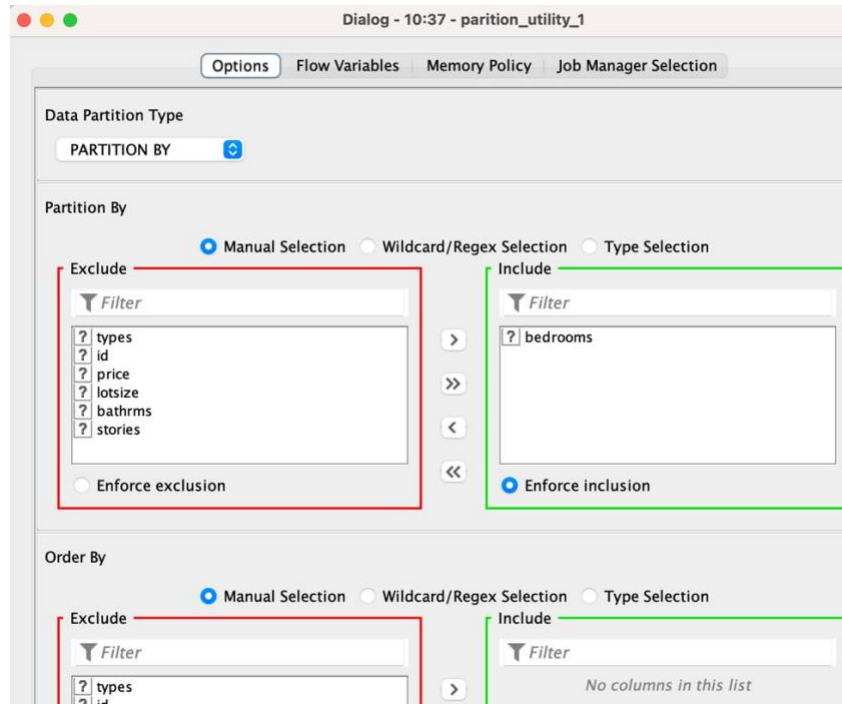
The fourth example is TD_ScaleFit, a SQLE function. This function shows the use of the “Teradata Data Table Partition” node.



In this example we have Schema as “dssDB” and Table as “scale_housing”.



For the “Teradata Data Table Partition” node, we set the Data Partition Type to “PARTITION BY” and choose to Partition By the column “bedrooms”.



For the specific parameters of TD_ScaleFit, we have TargetColumn set to “lotsize”, ScaleMethod set to “MEAN”, MissValue set to “KEEP”, GlobalScale as False, Multiplier as 1, and Intercept as 0.

Teradata KNIME Components, Release 2.0

TargetColumns

Manual Selection Wildcard/Regex Selection Type Selection

Exclude

Filter

- ? types
- ? id
- ? price
- ? bedrooms
- ? bathrms
- ? stories

Enforce exclusion

Include

Filter

- ? lotsize

Enforce inclusion

ScaleMethod (strings seperated by new line)

MEAN

MissValue

KEEP

GlobalScale

Multiplier (strings seperated by new line)

1

Intercept (strings seperated by new line)

0

After executing and clicking the “Teradata Metadata Table Reader” node, the output table is shown.

Console Node Monitor

Node: df_reader_1 (10:38)

State: EXECUTED

Port Output: Port 0 Load data Rows: 13, Columns: 2

ID	TD_STATYPE_SCLFIT	lotsize
Row0	scale	1.0
Row1	intercept	0.0
Row2	max	6650.0
Row3	count	10.0
Row4	null	0.0
Row5	multiplier	1.0
Row6	min	3060.0
Row7	sum	48420.0
Row8	ScaleMethodNumberMapping: [0:mean,1:sum,2:ustd,3:std,4:range,5:midrange,6:maxabs,7:rescale]	0.0
Row9	missvalue_KEEP	?
Row10	location	4842.0
Row11	globalscale_false	?
Row12	avg	4842.0

APPLY Component Setup

The APPLY component empowers users to utilize the APPLY table operator, an integral part of the Open Analytics Framework (OAF), on VantageCloud Lake systems. The Open Analytics Framework provides the flexibility to create and customize a dedicated Python or R environment within a VantageCloud Lake compute cluster. All the steps including creating and managing the user environment should be done outside KNIME.

When connected to these systems, users can execute the APPLY table operator (ATO) queries seamlessly. The APPLY node enhances analytics scalability by invoking and executing simultaneously instances of your script on containerized nodes in a VantageCloud Lake compute cluster.

To integrate the APPLY node into your KNIME workflow, follow these steps:

Begin by dragging the APPLY node from the Teradata>In-Vantage Scripting folder and dropping it onto your workflow canvas.

Next, ensure that the APPLY node is properly configured by providing it with two key inputs:

- Database Connection: Use the DB Connector node, a standard KNIME node designed for database connections, to establish a connection to your Teradata database. This step ensures that the APPLY node can interact seamlessly with the database environment.
- Table Input: Utilize the Teradata Metadata Table Selector node to select the specific table or data source that you want to process using the APPLY node.

For enhanced functionality and efficiency, consider utilizing the Teradata Connection Info node. This additional node enables you to set the default database for computations carried out within the APPLY node, streamlining your workflow and ensuring consistency in database interactions.

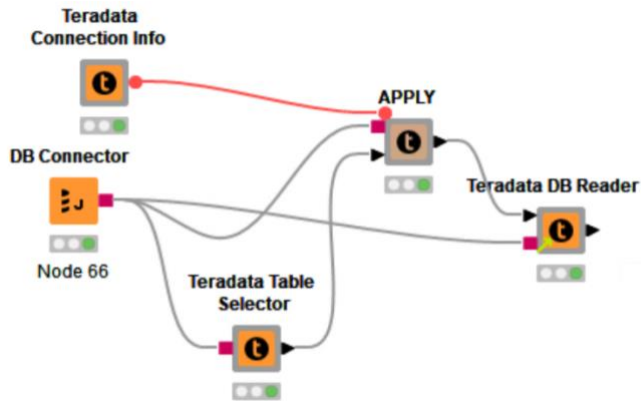
Additionally, ensure that your VantageCloud Lake system meets the following prerequisites to use the APPLY table operator effectively:

- The system must include the analytic compute cluster component. In compute clusters, standard or analytic compute groups of one or more standard or analytic compute profiles, respectively, can be specified.
- At least one analytic compute group must be created on the system. This is achieved by creating a compute group with its query strategy argument specified as "Analytic".
- Users must be given access privileges to the analytic compute group. Access is typically granted by your system administrator.
- Your session takes place in an analytic compute group you are a member of.

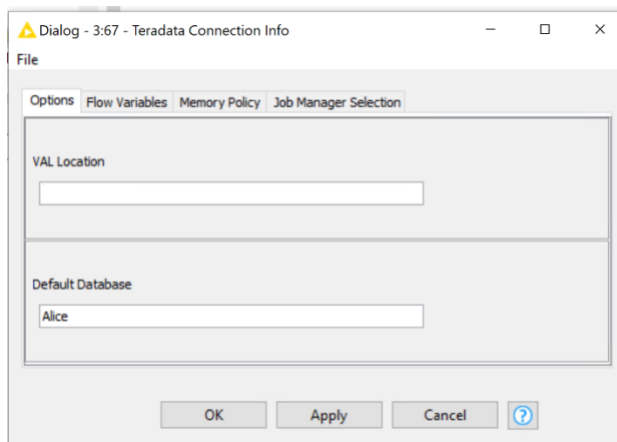
Let's understand the utilization of the component through a scoring use case. For this demonstration, we assume the following setup:

- You have a KNIME workflow with a testing Dataset "usecase5_table", whose rows you can score with a Python script "usecase5_knime_test.py" available in a Python environment "KNIME-DEMO" along with other associated files and required packages.

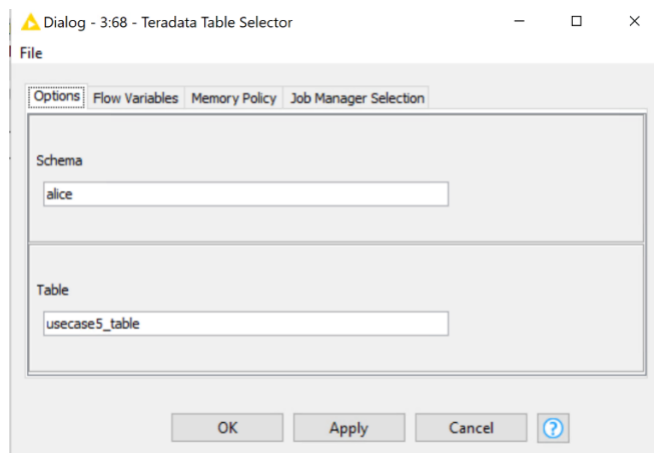
Workflow Example 5 – APPLY Scoring



In the “Teradata Connection Info” node, you set Default Database; in our use case example, this value is “Alice”. The VAL Location field is not related and can be left empty.



For the “Teradata Metadata Table Selector” node we specify the schema and the input table; in our use case example, the values are “alice” and “usecase5_table” respectively.



In the APPLY node, you have the option to specify arguments. You can hover over each argument to view its description and determine whether it is required or optional. Furthermore, clicking on the '?' symbol within the window provides detailed descriptions for all available arguments, enhancing your understanding and facilitating the configuration process.

Note: In the following, the node prevents using specific characters in some fields for security purposes as follows:

- When you specify column names in the input Dataset, the double quote " (unicode character U+0022) cannot be used in the name string. You can otherwise use any character in the column naming specification, according to the related Vantage documentation at: <https://docs.teradata.com/r/SQL-Fundamentals/July-2021/Basic-SQL-Syntax/Object-Names>
- The semicolon ; (unicode character U+003B) cannot be used in the Conditions field.

The following image shows the node configure screen.

Dialog - 8:61 - APPLY

File

Options | Flow Variables | Memory Policy | Job Manager Selection

Output Schema
alice

Output Table
output_test_table

Script Language
Python

Environment Name
KNIME-DEMO

Script Name
usecase5_knime_test.py

Quotechar
"

Delimiter
,

Arguments
4 5 10 6 480

Input Column(s)

Conditions

Data Partition
None

Nulls Listing
FIRST

Column name(s) and sequence to order by

Column name(s) to partition by

Output Columns
customer_visits VARCHAR(10),customer_renges VARCHAR(10)

OK Apply Cancel ?

The first two fields, the output schema, and the output table name, are mandatory for the component to run. Please specify these fields based on your desired output location and naming convention. The remaining arguments are explained as follows:

- Script Language
 - Select whether you are submitting a Python or an R script.
- Script Input Columns
 - Specify the column names of the recipe input dataset in the order needed by your script. Specify comma separated values and add as many column names as you need. If you leave the field blank and specify no names, then the script receives as input all columns in the input dataset.
 - This field corresponds to the APPLY Table Operator ON clause.
 - Optionally, use the Conditions field to specify any logical conditions for row selection across your script input columns. If specified, then conditions must be in SQL format. Multiple conditions should be separated by any of the **AND** and **OR** keywords. This field is reserved for the input to the optional WHERE clause in the APPLY Table Operator ON clause.
- Environment Name
 - Specify the name of the environment that the APPLY_COMMAND will execute in. Currently, the environment must be created outside the KNIME interface.
- Script Name
 - Specify the script name which is uploaded in the specific environment.
- Delimiter
 - Specify the delimiter to be used for reading columns from a row and writing the result columns.
- It must be a valid Unicode point.
 - The default value is comma (,).
- Quotechar
 - Specify the character used to quote all input and output values to the script.
 - The default value is double quote (").
 - Using QUOTECHAR enables the database to distinguish between NULL values and empty VARCHARs. A VARCHAR with length zero is quoted, while NULL values are not.
 - If the APPLY function finds a quote character in the data, the data escapes by a second quote character. For example:
 - He said, "Hey there!"
Default QUOTECHAR ("") becomes
"He said, ""Hey there!"""
- Arguments
 - Optional arguments that your script may need for execution. Specify space separated values in the argument fields.
- Data Partition Option
 - Select the default option **None** in the drop-down menu to have the APPLY input

data treated as a single partition. Select any of the other options so that one or more columns of your input Dataset are designated as partition columns in the **Column name(s) to partition by** field.

- If you specify to partition **By column values**, then different partitions are determined by sets of rows with unique values (if one partition column is specified) or with unique value combinations (if multiple partition columns are specified) in the designated partition column(s). This option corresponds to the APPLY Table Operator PARTITION BY clause.
 - If you specify to partition **By Database AMP Hash**, then data are distributed to Analytics Database AMPs based on AMP hash values contained in the designated partition column(s). This option corresponds to the APPLY Table Operator HASH BY clause.
- **Column name(s) to partition by**
 - Specify one or more column names from the recipe input dataset to use for data partitioning according to the **Data Partition Option** menu choice. Specify comma separated values to add column names as desired.
 - This option is valid only when the **Data Partition Option** is other than None.
 - **Column name(s) and sequence to order by**
 - Specify one or more column names from the recipe input dataset to use their data for row ordering in the partitions. Specify comma separated values to add column names as desired.
 - For each specified column, also specify the order sequencing separated by comma. By default, columns are ordered in descending values order.
 - This option is valid only when the **Data Partition Option** is other than None.
 - This option corresponds to the APPLY Table Operator ORDER BY clause (if data partitioning is by column values) or LOCAL ORDER BY clause (if data partitioning is by Database AMP hash).
 - **Nulls Listing**
 - Specify the positioning of null values in the output.
 - On selecting FIRST, NULL results are to be listed first.
 - On selecting LAST, NULL results are to be listed at the end.
 - By default, the value is set to 'N/A'.
 - **Output columns**
 - In this section you must specify all output variables of your script. Observe that you need to know the number and types of variables returned by your script. To this end, comma to add as many rows as needed. This specification corresponds to the APPLY Table Operator RETURNS clause. Ensure that the output column names, and their respective data types are separated by a space, with each new argument separated by a comma.

Upon setting up the arguments and executing the node, the output Dataset is populated with data from the Analytics Database table with the scoring results, as was specified in the node's output table argument. After executing and clicking the "Teradata Metadata Table Reader" node, the output table is shown.

Port 1 - 3:1 - Teradata DB Reader

File Edit Hilite Navigation View

Table "database" - Rows: 96 Spec - Columns: 2 Properties Flow Variables

Row ID	S customer_visits	S customer_renges
Row0	9	1
Row1	18	9
Row2	11	4
Row3	6	2
Row4	14	7
Row5	13	3
Row6	17	11
Row7	12	4
Row8	7	1
Row9	15	4
Row10	16	3
Row11	4	0
Row12	9	4
Row13	18	5
Row14	11	3
Row15	6	1
Row16	14	5

SCRIPT Component Setup

The SCRIPT node in KNIME facilitates the seamless execution of Python and R scripts within the Teradata Vantage environment. This functionality is achieved by interacting with table operator objects in the Analytics Database. It's important to note that this capability is specifically designed for use with VantageCloud Enterprise or Vantage Core systems. The pre-requisite steps like uploading the file should currently be done outside KNIME.

When connected to these systems, users can leverage the Script Table Operator (STO) queries seamlessly. The SCRIPT node enhances analytics scalability by invoking and executing instances of your script simultaneously on each processing unit of the Analytics Database.

To integrate the SCRIPT node into your KNIME workflow, follow these steps:

Begin by dragging the SCRIPT node from the Teradata>In-Vantage Scripting folder and dropping it onto your workflow canvas.

Next, ensure that the SCRIPT node is properly configured by providing it with two key inputs:

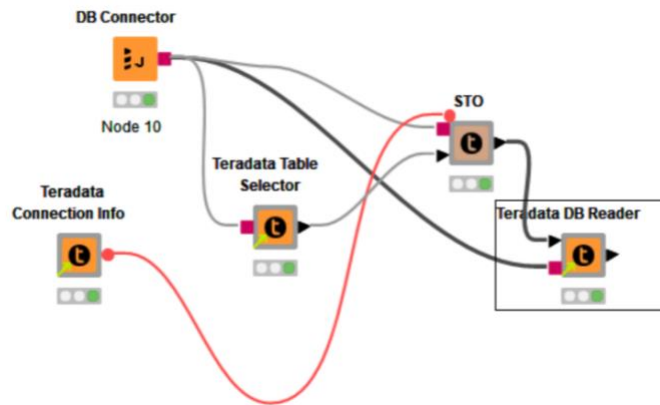
- **Database Connection:** Use the DB Connector node, a standard KNIME node designed for database connections, to establish a connection to your Teradata database. This step ensures that the SCRIPT node can interact seamlessly with the database environment.
- **Table Input:** Utilize the Teradata Metadata Table Selector node to select the specific table or data source that you want to process using the SCRIPT node.

For enhanced functionality and efficiency, consider utilizing the Teradata Connection Info node. This additional node enables you to set the default database for computations carried out within the SCRIPT node, streamlining your workflow and ensuring consistency in database interactions.

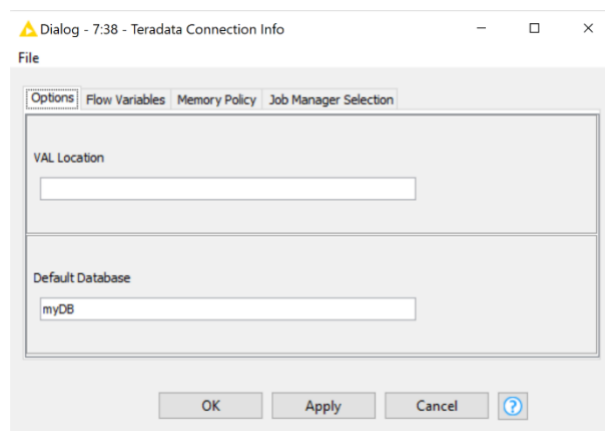
Let's understand the utilization of the component through a scoring use case. For this demonstration, we assume the following setup:

- You possess a KNIME workflow containing a testing Dataset named 'ads_final_test.' This dataset's rows are to be scored using a Python script available in the 'myDB' database, assuming you have the necessary permissions to access and execute scripts within this database.
- Your user account for the target Analytics Database has been granted specific privileges required to run SCRIPT effectively.

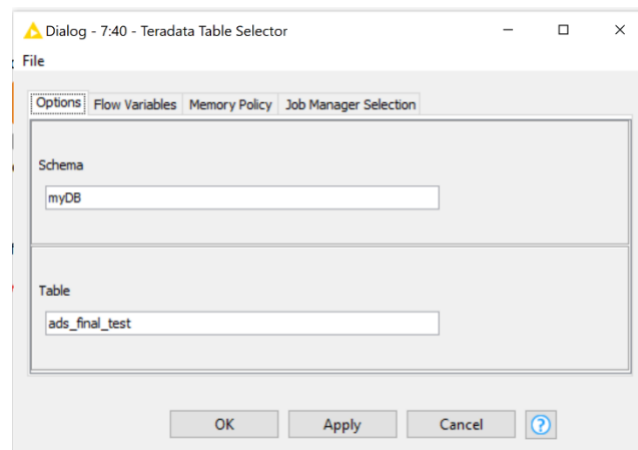
Workflow Example 6 – SCRIPT Scoring



In the “Teradata Connection Info” node, you can set Default Database; in our use case example, this value is “myDB”. The VAL Location field is not related and can be left empty.



For the “Teradata Metadata Table Selector” node we specify the schema and the input table; in our use case example, the values are “myDB” and “ads_final_test” respectively.



In the SCRIPT node, you have the option to specify arguments. You can hover over each argument to view its description and determine whether it is required or optional. Furthermore, clicking on the '?' symbol within the window provides detailed descriptions for all available arguments, enhancing your understanding and facilitating the configuration process.

Note: In the following, the node prevents using specific characters in some fields for security purposes as follows:

- When you specify column names in the input Dataset, the double quote " (unicode character U+0022) cannot be used in the name string. You can otherwise use any character in the column naming specification, according to the related Vantage documentation at: <https://docs.teradata.com/r/SQL-Fundamentals/July-2021/Basic-SQL-Syntax/Object-Names>
- The semicolon ; (unicode character U+003B) cannot be used in the Conditions field.

The following image shows the node configure screen.

Dialog - 7:41 - STO



File

Options | Flow Variables | Memory Policy | Job Manager Selection

STO Database
myDB

Output Schema
myDB

Output Table
output_test_table

Script Language
Python

Script Name
exIpScoViaDSS.py

Input Column(s)

Conditions

Arguments

Data Partition
By Database AMP Hash

Column name(s) to partition by
age,cust_id

Column name(s) and sequence to order by
age asc,
cust_id desc

Nulls Listing
LAST

Output Columns
Cust_ID INTEGER, Prob_0 FLOAT, Prob_1 FLOAT, Actual_Value INTEGER

The first two fields, the output schema, and the output table name, are mandatory for the component to run. Please specify these fields based on your desired output location and naming convention. The remaining arguments are explained as follows:

- STO Database
 - Specify the associated database name where the script files are uploaded.

- Script Language
 - Select whether you are submitting a Python or an R script.

- Script Input Columns
 - Specify the column names of the recipe input dataset in the order needed by your script. Specify comma separated values and add as many column names as you need. If you leave the field blank and specify no names, then the script receives as input all columns in the input dataset.
 - This field corresponds to the SCRIPT Table Operator ON clause.
 - Optionally, use the Conditions field to specify any logical conditions for row selection across your script input columns. If specified, then conditions must be in SQL format. Multiple conditions should be separated by any of the **AND** and **OR** keywords. This field is reserved for the input to the optional WHERE clause in the SCRIPT Table Operator ON clause.

- Arguments
 - Optional arguments that your script may need for execution. Specify space separated values in the argument fields.

- Data Partition Option
 - Select the default option **None** in the drop-down menu to have the SCRIPT input data treated as a single partition. Select any of the other options so that one or more columns of your input Dataset are designated as partition columns in the **Column name(s) to partition by** field.
 - If you specify to partition **By column values**, then different partitions are determined by sets of rows with unique values (if one partition column is specified) or with unique value combinations (if multiple partition columns are specified) in the designated partition column(s). This option corresponds to the SCRIPT Table Operator PARTITION BY clause.
 - If you specify to partition **By Database AMP Hash**, then data are distributed to Analytics Database AMPs based on AMP hash values contained in the designated partition column(s). This option corresponds to the SCRIPT Table Operator HASH BY clause.

- Column name(s) to partition by
 - Specify one or more column names from the recipe input dataset to use for data partitioning according to the **Data Partition Option** menu choice. Specify comma separated values to add column names as desired.
 - This option is valid only when the **Data Partition Option** is other than None.

- Column name(s) and sequence to order by
 - Specify one or more column names from the recipe input dataset to use their data for row ordering in the partitions. Specify comma separated values to add column names as desired.
 - For each specified column, also specify the order sequencing separated by comma. By default, columns are ordered in descending values order.
 - This option is valid only when the **Data Partition Option** is other than None.
 - This option corresponds to the SCRIPT Table Operator ORDER BY clause (if data partitioning is by column values) or LOCAL ORDER BY clause (if data partitioning is by Database AMP hash).

- Nulls Listing
 - Specify the positioning of null values in the output.
 - On selecting FIRST, NULL results are to be listed first.
 - On selecting LAST, NULL results are to be listed at the end.
 - By default, the value is set to 'N/A'.

- Output columns
 - In this section you must specify all output variables of your script. Observe that you need to know the number and types of variables returned by your script. To this end, comma to add as many rows as needed. This specification corresponds to the SCRIPT Table Operator RETURNS clause. Ensure that the output column names, and their respective data types are separated by a space, with each new argument separated by a comma.

Upon setting up the arguments and executing the node, the output Dataset is populated with data from the Analytics Database table with the scoring results, as was specified in the node's output table argument. After executing and clicking the "Teradata Metadata Table Reader" node, the output table is shown.

Port 1 - 7:1 - Teradata DB Reader

File Edit Hilite Navigation View

Table "database" - Rows: 4183 Spec - Columns: 4 Properties Flow Variables

Row ID	I Cust_ID	D Prob_0	D Prob_1	I Actual_...
Row0	31344768	0	1	1
Row1	31355923	1	0	0
Row2	17717206	1	0	0
Row3	24538104	0.9	0.1	0
Row4	28631988	0.9	0.1	0
Row5	24535494	0	1	1
Row6	23175148	1	0	0
Row7	29996780	1	0	0
Row8	19076736	0	1	1
Row9	21805952	0	1	1
Row10	16357512	0	1	1
Row11	24525828	0	1	1
Row12	20448420	0.9	0.1	0
Row13	17712760	0	1	1
Row14	14994760	0.1	0.9	1
Row15	17721223	0.3	0.7	1
Row16	19081426	0.3	0.7	1

BYOM Scoring Component Setup

The BYOM-scoring node facilitates efficient and scaled scoring operations directly within the Analytics Database, leveraging table data where one or more models have been imported. It is essential to configure the model source table and model names outside the KNIME platform prior to using this node.

To integrate the BYOM-Scoring node into your KNIME workflow, follow these steps: Begin by dragging the BYOM-Scoring node from the Teradata>BYOM folder and dropping it onto your workflow canvas.

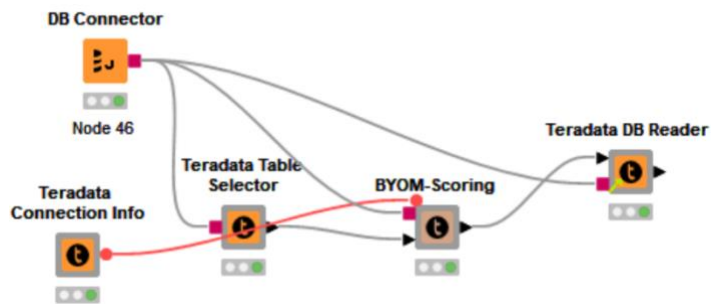
Next, ensure that the BYOM node is properly configured by providing it with two key inputs:

- **Database Connection:** Use the DB Connector node, a standard KNIME node designed for database connections, to establish a connection to your Teradata database. This step ensures that the BYOM-Scoring node can interact seamlessly with the database environment.
- **Table Input:** Utilize the Teradata Metadata Table Selector node to select the specific table or data source that you want to process using the BYOM-Scoring node.

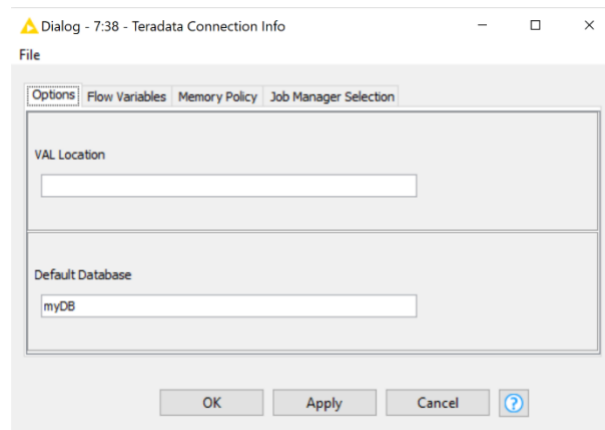
For enhanced functionality and efficiency, consider utilizing the Teradata Connection Info node. This additional node enables you to set the default database for computations carried out within the BYOM-Scoring node, streamlining your workflow and ensuring consistency in database interactions.

To illustrate the steps involved, let's consider a use case where you wish to consume a saved Predictive Model Markup Language (PMML) model that was previously exported to a Teradata Vantage system.

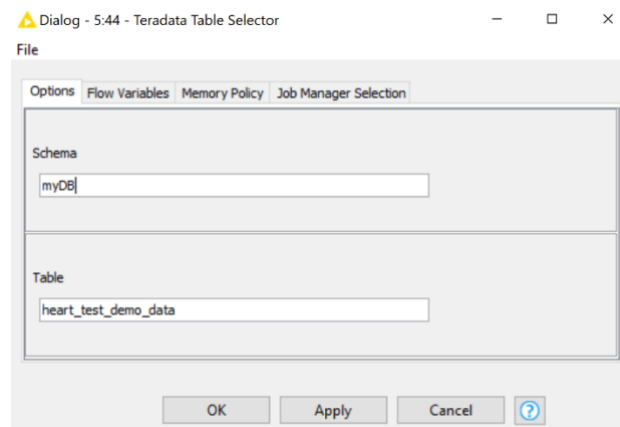
Workflow Example 7 – BYOM Scoring



In the “Teradata Connection Info” node, you can set Default Database; in our use case example, this value is “myDB”. The VAL Location field is not related and can be left empty.




For the “Teradata Metadata Table Selector” node we specify the schema and the input table; in our use case example, the values are “myDB” and “heart_test_demo_data” respectively.



In the BYOM-Scoring node, you have the option to specify arguments. You can hover over each argument to view its description and determine whether it is required or optional. Furthermore, clicking on the '?' symbol within the window provides detailed descriptions for all available arguments, enhancing your understanding and facilitating the configuration process.

The following image shows the node configure screen.

 Dialog - 5:48 - BYOM-Scoring

File

Options	Flow Variables	Memory Policy	Job Manager Selection
Output Schema			
<input type="text" value="myDB"/>			
Output Table			
<input type="text" value="knime_output_tbl"/>			
Scoring Model Type			
<input type="text" value="PMML"/>			
H2O Model type (If applicable)			
<input type="text"/>			
Predict Function Database Name			
<input type="text" value="mldb"/>			
Model Table Name Input			
<input type="text" value="byom_models"/>			
Model DB Name Input			
<input type="text" value="alice"/>			
Model ID			
<input type="text" value="pmml"/>			

The first two fields, the **output schema**, and the **output table name**, are mandatory for the component to run. Please specify these fields based on your desired output location and naming convention.

The next field is the **Scoring Model Type**. You can choose among the options of PMML, H2O MOJO, ONNX, DATAROBOT and a JAR scoring lib; the latter is the native Dataiku model format.

Note for H2O MOJO models: If you select the H2O MOJO option as the scoring model type, then an additional field **H2O Model Type** must be specified. You can choose between the options:

- OPENSOURCE, and
- DAI (Driverless AI)

Furthermore, if you have an H2O DAI model to use, then you are required to specify the associated license information too. In this scenario, the additional couple of fields **H2O DIA License DB Input**, and **H2O DIA License DB Table Input** appear with a 'If Applicable' option,

in which you need to specify the database and table names where your H2O DIA model license resides on the connected server.

After the model type has been selected, in **Predict Function Database Name** field, you specify the database name in the Vantage server where the prediction function resides. The default installation location for the BYOM software inside an Analytics Database is a user database called **mladb**. If unsure, first validate this information with your Teradata Vantage Database Administrator.

The latter section asks you to

- pinpoint the model you wish to use for the scoring task on the Teradata Vantage system where the input dataset table is located, and
- specify additional options.

Sequentially, the section presents you with the following fields:

Model DB Name Input

- You will be provided an empty field to specify explicitly the database name.

Model Table Name Input

- You will be provided an empty field to specify explicitly the target table name.

Model Id/Name:

- Models are named based on their model ID. You will have to specify the corresponding model id.

If you specified your model to be of type H2O DAI, then you will have to specify a couple more fields at this point, as follows:

H2O DIA License DB Input:

- You will be provided an empty field to specify explicitly the database name.

H2O DIA License Table Input

- You will be provided an empty field to specify explicitly the target table name.

The remaining settings on the configuration screen enable specification of syntax elements of the Analytics Database BYOM prediction function that are supported by the node, as follows:

The **Accumulate** options specify to show all or a desired number of columns of the test table in the recipe's output table. The recipe enables you to

- accumulate all columns by clicking the corresponding **Accumulate All Columns** button, or
- accumulate specific columns, by clicking one or more times on the column name. You can specify the columns you want to accumulate by adding them to the 'Include' list.

The **ModelOutputFields Columns** enables you to optionally specify the scoring fields to output as individual columns.

Note 1: When you select the **Accumulate All Columns** check box, then the **Accumulate Specific Columns** is ignored.

Note 2: When you specify to use the **ModelOutputFields** check box then the **ModelOutputFields Columns** is ignored.

The **Overwrite model cache** check box enables you to activate the OverwriteCachedMode option of the Analytics Database BYOM prediction function. This option specifies any model left in cache be cleared. It is typically used when you update a model, so that the newer model version is used instead of the cached.

For further details about these syntax elements, see the Analytics Database BYOM prediction functions Syntax Elements sections at the Teradata documentation address:

https://docs.teradata.com/r/Enterprise_IntelliFlex_Lake_VMware/Teradata-VantageTM-Bring-Your-Own-Model-User-Guide/BYOM-Functions.

Upon setting up the arguments and executing the node, a corresponding query is formed in the node back-end and pushed to the connected Analytics Database for execution. The output Dataset is populated with data from the Analytics Database table with the scoring results, as was specified in the node's output table argument. After executing and clicking the "Teradata Metadata Table Reader" node, the output table is shown.

Port 1 - 5:45 - Teradata DB Reader

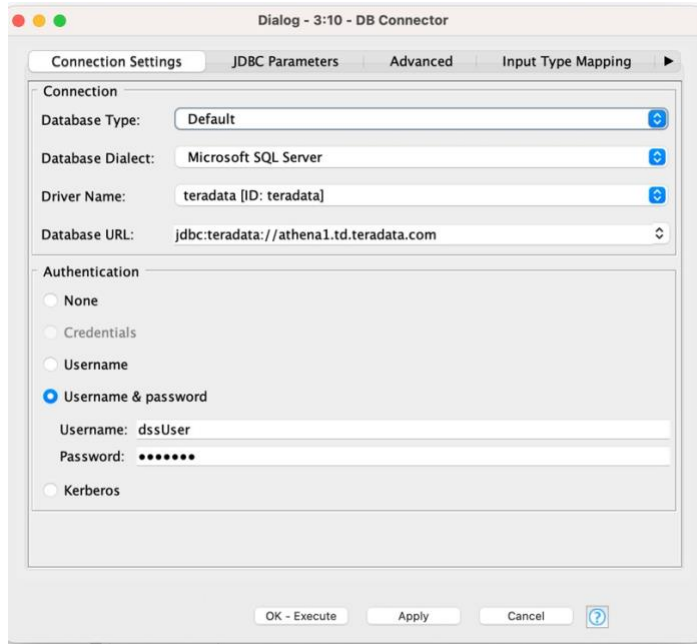
File Edit Hilite Navigation View

Table "database" - Rows: 5766 Spec - Columns: 3 Properties Flow Variables

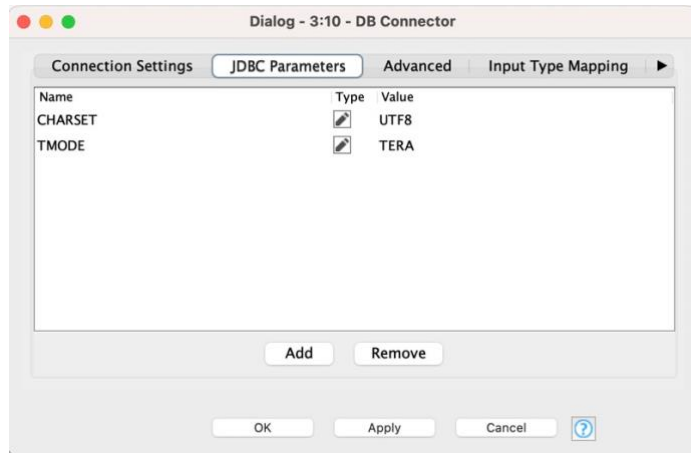
Row ID	heart_f...	json_report
Row0	1	{"prediction": "0", "proba_0": 0.6415942236355212, "proba_1": 0.35840577636447873}
Row1	0	{"prediction": "0", "proba_0": 0.7331168918775449, "proba_1": 0.2668831081224551}
Row2	1	{"prediction": "1", "proba_0": 0.47752762295325235, "proba_1": 0.5224723770467476}
Row3	0	{"prediction": "0", "proba_0": 0.7007438843830714, "proba_1": 0.29925611561692855}
Row4	1	{"prediction": "1", "proba_0": 0.42431568571612055, "proba_1": 0.5756843142838795}
Row5	0	{"prediction": "0", "proba_0": 0.6260217894488714, "proba_1": 0.37397821055112856}
Row6	1	{"prediction": "1", "proba_0": 0.5095292412297068, "proba_1": 0.4904707587702932}
Row7	0	{"prediction": "0", "proba_0": 0.6999543575341416, "proba_1": 0.3000456424658584}
Row8	1	{"prediction": "0", "proba_0": 0.7008792869583685, "proba_1": 0.29912071304163146}
Row9	0	{"prediction": "1", "proba_0": 0.5514672103880327, "proba_1": 0.44853278961196724}
Row10	1	{"prediction": "0", "proba_0": 0.663580412746479, "proba_1": 0.3364195872535209}
Row11	0	{"prediction": "1", "proba_0": 0.4855569443981615, "proba_1": 0.5144430556018385}

Appendix: DB Connector Setup prior to version 5.2

The UI for DB Connector will ask to specify the “Database Type”, leave this as “Default”. For “Database Dialect” make sure to choose “Microsoft SQL Server”. The “Driver Name” will be “teradata [ID: teradata]” (name based on what you specified for database driver step before). Enter your Username and Password for your Teradata system.



For the JDBC Parameters set “CHARSET” to the value “UTF8” and set “TMODE” to the value “TERA”.



In the Advanced tab, click the checkbox on for “Automatically reconnect to database” and “Restore database connection”, which will ensure that the database settings will be saved after exiting out of the application.

