KEYSQL® STUDIO
USER DOCUMENTATION

28-MAY-2022

The KeySQL Studio application, described in this document, provides an easy-to-use interface for analytics and working with KeySQL Server.

This tutorial is a companion to the KeySQL® Definitive Guide which is the primary reference for the KeySQL language. Please consult the Definitive Guide for a full explanation of the KeySQL data model as well as additional command.

KeySQL Studio is integrated with KeySQL Data Modeling Tool which is described in the KeySQL® Data Modeling Tool User Guide.

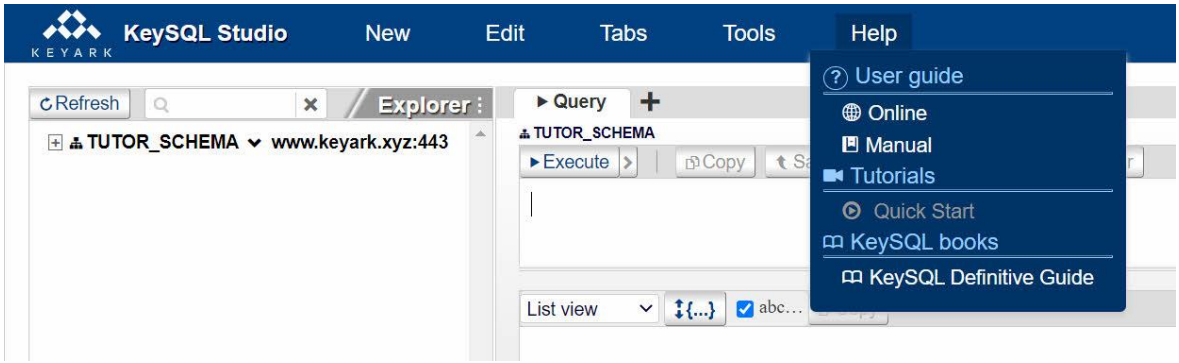
CONTENTS

1 GETTING ACQUAINTED

KeySQL® Studio is a browser-based app for viewing, analyzing, and manipulating data stored in KeySQL Server. It runs on the most modern browsers including:

- Chrome
- Firefox
- Microsoft Edge
- Safari (iOS & macOS only)
- Opera

1.1 ONLINE HELP

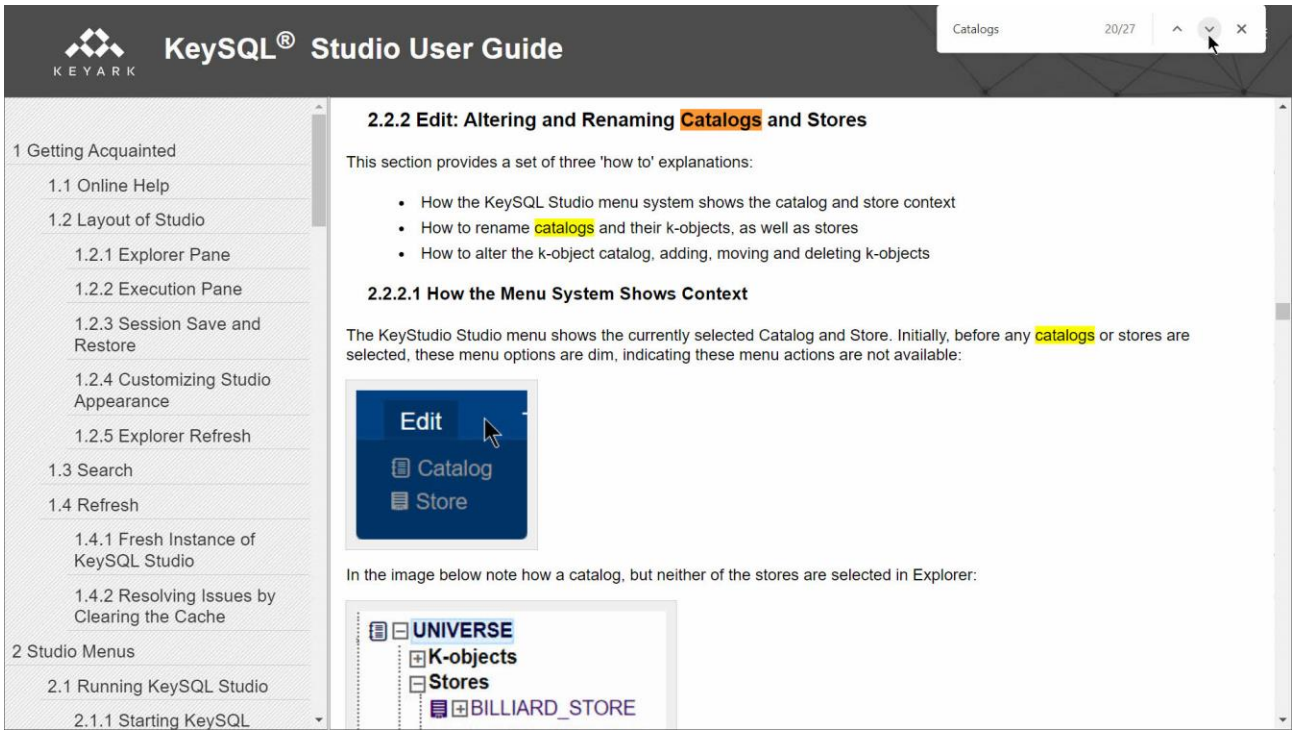


KeySQL Studio offers help in four ways:

- User guide as an online wiki
- User guide as downloadable PDF
- Tutorials published on Keyark's YouTube channel
- Tooltips shown when the cursor is positioned over a control

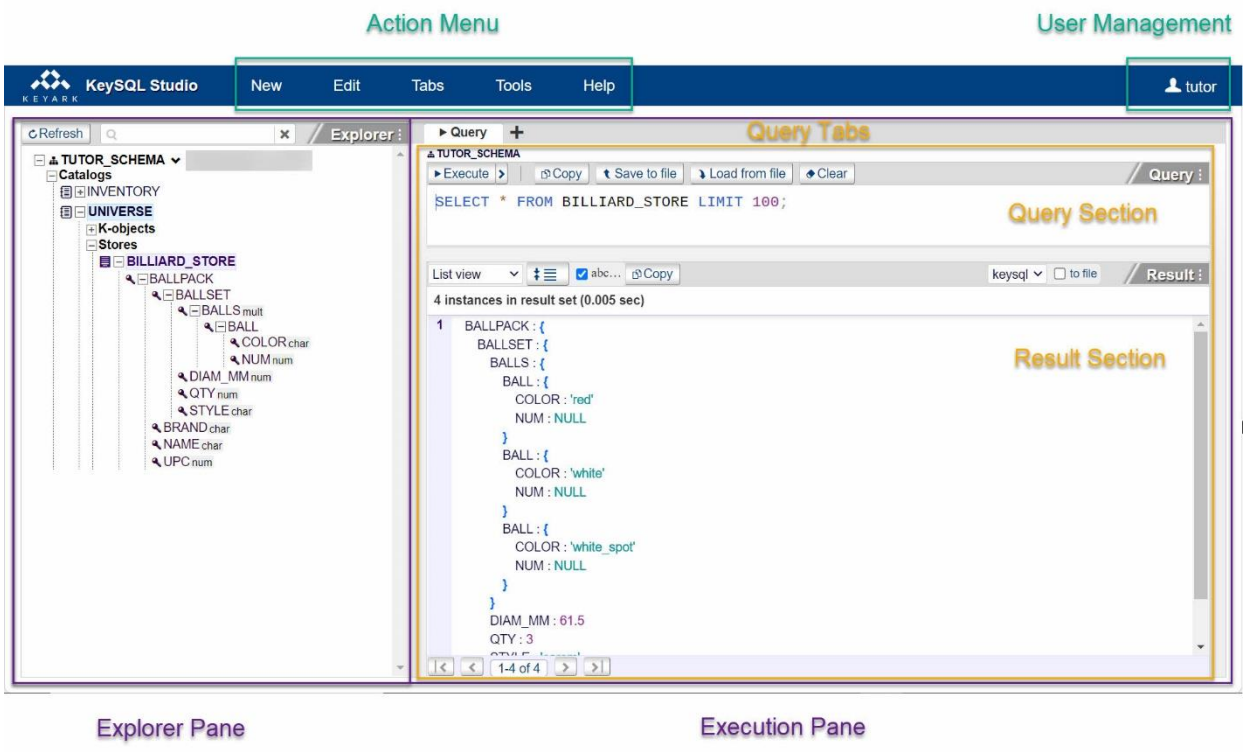
Both the User guide and the Tutorials are accessed from the Help menu pad, at the far right of the menu.

The online version of the User guide can be searched by clicking CTRL+F (Windows) or Command+F (Mac) as shown here:



1.2 LAYOUT OF STUDIO

The screenshot below illustrates a query to examine the contents of store BILLIARD_STORE, found within the catalog UNIVERSE, as shown in the *Explorer* pane. Your query statements are entered into the *Query* section of the tab, and results appear in the *Result* section of that tab. Clicking the + icon to the right of the initial Query tab will open a new tab.



1.2.1 Explorer Pane

Quick explanation for the Explorer pane layout and purpose

Pane	Purpose	More Detail
Explorer pane	Shows the catalogs such as UNIVERSE, and the k-objects for each catalog. For each catalog shows the stores such as BILLIARD_STORE	This is a tree structure that when fully expanded shows catalogs, their stores, and k-objects within. Left click and select KeySQL statements that will be pasted into the Execution pane. Export commands are executed immediately.

This pane facilitates easy manipulation of catalogs, stores, and k-objects within.

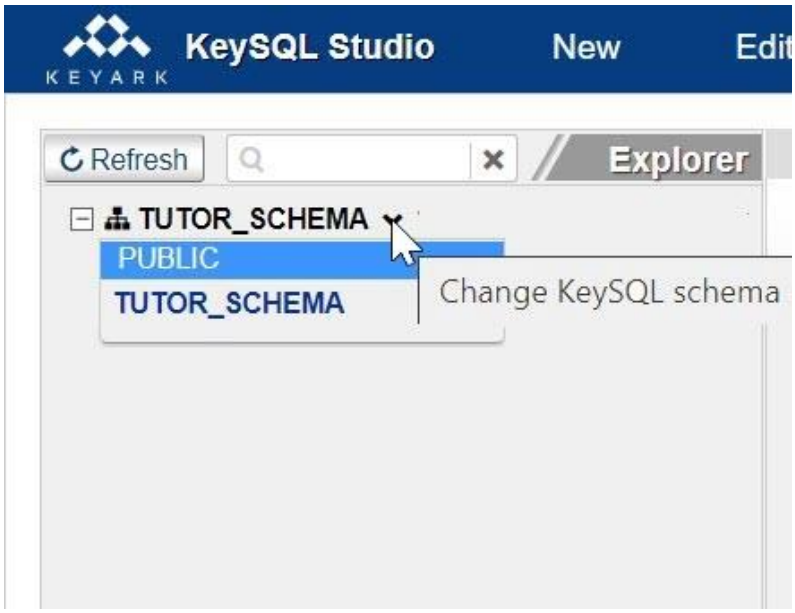
Selecting a menu choice provides for easy automation of a task by:

- generating KeySQL statements inserted into a newly opened tab, such as for dropping a catalog or store, ready for you to execute
- opening a specialized pane, such as for renaming a catalog or store, where your input is needed before the statements can be generated

1.2.1.1 WORKING WITH SCHEMAS

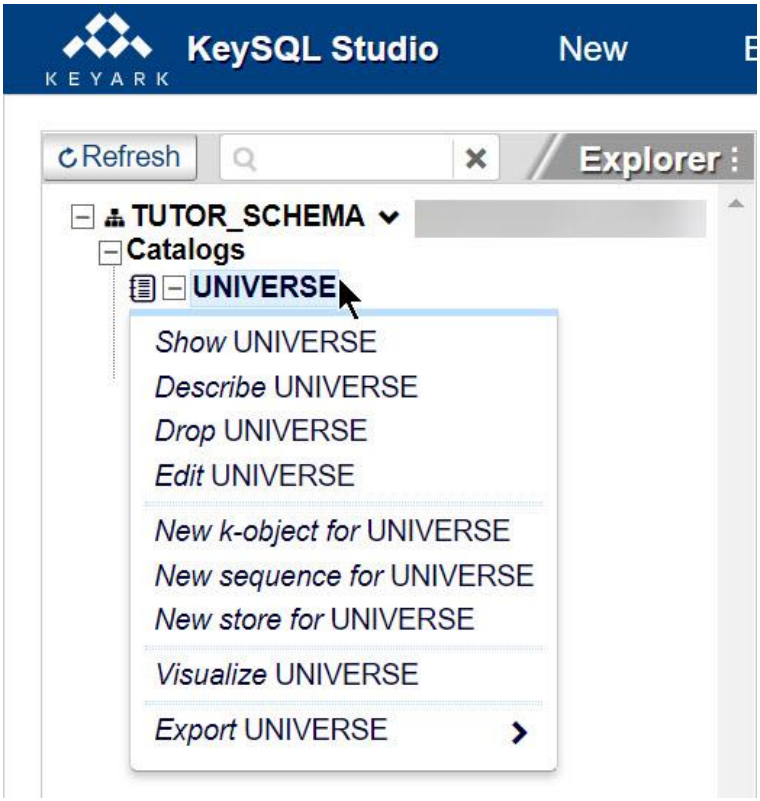
Upon login, KeySQL Studio will initially show the *default* schema for that user. That's the location where Studio will assume you will store data and look for data. In the screenshot above, the default schema for user TUTOR is TUTOR_SCHEMA, and the sole catalog UNIVERSE in that schema is listed.

A user may have permissions to schemas beyond just their default schema. To see the list of accessible schemas, click the ⌄ that's visible at the tip of the cursor, for which the tooltip states *Change KeySQL schema*. Clicking here will display a list of accessible schemas in alphabetical order. To switch to a schema, click its name. Explorer will automatically refresh, showing the catalogs in that schema.



1.2.1.2 WORKING WITH CATALOGS

Performing a regular click (left-click) on a catalog presents a menu of catalog manipulation statements:



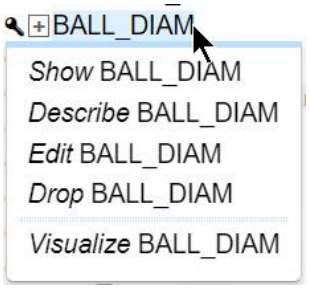
FYI: KeySQL Studio displays only context menu options that are allowed for the selected k-object. For example, you will see Drop when a catalog or store is selected, as you are allowed to drop an entire catalog or store. However, if you select a k-object within a catalog, for example ITEM_ID, you will see the Drop menu option for ITEM_ID only if ITEM_ID is stand-alone, meaning not a constituent member of another k-object.

Quick explanation for the catalog manipulation statements:

Menu Command	Purpose	KeySQL Statements Generated	More Detail
Show <catalog name>	Provides a complete description of the catalog k-objects and list of stores	**SHOW CATALOG** <catalog name>;	Perform 'Execute' to see the statement result.
Describe <catalog name>	Provides a script for recreating the catalog k-objects and list of stores	**DESCRIBE CATALOG** <catalog name>;	Perform 'Execute' to see the statement result.
Drop <catalog name>	Drop the catalog	**DROP CATALOG** <catalog name>;	Prompts you to drop all stores in that catalog before it allows for dropping the catalog.
Copy <catalog name>	Copy the catalog, assigning a different name; optionally drop or change k-object data type. A retractable Copy catalog pane opens.	**CREATE CATALOG** <catalog new_name>; **CREATE KEYOBJECT** <k-object name> **IN CATALOG** <catalog new_name>;	Copies the catalog k-objects. CREATE statements generated when 'Save' pressed. Perform 'Execute' to execute the statements and see the statement result.
Edit <catalog name>	Edit the k-object structure of the catalog. Also rename the catalog. A retractable Edit catalog pane opens. Typical operations are add, drop or change k-objects, as well as change data type.	**ALTER KEYOBJECT** <k-object name>=<k-object name>**{** new structure **}** **IN CATALOG** <catalog name>; **RENAME CATALOG** <catalog current_name> **TO** <catalog new_name>;	ALTER and RENAME statements generated when 'Save' pressed. Perform 'Execute' to execute the statements and see the statement result.
New K-object for <catalog name>	Create a new k-object for the catalog. A retractable New k-object pane opens.	**CREATE KEYOBJECT**<new k-object name> <k-object type> **IN CATALOG** <catalog name>;	CREATE statements generate if 'Save' pressed. Perform 'Execute' to see the statement result.
New Sequence for <catalog name>	Create a new sequence for the catalog. A retractable New sequence pane opens.	**CREATE SEQUENCE** <new sequence name> **IN CATALOG** <catalog name>;	CREATE statements generate if 'Save' pressed. Perform 'Execute' to see the statement result. For an explanation of sequences, please see the KeySQL Definitive Guide.

New Store for <catalog name>	Create a new store for this catalog. A retractable New Store pane opens.	**CREATE STORE** <new store name> **FOR CATALOG** <catalog name>;	CREATE statements generate if 'Save' pressed. Perform 'Execute' to execute the statements and see the statement result.
Visualize <catalog name>	Launch the Data Modeling Tool, ready to display the current catalog		See the *KeySQL Data Modeling Tool User Guide* for further information.
Export <catalog name>	**K-objects:** Export k-objects for the catalog to a file in the form of ready to execute KeySQL statements **Stores:** Export Stores for the catalog to a file in the form of ready to execute KeySQL statements **Entire design:** Export the entire design including k-objects, stores, sequences, and constraints in the form of ready to execute KeySQL statements		Depending on your Browser setting, either a standard dialog will pop-up, letting you pick the file name, or the file will be immediately written to your download folder with a default filename.

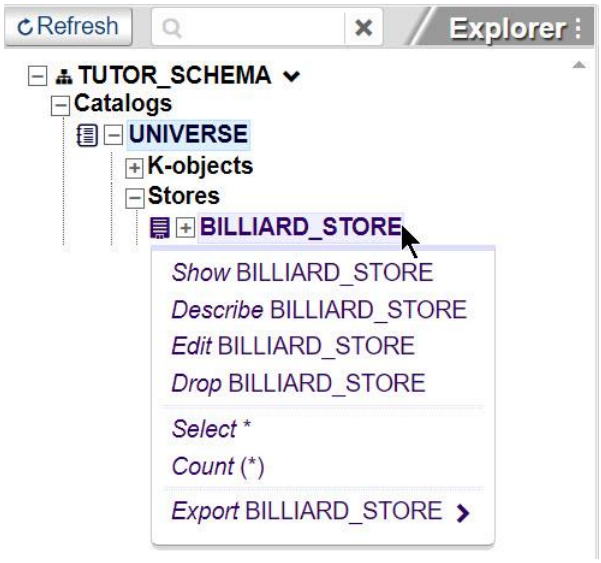
Notice how left click on a k-object within the catalog displays a menu of valid options:



This is explained below in 1.2.1.4 WORKING WITH SPECIFIC K-OBJECTS SHOWN FOR THE CATALOG.

1.2.1.3 WORKING WITH STORES

Performing a click on a store presents a menu of store manipulation statements:



Quick explanation for the store manipulation statements

Menu Command	**Purpose**	**KeySQL Statements Generated**	**More Detail**
Show <store name>	Show k-objects for the store	**SHOW STORE** <store name>;	Perform 'Execute' to see the statement result
Describe <store name>	Script for recreating k-objects for the store	**DESCRIBE STORE** <store name>;	Perform 'Execute' to see the statement result
Edit <store name>	Rename the store A Edit store pane opens.	**RENAME STORE** <current store name> **TO** <new store name>;	Perform 'Execute' to see the statement result
Drop <store name>	Drop the store	**DROP STORE** <store name>;	All data within that store will be permanently deleted without warning after 'Execute' is performed

Select *	Display data within that store	**SELECT * FROM** <store name> **LIMIT 100**;	Perform 'Execute' to see the statement result
Count (*)	Give count of instances within that store	**SELECT COUNT(*) FROM** <store name>;	Perform 'Execute' to see the statement result
Export <store name>	Export all the store data to a file in form of ready to execute INSERT KeySQL statement, JSON data or CSV data.		Depending on your Browser setting, either a standard dialog will pop-up, letting you pick the file name, or the file will be immediately written to your download folder with a default filename.

The Export <store name> menu may show up to three choices as shown here:

As shown above, the HOUSEHOLD_FINANCE store contains data that can be easily shared as rows, like what you would see in Excel, hence all three export format options are available.

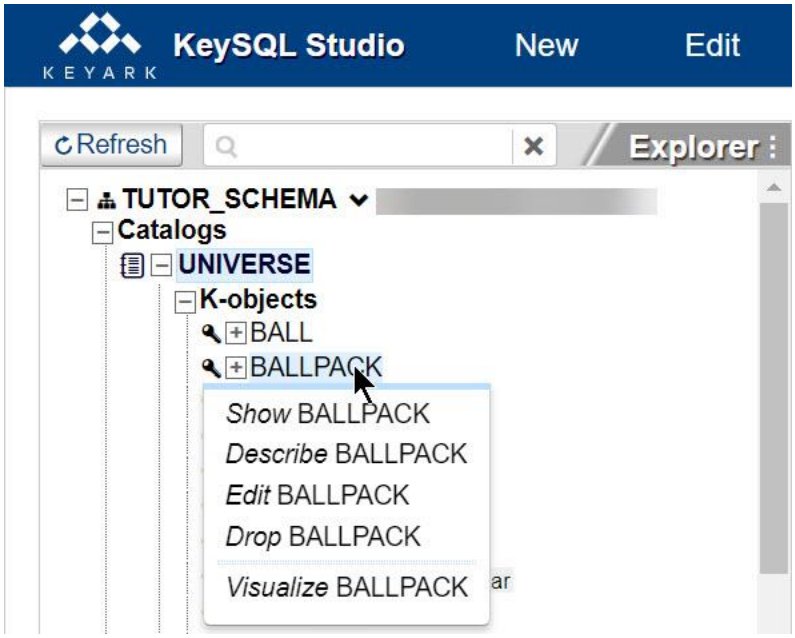
In contrast, BILLIARD_STORE contains more complex multi-composite data, where there can be multiple values such as for phone, where the instance for one person lacks a phone number, but another has both a business and home phone number. Such multi-composite data cannot be easily represented as rows. Accordingly, only the KeySQL and JSON format options are available:

In Explorer you can see the k-objects specific to that store, for which data exists, such as BALLPACK in the screenshot below. Click on that k-object to display a menu of options for that k-object:

This is explained below in 1.2.1.5 WORKING WITH SPECIFIC K-OBJECTS SHOWN FOR THE STORE.

1.2.1.4 WORKING WITH SPECIFIC K-OBJECTS SHOWN FOR THE CATALOG

Click of a k-object within the catalog displays a menu of options for working with that k-object:



Quick explanation of the k-object manipulation statements:

Menu Command	Purpose	KeySQL Statements Generated	More Detail
Show	Shows details for this k-object.	**SHOW KEYOBJECT** <k-object> **FROM CATALOG** <catalog name>;	
Describe	Shows script for recreating this k-object.	**DESCRIBE KEYOBJECT** <k-object> **FROM CATALOG** <catalog name>;	
Edit	Creates script for altering this k-object, such as renaming or, for composite k-object,	**ALTER KEYOBJECT** … **IN CATALOG** <catalog_name>;	The Edit panel appears, which generates the appropriate statements.

	changing the list of constituent k-objects.		
Drop	Drop the k-object.	**DROP KEYOBJECT** <k-object name> **FROM CATALOG** <catalog name>;	This is not allowed if this k-object exists in any of the stores.
Visualize	Launch the Data Modeling Tool, ready to display the selected k-object		See the *KeySQL Data Modeling Tool User Guide* for further information.

1.2.1.5 WORKING WITH SPECIFIC K-OBJECTS SHOWN FOR THE STORE

You can drag-and-drop both the store and its constituent k-objects to the Query Tab, perhaps to create a SELECT statement.

Click of a k-object within the store displays a menu of options, which include both **compose** and **add to**, which assist you in constructing a SELECT. In addition, **Export** does precisely that, exports data for this k-object to a file.

Quick explanation for the store commands:

Menu Command	Purpose	KeySQL Statements Generated	More Detail

compose	Display the specific k-object data within that store	**SELECT** <k-object> **FROM** <store name> **LIMIT 100;**	Create or update a SELECT in the current Query section, replacing any commands already there.
add to	Additionally display this k-object data	K-object is added to the most recently generated SELECT.	If the current tab lacks a SELECT, the *add to* command behaves the same as *compose.*
Export <k-object name>	Export the k-object data to a file.	The file will contain ready to execute INSERT KeySQL statements or JSON data or CSV data, depending on the data complexity and your selection of format.	Depending on your Browser setting, either a standard dialog will pop-up, letting you pick the file name, or the file will be immediately written to your download folder with a default filename.

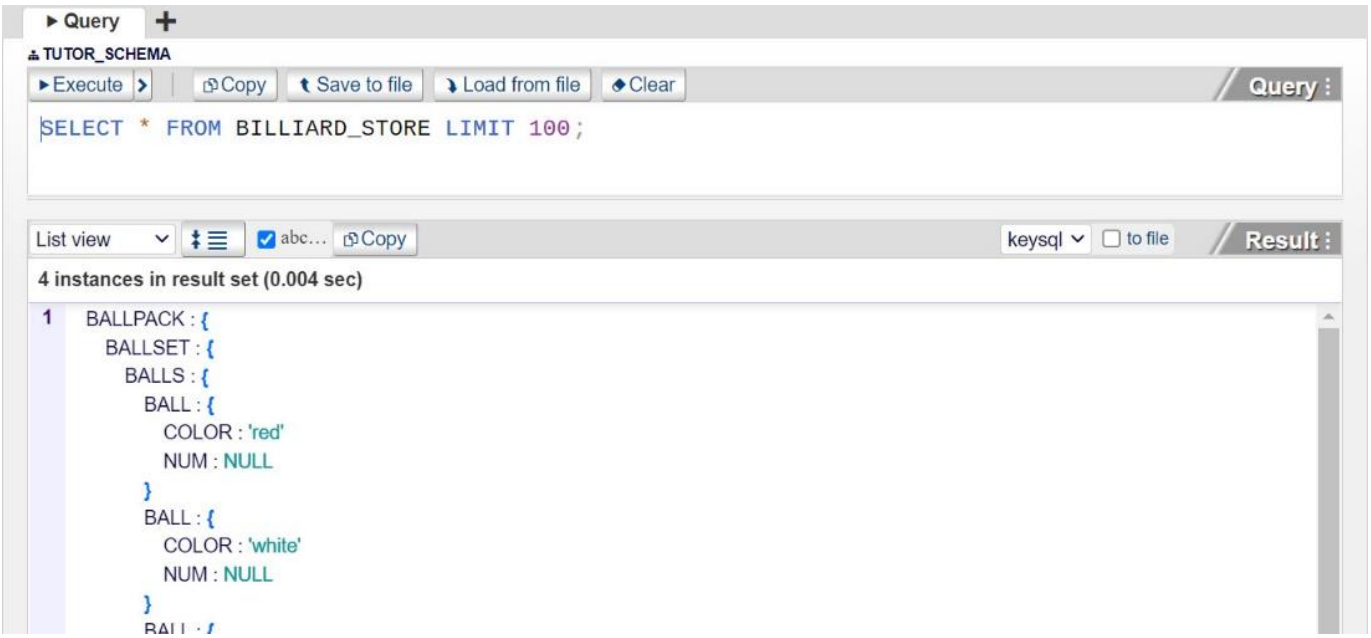
FYI: These two menu commands are exceptional in that they operate *only* on the most recently created tab, the one on the far left labeled "Query". However, you can accomplish the same task by using drag-and-drop to bring in k-objects from any store, dragging them to any tab.

1.2.2 Execution Pane

Quick explanation of the Execution pane layout and purpose

Pane		Section	Purpose	More Detail
Execution pane	Query Tabs	Query Statement	Enter KeySQL statements to view and manipulate your data	Select the Execute button to run KeySQL statements. Select the Save to file button to save the statements shown in your Query pane. Select the Load from file button to restore instructions from a file.
		Query Result	Displays the result of the KeySQL statement executed in the Execution pane	The View dropdown lets you select among three different presentations for your query results.

The Execution Pane is designed to contain unlimited tabs. The + button to the right of the rightmost tab, when clicked creates a new tab. Each tab is divided into Query and Result sections:



When you perform a SELECT, your data will appear in List view, as shown above. Use the scroll bars to skim through your data.

FYI: In the screenshot above the KeySQL SELECT assumes the default schema, TUTOR_SCHEMA, as indicated by TUTOR_SCHEMA at the top left of the Query area. You can add a schema qualification, to specify a store in another schema. For example, the KeySQL statement

SELECT * FROM **PUBLIC**.BILLIARD_STORE

directs KeySQL Server to SELECT from another BILLIARD_STORE, one found in the PUBLIC schema.

1.2.2.1 TAB QUERY SECTION

The most important button is the Execute button which will run a query, sending output to the Result section.

Quick explanation for Query section icons and purpose:

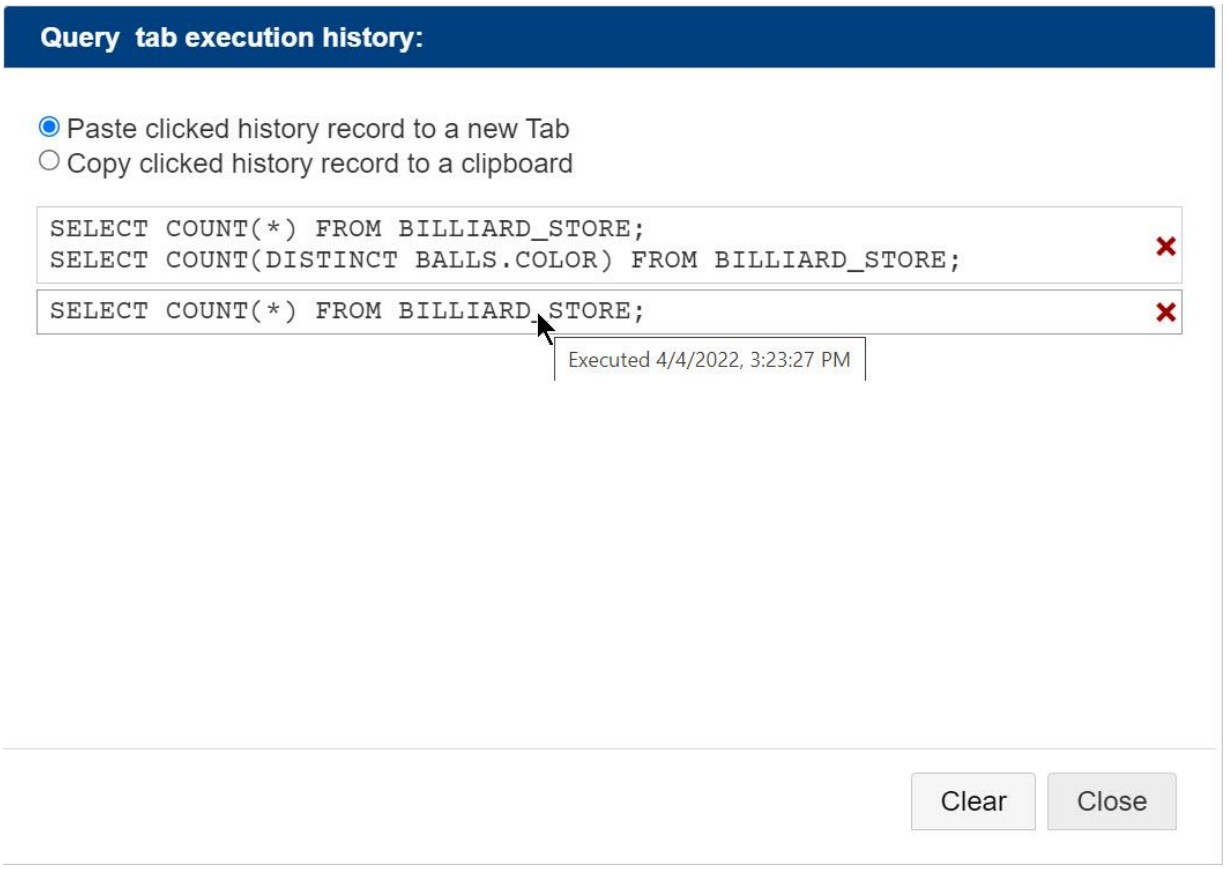
Icon	Button	Studio Label	Purpose	More Detail

▶	Execute	Execute queries	Execute statements in the Query section of the Execution pane	Run all the statements. To run specific statements, select those lines with your mouse.
›		Queries, executed in this tab	Display the query tab execution history to review/rerun prior statements	When selected, a new tab is opened and the statements are pasted into the Query section, ready for you to execute.
	Copy	Empty query area	Copy statements in the Query section to the clipboard	
	Save to file	Save to text file	Save statements currently in the Query section to a text	
	Load from file	Load from text file	Intended for loading prepared statements from a text file into the Query section	Should the content be too big to display, a dialogue will appear that will ask if it's okay to "Proceed without preview?" Click **Yes** to execute these commands without being able to inspect.
	Clear	*none*	Clear both the Query section and the Result section	

EXECUTE FROM HISTORY

The screenshot below depicts using your cursor to click the **Execute History** button, the tiny › to the right of **Execute** button:

When **Execute History** is clicked, a dialogue window shows the history of KeySQL statements that have been executed in that tab. The most recent statement is at the top. Placing the cursor over a statement displays a tooltip with the datetime it was executed:



When you click a record, the statements you see in the record will be copied to a new Tab.

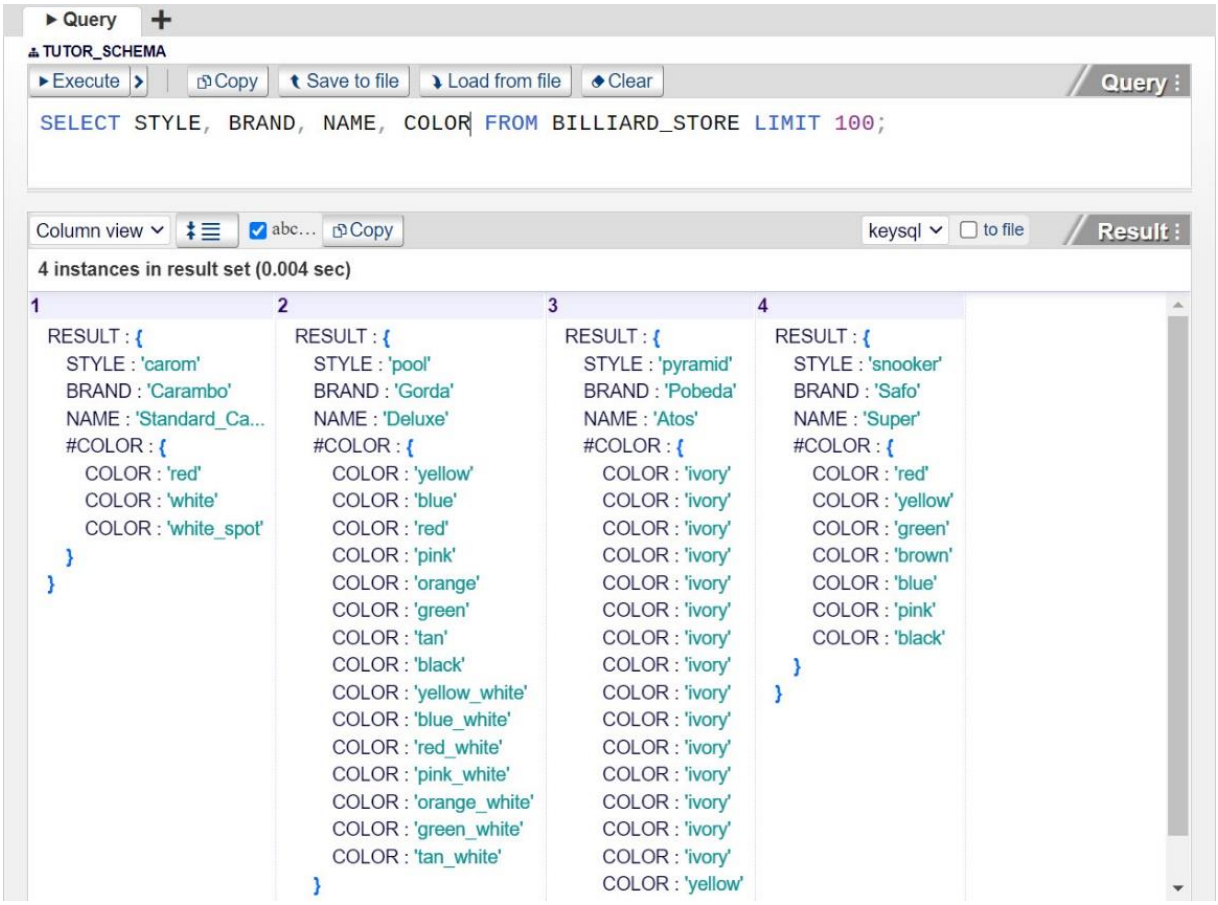
Alternatively, to paste to another destination, select the radio button **Copy clicked history record to clipboard** before you click.

The **Clear** button will erase this history.

The **Close** button will take you back to your Tab, without any change.

1.2.2.2 TAB RESULT SECTION

The screenshot below shows the Column view where each instance appears in its own column:



This screenshot gives a glimpse of the power of KeySQL. Each instance of BALLPACK in BILLIARD_STORE can have multiple instances of BALL, where each BALL has its own COLOR. KeySQL Studio concisely displays this multi-composition.

FYI: The KeySQL data model and structured query language is fully described in the companion document **KeySQL Definitive Guide**.

CATALOG OF THE USER INTERFACE CONTROLS

It is evident in the screenshot above that the Results area contains numerous user interface controls, including buttons, checkboxes, and dropdowns. The two tables that follow provide a succinct summary of their purpose. This is followed by sections with additional explanations.

TABLE OF BUTTONS

Button	Studio Label	Purpose	More Detail

Expand (Collapse) 		Expand instance to show detail, or collapse to show more instances	Button toggles to show instance detail. Leave collapsed for faster scrolling.
Copy 	Copy the Result to Clipboard	Copy the content of the Result area to the clipboard. If the size is too large for the clipboard, KeySQL Studio will save to a text file instead.	When saving to a text file, the behavior depends on your Browser setting. Either a standard dialog will pop-up, letting you pick the file name, or the file will be immediately written to your download folder with a default filename. The filename will be like keysql_Query_result.txt , keysql_Query_result (1).txt etc.

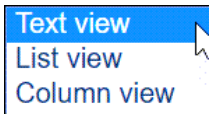
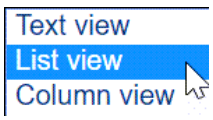
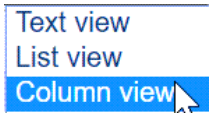
TABLE OF CHECKBOXES & DROPDOWN

	Studio Label	Purpose	More Detail
Truncate values 	Truncate long values with ellipses	By default, truncate value of atomic k-object with ellipsis in 'List' and 'Column' result view.	Show the whole value on mouse-over; truncate/expand the value on mouse-click.
Output Data format 	Select data format	Choose the data format, similar to store export: either KeySQL, CSV, JSON or XML.	The same choices are available for catalog and store export.
To file 	Output the execution result to a local file	When checked, clicking the Execute button will output to a text file rather than display results. The Result area shows the name of the file created.	If checked an execution result of *any* request (SELECT, CREATE, SHOW, ALTER, INSERT, SHOW) delivers output to a file rather than to the Result area.
Pretty Print 	Output the result to a local file in	When either KeySQL or JSON is written to a disk file, a CR & LF character combination separates the	The pretty print checkbox appears only when the **to file** checkbox is selected.

	a pretty format	instances. When pretty print is selected, additional CR & LF character combinations are written, so that output is easy to read, like what you normally see in the Results area with List view and output expanded.	
Output Refresh 	Refresh result	Refresh the presentation format for results from your last command.	Button appears when you change your preferred format (keysql/csv/json/xml) or output method (to screen/to file) and disappears once the results appear in the new format.

DROPDOWN VIEWS

There are three available views for the query output.

View	Purpose	More Detail
Text view 	Shows k-object instances in a compact JSON-like format	Should the query result prove too large for the List view, Studio automatically switches to the Text view Should scrolling prove slow in List view or Column view, switch to Text view for faster response.
List view 	Shows enumerated instances and their k-objects	Default view; click on a blue brace to collapse/expand the enclosed value
Column view 	Focus on individual instances	Click on a blue brace to collapse/expand the enclosed value

EXPANDING AND CONTRACTING INDIVIDUAL K-OBJECT

Note how the BILLIARD_SCREEN screenshot shows the first four instances in the BILLIARD_STORE, but with increasing level of detail. The Result pane initially displays your data with a fully expanded k-object hierarchy level detail, as seen in the 4th instance. However, you have full control over which detail should be displayed.

Display Control	**Purpose**

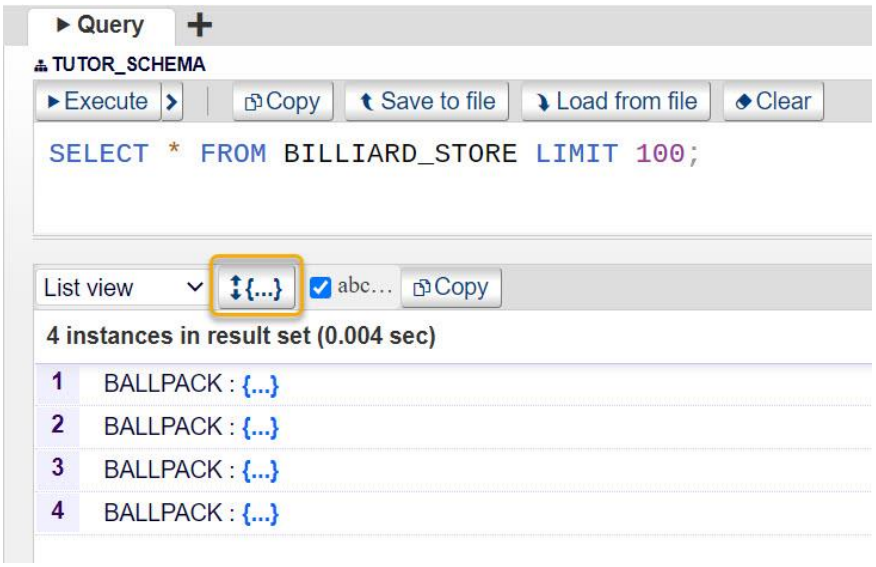
Compress
{
}
	Click either semicolon to compress the k-object

Expand {...}	Click the ellipsis to expand the k-object

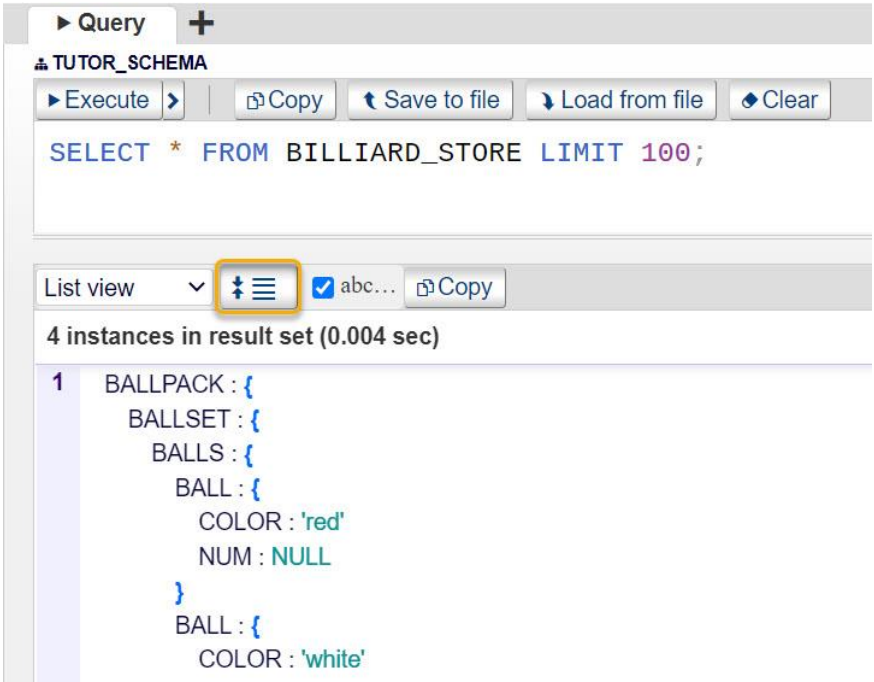
This technique works with the Column and List view only.

EXPANDING AND CONTRACTING ALL INSTANCES

Immediately to the right of the View dropdown, there is an **Expand** button. This allows one to toggle how instances are displayed between fully expanded and fully contracted. The screenshot below shows instances fully contracted:



The screenshot below shows the Results area with the output expanded. Note how the button label toggles to reflect what state will be shown once it is clicked. Clicking it now would collapse the output:



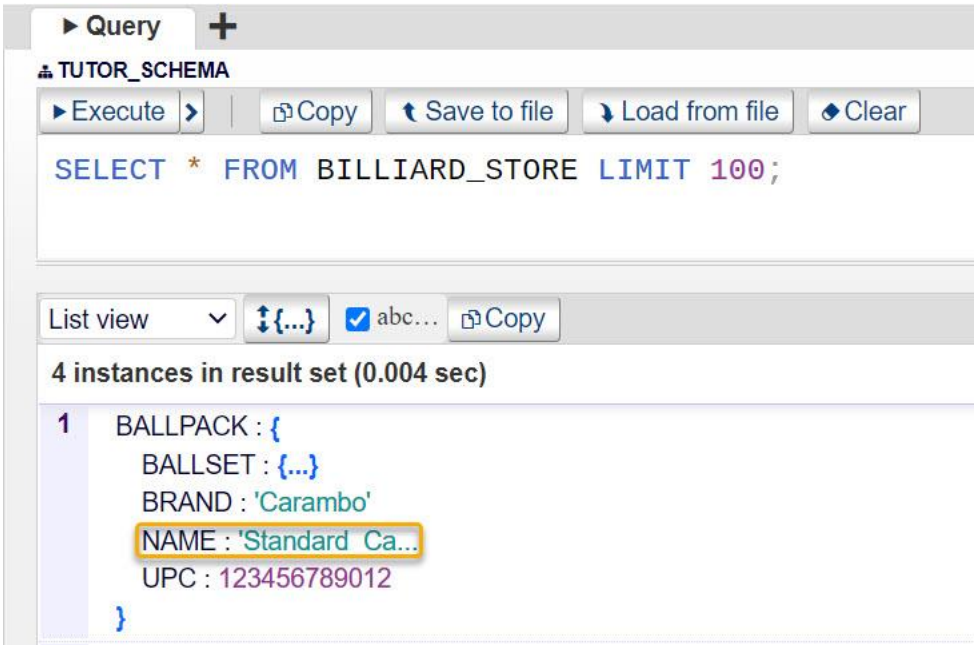
This button is enabled in the Column and List view only, and for smaller result sets only.

TRUNCATING VALUE DISPLAY

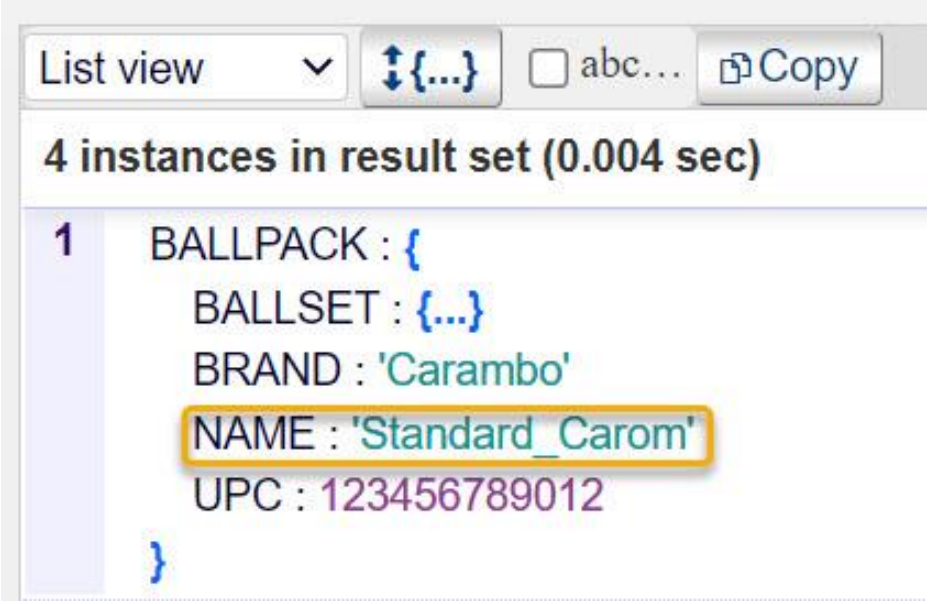
Many data sets include long text values, which unnecessarily fill the Result pane. The Result pane has a truncate values checkbox (abc…), highlighted in the screenshot below. This is checked by default.

In the screenshot below, note the ellipsis that indicates that the data display is truncated in display to just

"Standard Ca…":



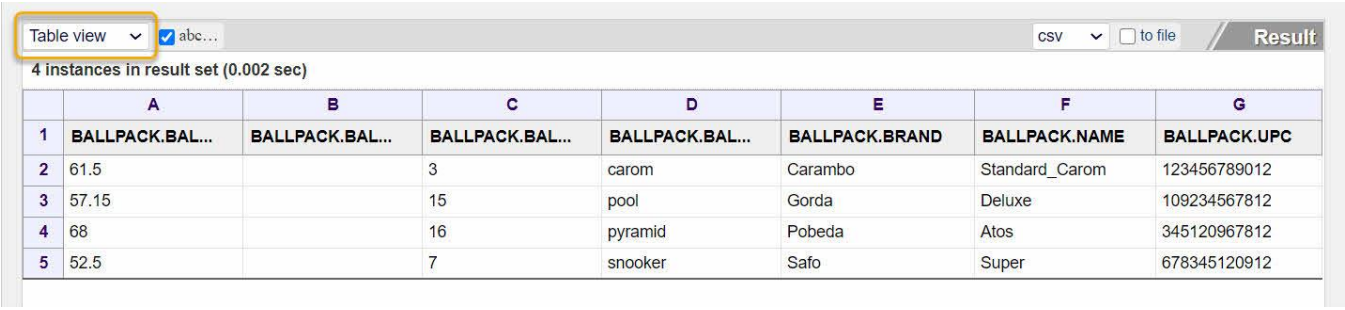
To show the full value, uncheck the **abc…** truncate values checkbox either before or after your query execution. Now that the ellipsis is gone, and the full value "Standard_Carom" is shown:



You can fully reveal an individual value by clicking on that value's ellipsis.

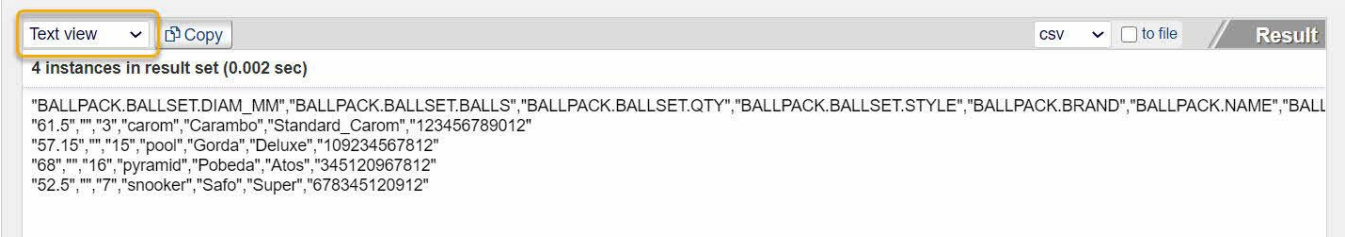
VIEW CHANGE FOR CSV

When CSV is selected from the Data Format dropdown, the look of the Results area changes dramatically, as the View dropdown shows two choices. The Table view, which resembles the presentation of spreadsheet data, is specific to CSV, and is shown below for SELECT * FROM BILLIARD_STORE LIMIT 100:



	A	B	C	D	E	F	G
1	BALLPACK.BAL...	BALLPACK.BAL...	BALLPACK.BAL...	BALLPACK.BAL...	BALLPACK.BRAND	BALLPACK.NAME	BALLPACK.UPC
2	61.5		3	carom	Carambo	Standard_Carom	123456789012
3	57.15		15	pool	Gorda	Deluxe	109234567812
4	68		16	pyramid	Pobeda	Atos	345120967812
5	52.5		7	snooker	Safo	Super	678345120912

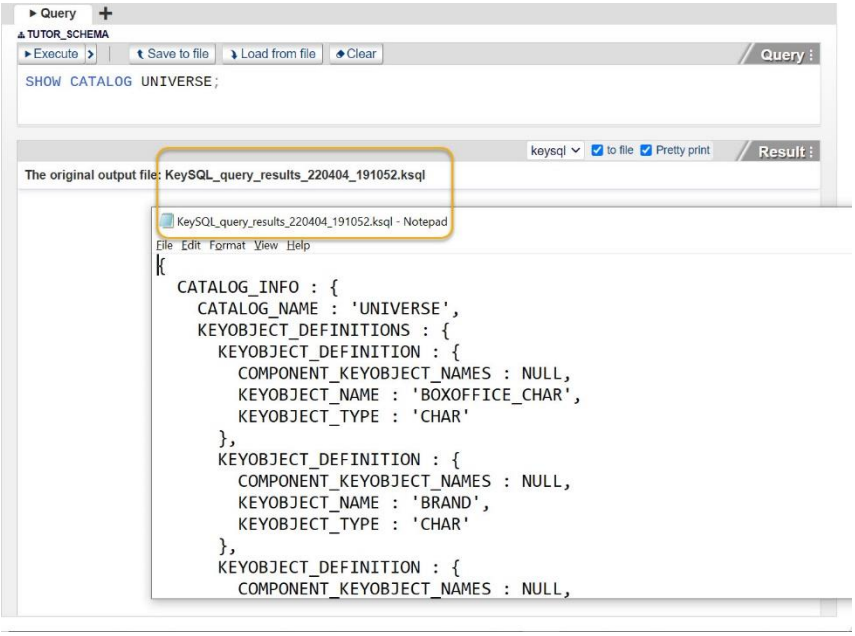
You can toggle from Table view to Text view and back, where Text view displays a CSV file as it would be seen in a text editor:



Note how KeySQL Studio has "flattened" the non-flat **Billiard_Store** data. The column names are a concatenation, joined by a period, of the element names when descending through the complex data structure. As there is no clean way to flatten multiple value data (e.g. ballpack.ballset.balls) into a two-dimensional (column and row) spreadsheet format, if your store contains multi-composite k-object data, the CSV output will display a column which is empty.

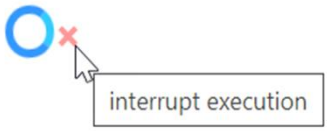
TO FILE CHECKBOX

The screenshot below demonstrates the effect of the **to file** checkbox. Once the KeySQL statement SHOW CATALOG UNIVERSE is executed, the output is written directly to a text file. Rather than show the output in the Result area, KeySQL shows only the output filename. The Notepad text editor shows the file content:



1.2.2.3 QUERY CANCELLATION

If you need to interrupt a long-running query or import, you can click the red X mark adjacent to the spinner:



1.2.2.4 DRAG AND DROP

Several KeySQL Studio panes, including Explorer, support drag and drop. This is most useful in two situations:

- Composing a SELECT statement
- Altering a catalog, creating or altering composite k-object

DRAGGING A K-OBJECT FROM AN EXPLORER TO A SELECT STATEMENT

The Catalog name, Store name, k-objects can be dragged to the Query text area. This is most useful for building SELECT statements. Type SELECT, then drag the list of k-objects, then type FROM and drag the store name.

DRAGGING A K-OBJECT TO ALTER A CATALOG OR COMPOSITE K-OBJECT

This technique applies to these panes:

- Edit Catalog
- Edit K-object
- New Catalog

1.2.3 Session Save and Restore

You can easily save and restore your KeySQL Studio tab session by taking advantage of the Tabs menu commands:

Tabs Menu	Purpose	Details
Save active tab	Saves the active Query tab session, both the Execution and Result pane.	Saves to a text file with a default filename of tab_Query.tab
Restore saved tab	Restores to the current tab a previously saved tab session	Load from a text file with a default filename of tab_Query.tab

1.2.4 Customizing Studio Appearance

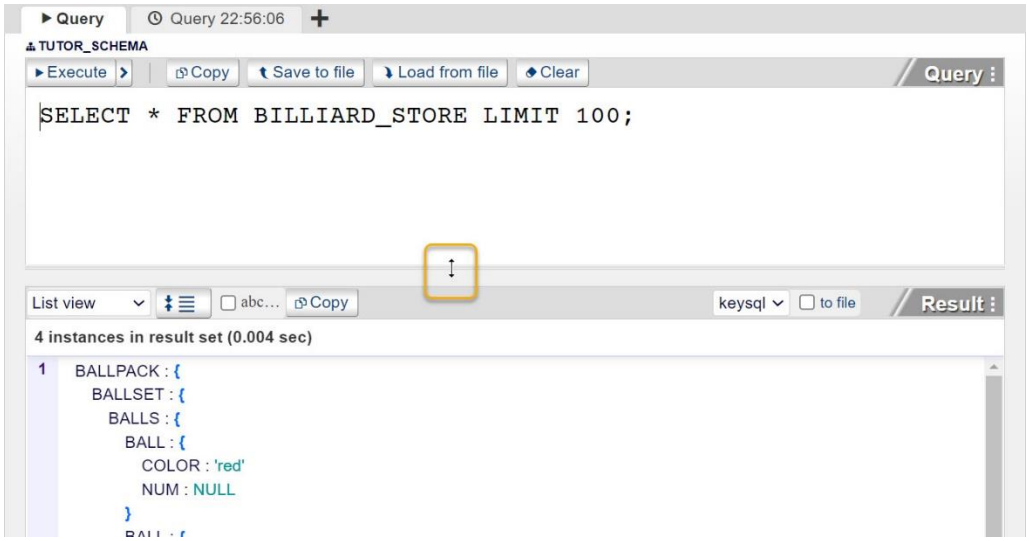
You can adjust the panes in several ways:

- relative pane size
- font size
- syntax color
- font

In addition, you can control the refresh behavior of Explorer.

1.2.4.1 RELATIVE PANE SIZE

To adjust the relative size of adjacent panes, click on the border and drag. Note how the cursor, positioned on the border of the Query and Result panes, has changed shape, ready for you to drag:



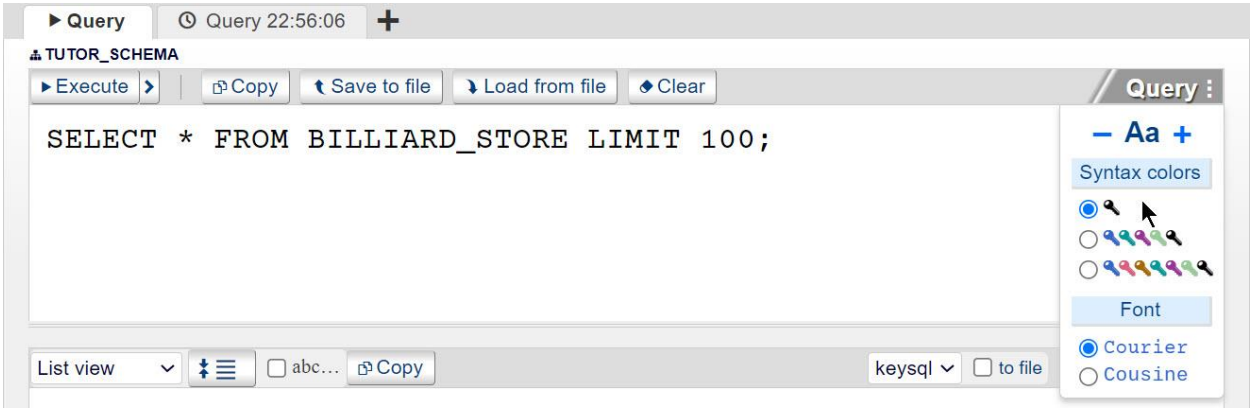
1.2.4.2 FONT SIZE

To adjust the font size, click on the pane title to display the pane customization control. In the screenshot below for the **Query** pane title, the revealed + button was clicked repeatedly.

FYI: The screenshot above illustrates Query pane with two tabs. The most recently created tab is labeled "Query". The prior "Query" tab is pushed to the right, and given a timestamp "Query HH:MM:SS" for when this happened. The label presents the hour, minute, and second in 24-hour notation. In the example above, the farthest right tab was pushed to the right at 10:56:06 PM.

1.2.4.3 SYNTAX COLORS

Studio automatically applies coloration to KeySQL commands to make the syntax easier to read. You can choose to use a simpler color palette, or just black and white as shown in this screenshot:

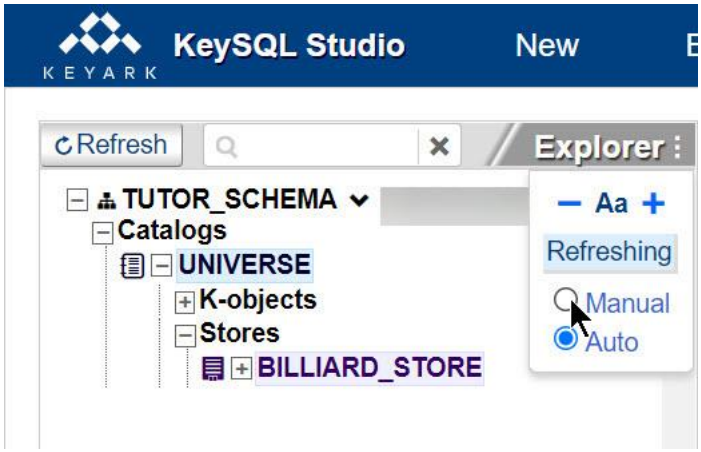


1.2.4.4 FONT STYLE

The default font style is Courier, but you can change to Cousine which is an innovative, refreshing sans serif design that is metrically compatible with Courier New™:

1.2.5 Explorer Refresh

The Explorer pane has a unique customization: Auto vs Manual refreshing. The default setting is Auto, which causes the Explorer object tree to automatically refresh when an object is inserted, renamed, or deleted. In some situations, when a lot of changes are underway, the automatic refresh can slow down your work, and you can select Manual refresh. To force a refresh, click the Refresh button.



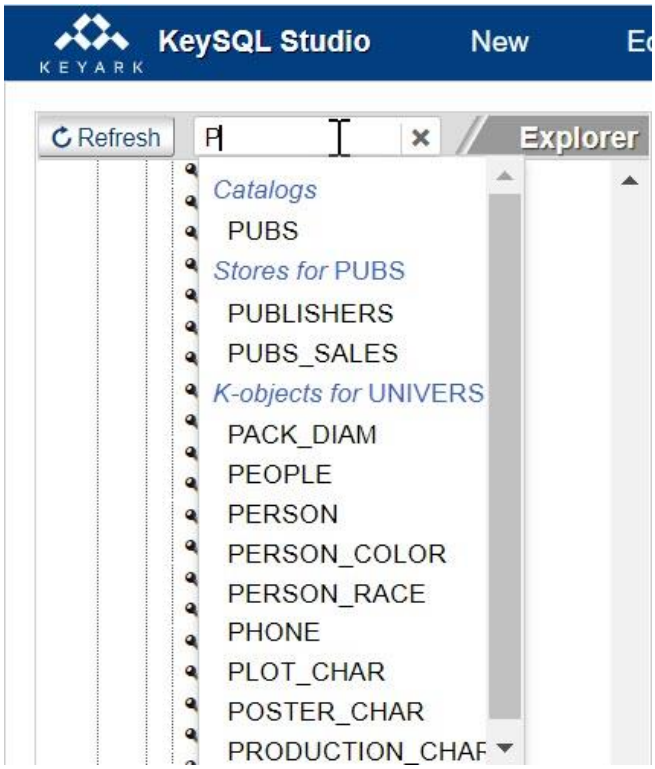
1.3 SEARCH

The Search box provides an exhaustive, instant search for

- catalogs within the schema
- stores within each catalog
- k-object within each store

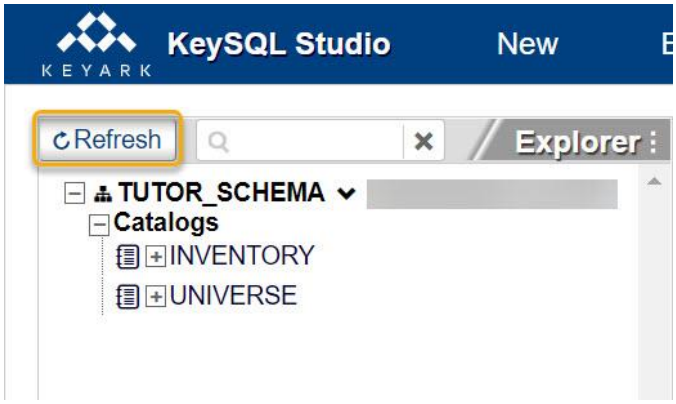
As you type a search term, the list of possible matches is narrowed.

Here you can see all the catalogs, stores and k-objects that begin with the letter “P”. You can narrow the list by typing additional characters, or simply click an item to jump there:



1.4 REFRESH

The **Refresh** button at the top left of the Explorer pane serves to fetch fresh information from KeySQL Server. When you load KeySQL Studio, the Explorer tree is populated with a list of catalogs and stores provided by KeySQL Server.



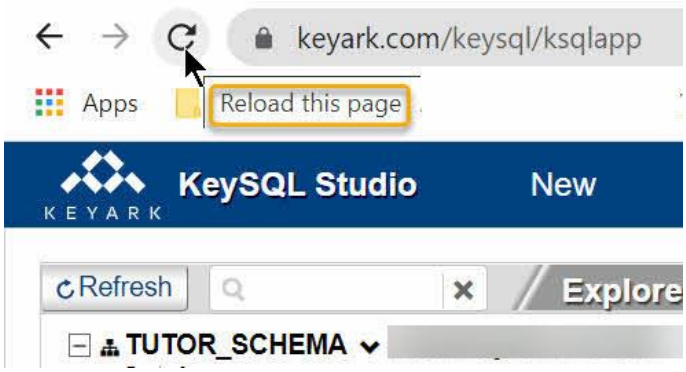
In the course of your work, you probably do not need to **Refresh**. However, if you have coworkers working on the same catalog, you may want to perform a Refresh to see their changes.

1.4.1 Fresh Instance of KeySQL Studio

After about 30 minutes of no activity, KeySQL Studio will time-out as a security measure. This will necessitate either:

- Clicking the reload button, or
- Opening a new tab, and re-entering the URL

The screenshot below shows the reload button specific to Google Chrome:



1.4.1.1 CHECKING VERSION

To ensure your browser is using the very latest version of KeySQL Studio, do a reload as described above.

Click on the KeySQL Studio label to display the current build date and version number.

1.4.2 Resolving Issues by Clearing the Cache

Clearing your cache and doing a hard (forced) reload is often an easy way to resolve issues, and ensures that you have the most recent upgrade of KeySQL Studio, including Javascript libraries that it uses. Section “Checking Version” immediately above shows how to check.

To do a hard refresh of your KeySQL Studio browser page there is a special key combination:

- For most browsers running on Windows press Ctrl + Fn + F5
- For browsers running on the Mac, and for a few Windows browser, there are similar key combinations as described in the article below:

https://support.planwithvoyant.com/hc/en-us/articles/360046611171-How-to-do-hard-refresh-in-Chrome-Firefox-Safari-and-Microsoft-Edge

2 STUDIO MENUS

This section starts with the basics of running KeySQL Studio:

- Starting KeySQL Studio
- Quitting KeySQL Studio

Next a quick introduction to working with catalogs and stores is provided, accessing this capability through the menu system:

- New Catalogs and Stores
- Edit Catalogs and Stores

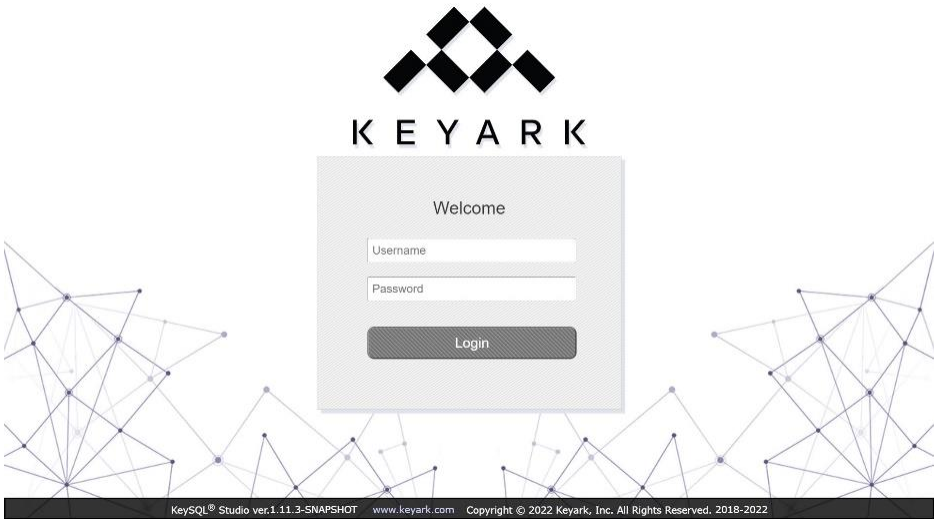
Finally, you are directed to sections of this guide that address KeySQL Studio Tools for import and export.

2.1 RUNNING KEYSQL STUDIO

The process for starting and quitting KeySQL Studio is similar to other web applications.

2.1.1 STARTING KEYSQL STUDIO

Nothing needs to be installed to run Studio. To start you only need the URL of your KeySQL Server, which is both a database server and a web server:

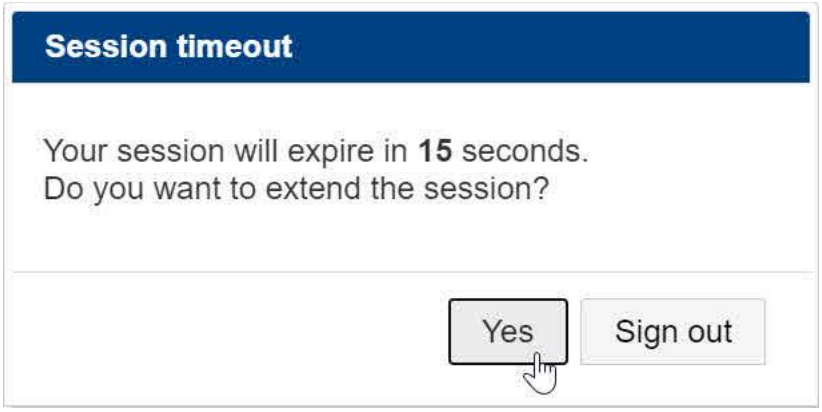


2.1.2 RECONNECTING KEYSQL STUDIO

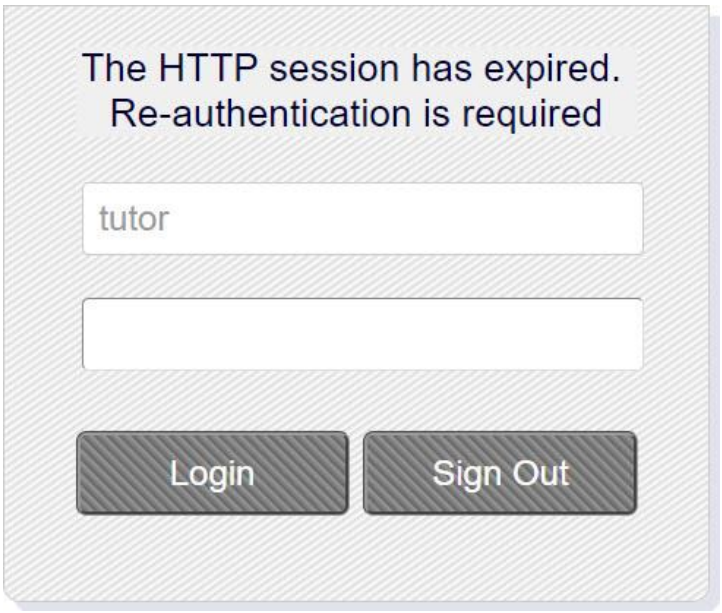
To ensure security of data on KeySQL Server, if there is no activity in KeySQL Studio for 30 minutes, KeySQL Studio will time-out.

2.1.2.1 SESSION TIMEOUT

Just before the timeout, you will be prompted to extend the session, which avoids signing in again:

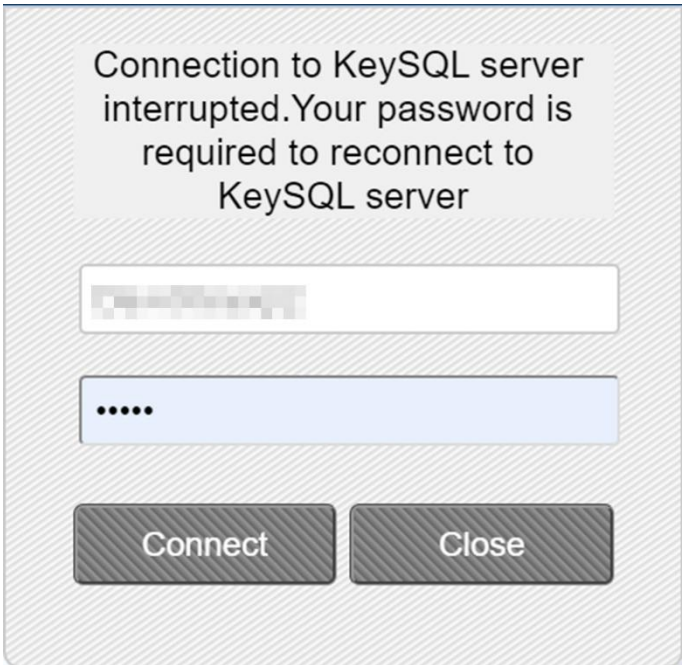


If you don't respond quickly enough, you will be prompted to login again, or to sign out:



2.1.2.2 NETWORK CONNECTION INTERRUPTION

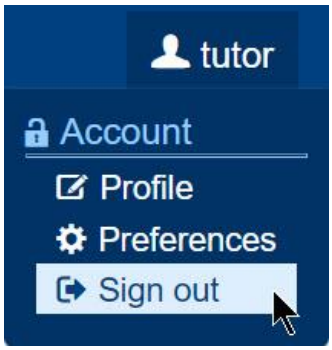
When there is an interruption in your network connection, perhaps due to a glitch in your WiFi, you will be advised of the situation and prompted to log-in again:



Enter your Username and Password and click Connect.

2.1.3 SIGN OUT

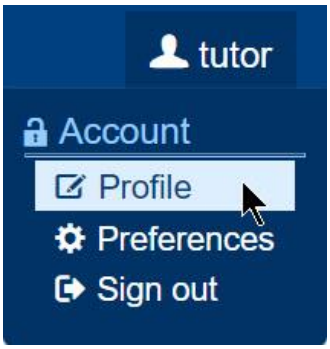
You can either close the browser tab, or click your login name to reveal Sign out:



2.1.4 PROFILE

The Profile menu option takes you to a form with two tabs:

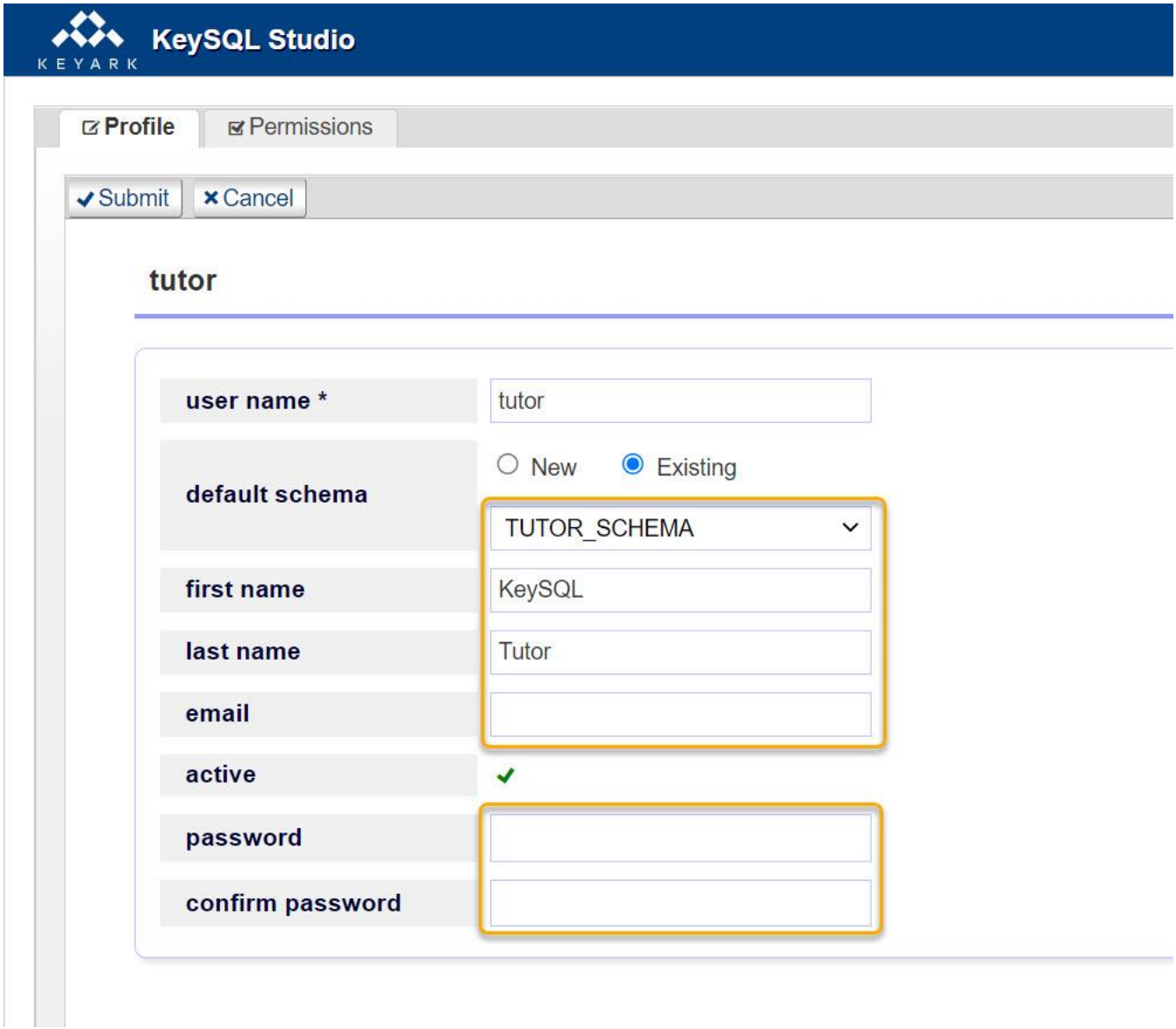
- Profile
- Permissions



Full detail is provided in Chapter 5, a chapter intended for administrators. An abbreviated explanation for non-administrative users is provided here.

2.1.4.1 PROFILE INCLUDING PASSWORD

The Profile tab displays your **user name**, **default schema**, **first name** and **last name**, and **email** address and more.



The fields that can be updated are denoted by gold boxes.

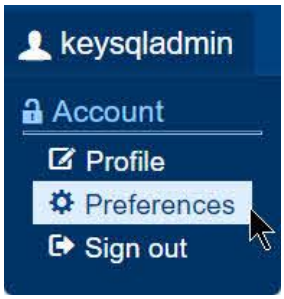
Note that you use this form to update your password. There is no need to enter your current password; enter your desired new password twice.

2.1.4.2 PERMISSIONS

The Permissions tab shows the permissions provided to you by an administrator for access to schemas, catalogs in those schemas, and stores within those catalogs. You can see but not change these settings.

2.1.5 PREFERENCE

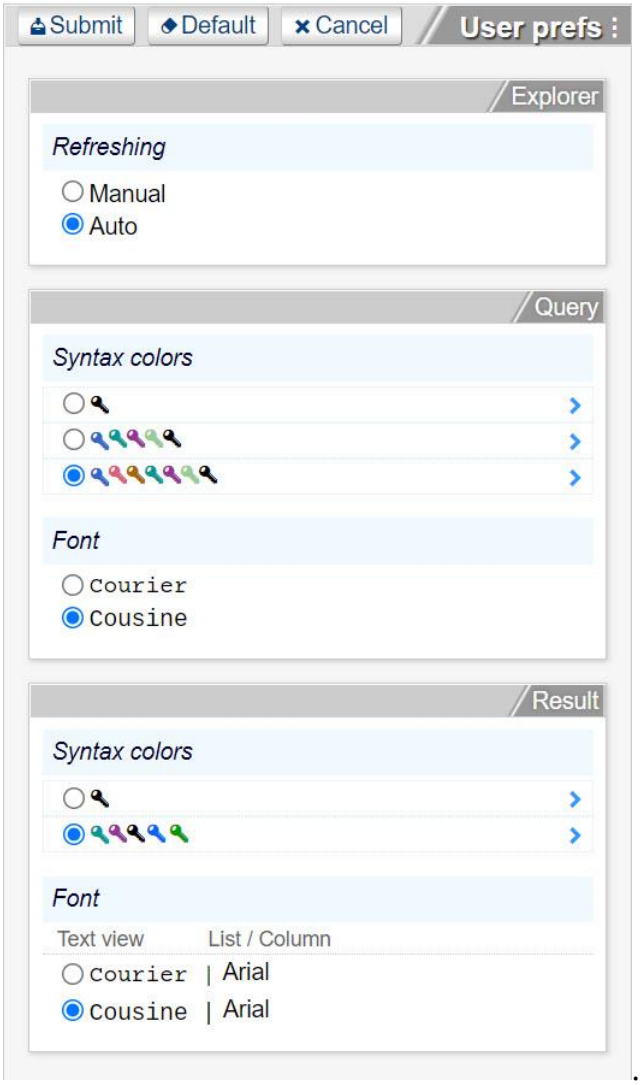
The Profile menu allows you to customize the user interface of Studio.



Upon selection of Preference, you are shown a panel with controls for these three panes:

- Explorer
- Query
- Results

There are distinct options for each pane:



2.1.5.1 EXPLORER PANE PREFERENCES

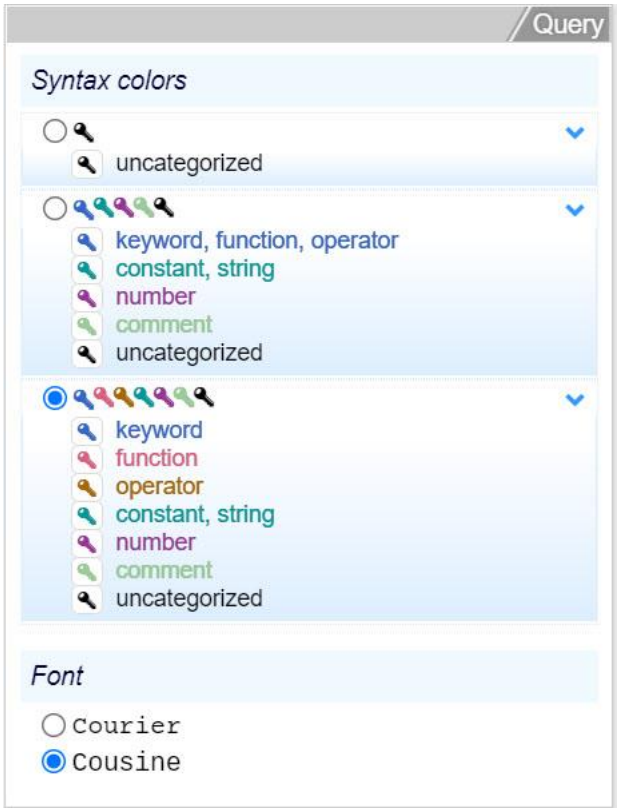
The radio button controls whether the Explorer tree control is automatically updated to show catalog changes you make using Studio, such as adding or deleting k-objects.

As the automatic refresh process can slow down Explorer when making many changes, you can toggle this control to Manual. When you are ready to see the changes, click the Refresh button.

2.1.5.2 QUERY PANE PREFERENCES

The Query pane gives you control over

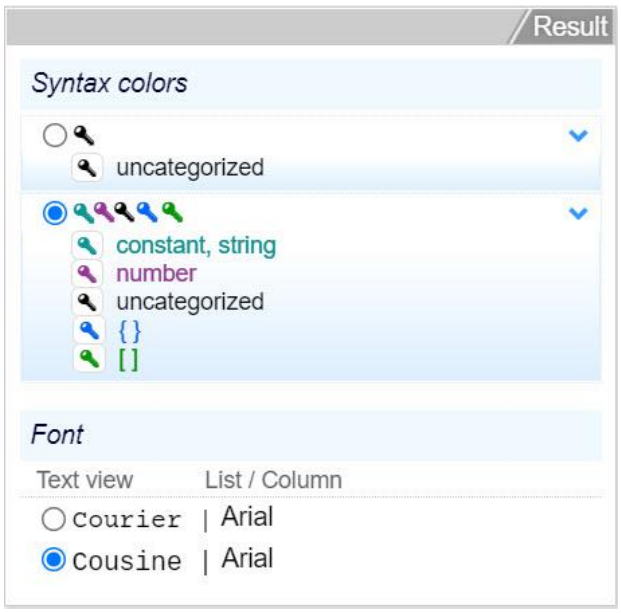
- Syntax colors
- Font



2.1.5.3 RESULT PANE PREFERENCES

The Result pane gives you control over

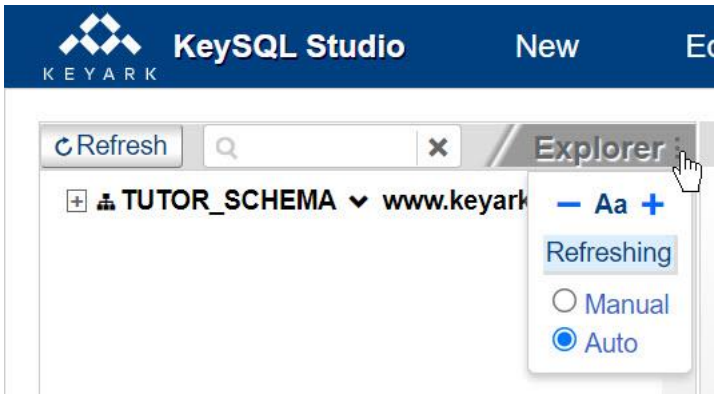
- Syntax colors
- Font



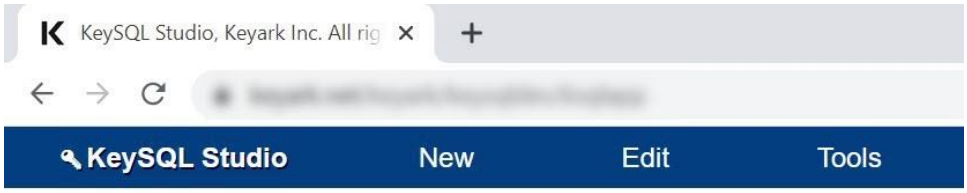
Note that the choice of font applies only to the Text view. Both the List view and Column view are Arial only.

2.1.5.4 FONT SIZE

For all three panes you can directly set the preferences by clicking on the pane name, as shown below for Explorer.



2.2 CREATING AND ALTERING SCHEMAS, CATALOGS AND STORES



These three menu choices are the jumping off point for getting your work done: **New**, **Edit** and **Tools**.

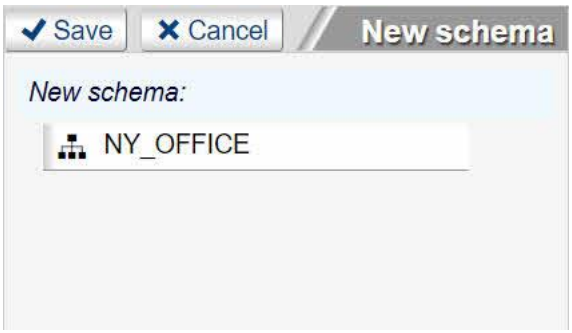
2.2.1 NEW: CREATING NEW SCHEMAS, CATALOGS AND STORES

2.2.1.1 NEW SCHEMA

A *schema* is a container for catalogs and stores that ensures uniqueness and is very similar to their counterpart in RDBMS. When a user account is created, it is assigned a specific schema, which is often the same name as the user account.

FYI: Permissions
The ability to access or create a schema is controlled by permissions Typically only user accounts that are explicitly granted administrative rights are empowered to create or alter a schema.

The screenshot below shows the New schema pane where you specify a schema such as PROD, DEV, QA, or an organizational unit such as the New York office:

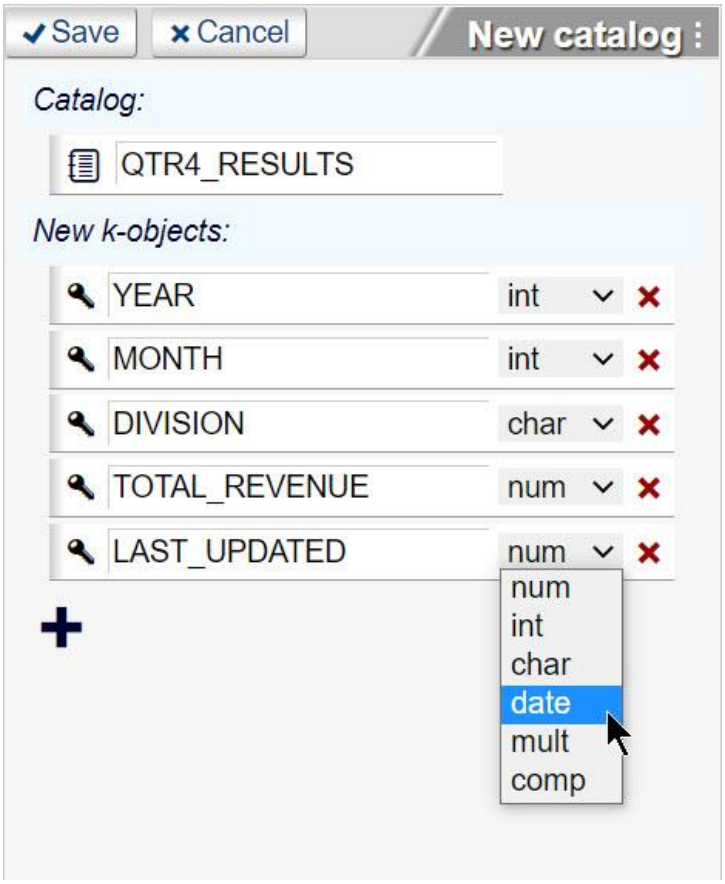


2.2.1.2 NEW CATALOG

A *catalog* is a namespace that ensures uniqueness of KeySQL objects (k-objects) within each catalog. Each catalog is a set of k-objects that are defined using CREATE KEYOBJECT statement. For example, there may be one catalog for personnel data k-objects, and another for accounting data k-objects.

Each elementary k-object can store a specific type information such as a character string (*char*), number (*num*), or date (*date*). These are analogous to fields in an SQL database. Besides the elementary k-objects, there are composite (*comp*) k-objects which comprise other k-objects as their elements. A special case of a composite k-object is a multi-composite k-object (*mult*) that is comprised of multiple copies of another single k-object.

The screenshot below shows the New catalog pane where you design the catalog, in this case a catalog called QTR4_RESULTS:



Once the catalog is designed, KeySQL will generate KeySQL statements that upon execution will create this catalog. The sequence is:

1. Enter the catalog name
2. Create the k-objects
3. Click the **Save** button to generate the KeySQL CREATE CATALOG statement
4. Click the Execution Pane **Execute** button

FYI: Important Background:
KeySQL names, including catalog, store and discrete k-objects, may begin with either a letter or underscore only. Names are capital insensitive, hence displayed in all capital.

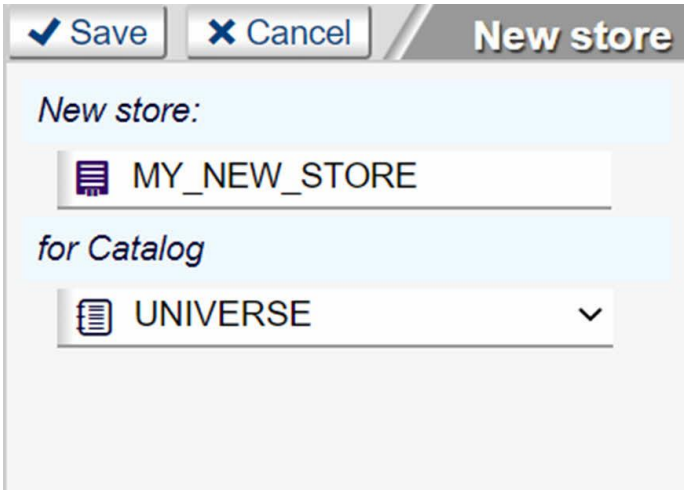
As each k-object is defined, the data type is selected from the dropdown.

Where KeySQL is especially flexible is the availability of the *mult* and *comp* data types. Please see the KeySQL reference guide for an explanation.

2.2.1.3 NEW STORE

A KeySQL *store* is analogous to an SQL table and SQL database at the same time. It is a container for data, each of which is an *instance* of some k-object from the specified catalog on which the store is based. For example, an accounting catalog might be used to create stores for physical assets, financial assets, audits, government filings and so forth. Those can be several stores or just one store since the store definition generally does not specify what catalog k-object instances it can contain.

The screenshot below shows the New store pane, depicting the creation of a new store MY_NEW_STORE that will store data for you in the UNIVERSE catalog:



The steps are:

1. Enter the store name
2. Select from the dropdown the catalog that will contain this new store
3. Click the **Save** button to generate the KeySQL CREATE STORE statement
4. Click the Execution Pane **Execute** button

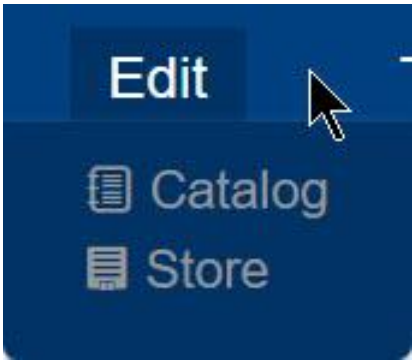
2.2.2 EDIT: ALTERING AND RENAMING CATALOGS AND STORES

This section provides a set of three ‘how to’ explanations:

- How the KeySQL Studio menu system shows the catalog and store context
- How to rename catalogs and their k-objects, as well as stores
- How to alter the k-object catalog, adding, moving and deleting k-objects

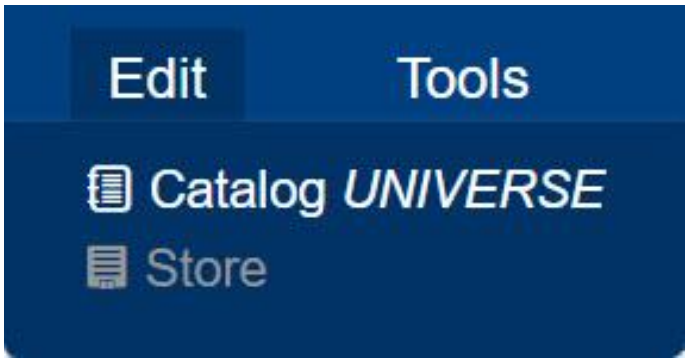
2.2.2.1 HOW THE MENU SYSTEM SHOWS CONTEXT

The KeyStudio Studio menu shows the currently selected Catalog and Store. Initially, before any catalogs or stores are selected, these menu options are dim, indicating these menu actions are not available:



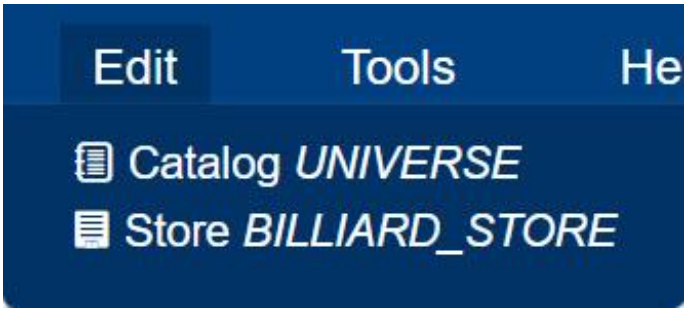
In the image below note how a catalog, but neither of the stores are selected in Explorer:

Accordingly, only the catalog menu option is available:



In the image below the BILLIARD_STORE was selected in Explorer:

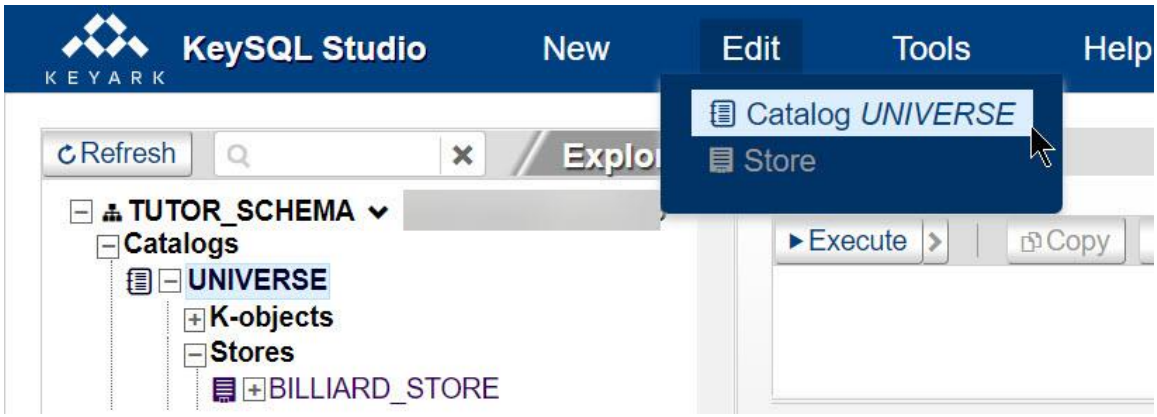
Now additionally the store menu option is available:



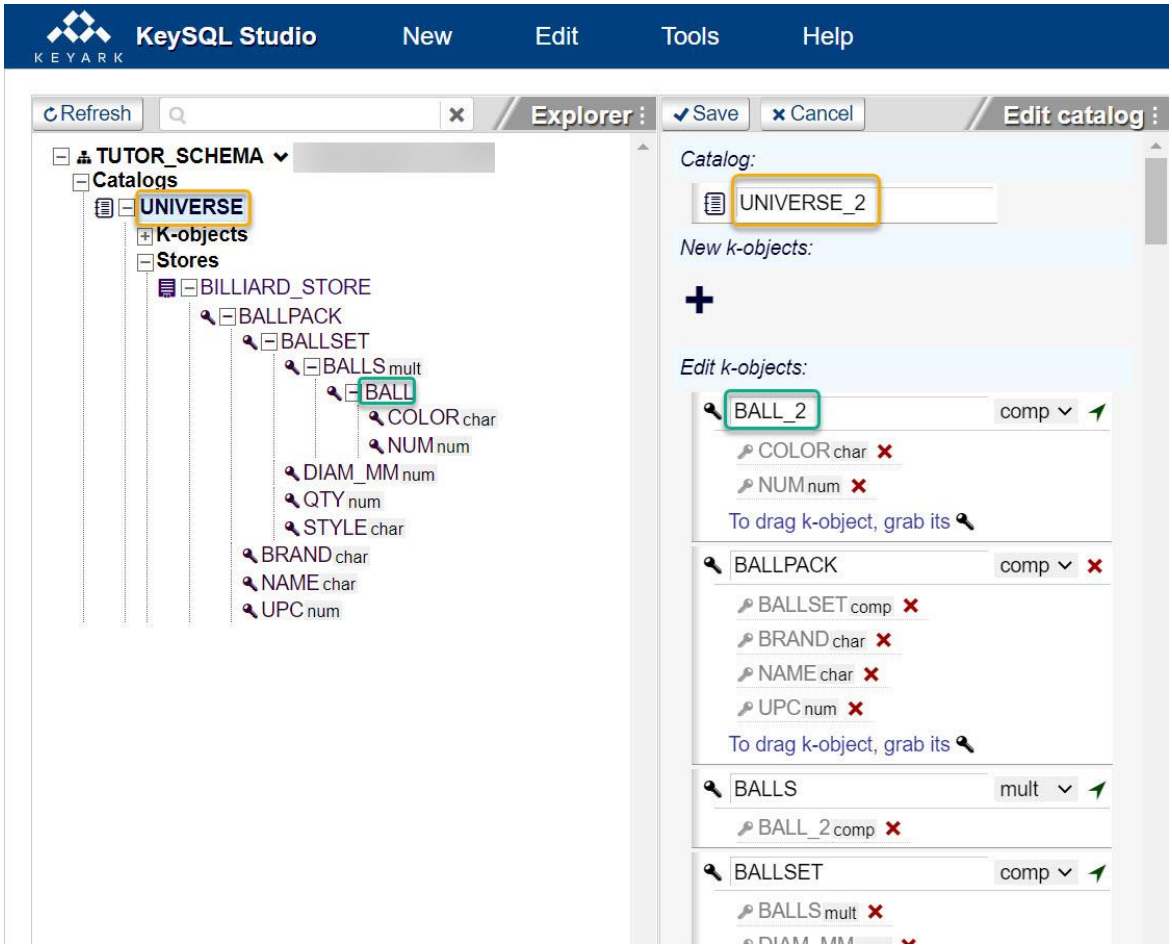
2.2.2.2 HOW TO RENAME A CATALOG, ITS K-OBJECTS, AND RENAME STORES

Studio makes it easy to rename and alter catalogs by providing a drag-and-drop interface. Upon a click of the Save button, KeySQL commands are presented in the query window. Just click Execute, same as for a query, and the catalog or store changes take place.

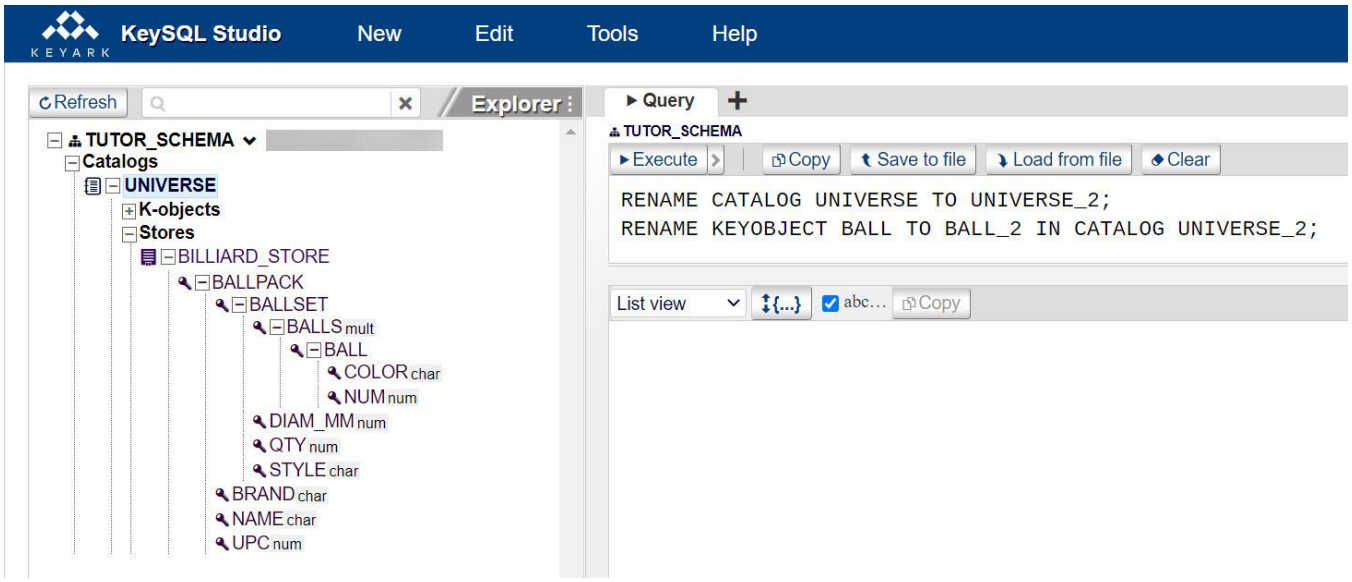
RENAME A CATALOG



The Rename catalog pane enables you to rename both the catalog and any of the constituent k-objects. In this example the catalog UNIVERSE is being renamed to UNIVERSE_2, and simultaneously the composite k-object BALL is being renamed to BALL_2:



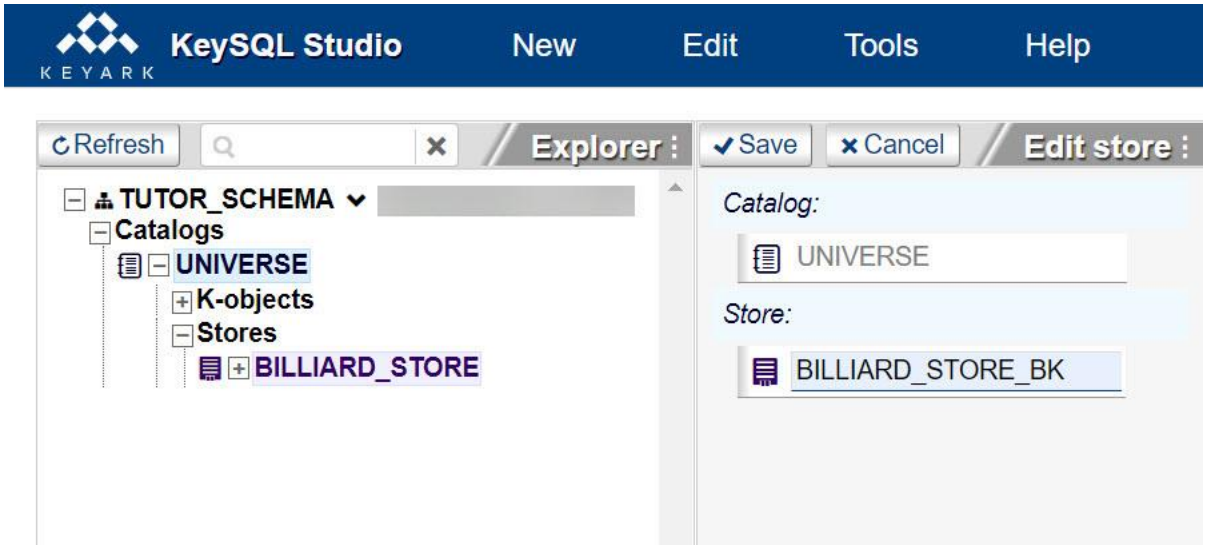
When you click the **Save** button, ready-to-run KeySQL statements are written on the Query pane:



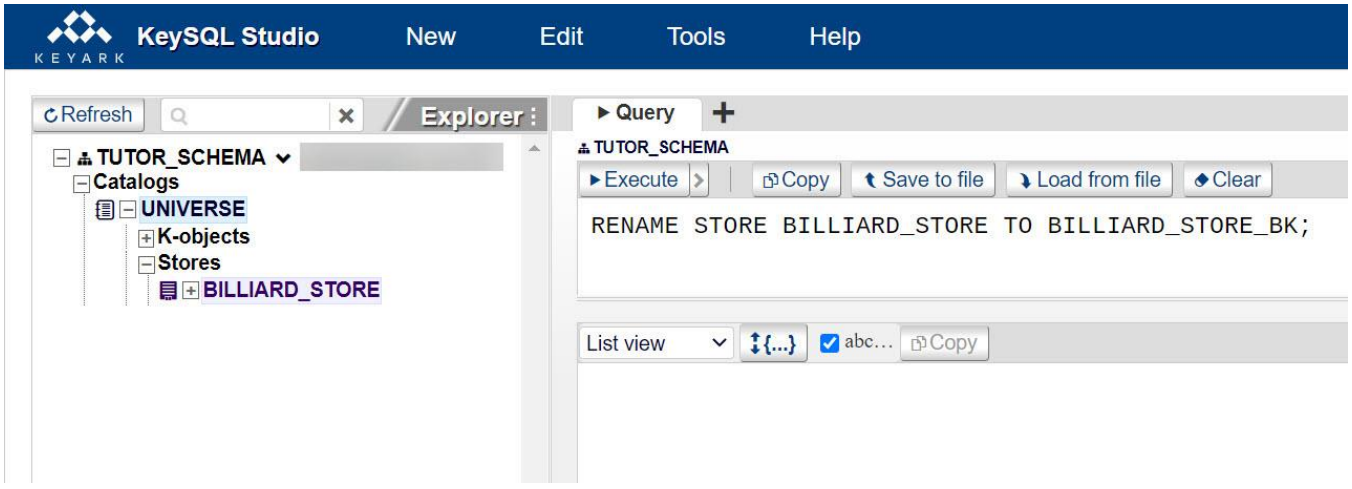
When you click the **Execute** button, the catalog and k-object(s) are renamed, and the Explorer tree view is refreshed to show the renames.

RENAME A STORE

The Edit store pane enables you to rename just the store:



When you click the **Save** button, a ready-to-run KeySQL statement is written for you.

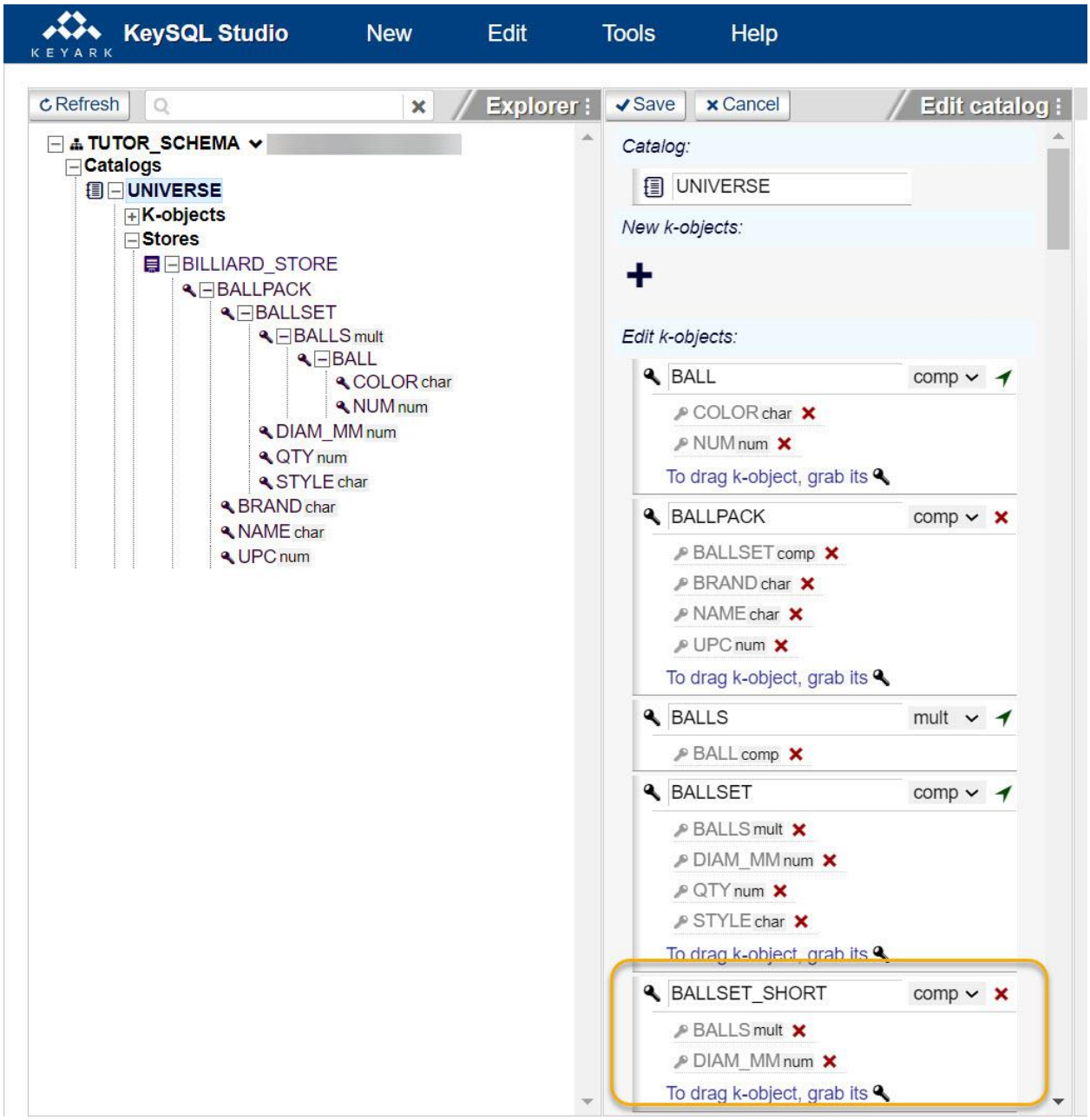


When you click the **Execute** button, the store is renamed, and Explorer is refreshed.

2.2.2.3 HOW TO EDIT THE CATALOG

As shown earlier, the Edit catalog pane enables you to rename a catalog and its constituent k-objects. It also allows you to create new k-objects and re-organize the k-objects already in the catalog. You can also delete k-objects, if not already in use either in a store or within another k-object.

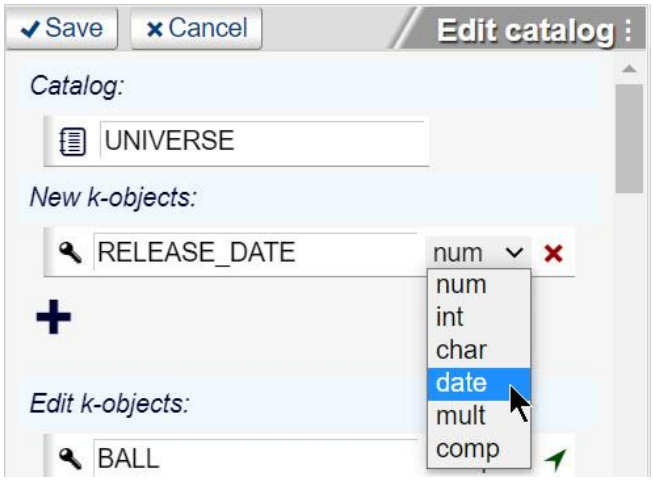
Please note that k-objects are presented in alphabetical order. Keep in mind that the k-objects seen in the catalog are defined for use beyond just BILLIARD_STORE; catalog k-objects can be defined in advance of use. And a given k-object, such as COLOR, can be utilized by any number of stores associated with that catalog. In the screenshot below note the existence of k-object BALLSET_SHORT:



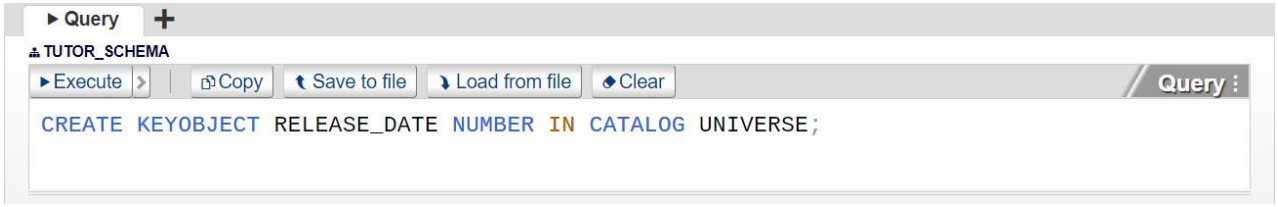
In the Explorer pane, the k-objects specific to store BILLIARD_STORE are fully expanded. Now notice the last k-object shown in the Edit Catalog pane, which is BALLSET_SHORT. This k-object exists in the catalog, in anticipation of future use, but is not part of BILLIARD_STORE. There is no penalty in adding k-objects, either in consumption of disk space nor in waiting for the operation to complete.

ADD K-OBJECT TO THE CATALOG

To define a new k-object, click the + icon. The *New k-objects* field appears. Enter a name (begin with a letter) and select the data type from the drop-down:



Studio is smart enough to identify all the changes made in the Edit catalog menu and generate only those alter statements. When you click the **Save** button, these ready-to-run KeySQL statements are written to the Query results area, as shown below for a new k-object RELEASE_DATE of type DATE:

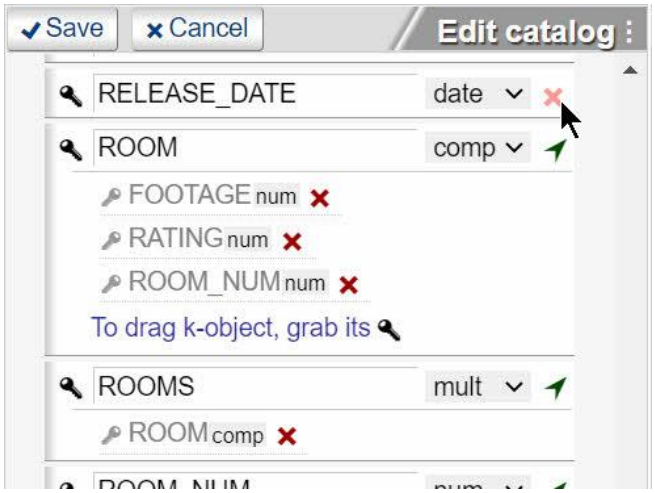


When you click the **Execute** button, this k-object is created and immediately visible in Explorer.

DROP K-OBJECT FROM THE CATALOG

To drop (database terminology for *remove*) a k-object click on this red icon:

Studio protects you from making mistakes in terms of dropping k-objects. You can directly remove objects with the red X (versus the green pointer):

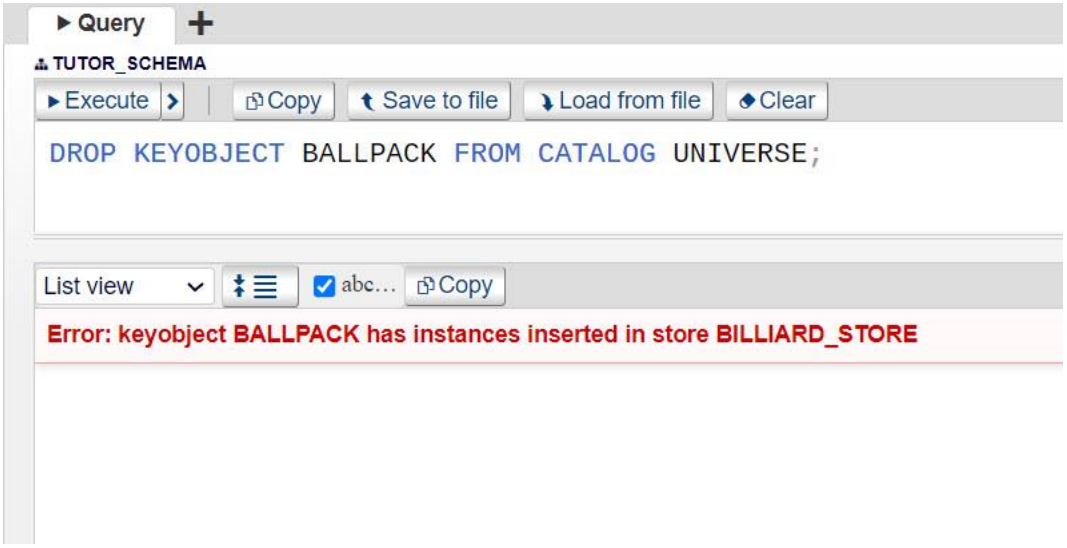


In the screenshot above note how the composite ROOM shows a green pointer, meaning you can move it but cannot drop it. To drop ROOM, you must first remove each of its composite objects, FOOTAGE, RATING and ROOM_NUM, by clicking their red X.

When you click the **Save** button, your ready-to-run KeySQL statements are written for you, as shown below for a drop of the recently created k-object RELEASE_DATE:

When you click the **Execute** button, Studio checks to see whether this k-object is found in any of the instances (data) across all the stores in the catalog. If the k-object exists only as a data definition in the

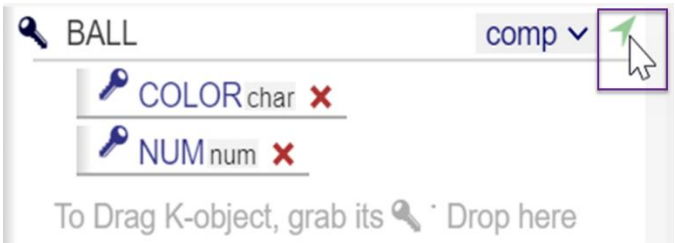
catalog, meaning that none of the stores contain instances with this element of data -- the situation for the just created RELEASE_DATE -- the k-object can be removed from the catalog. If there are instances with data for this k-object, an explanatory error message is displayed. For example, all instances in BILLIARD_STORE have a host object of BALLPACK, so attempting to drop BALLPACK gives the error:



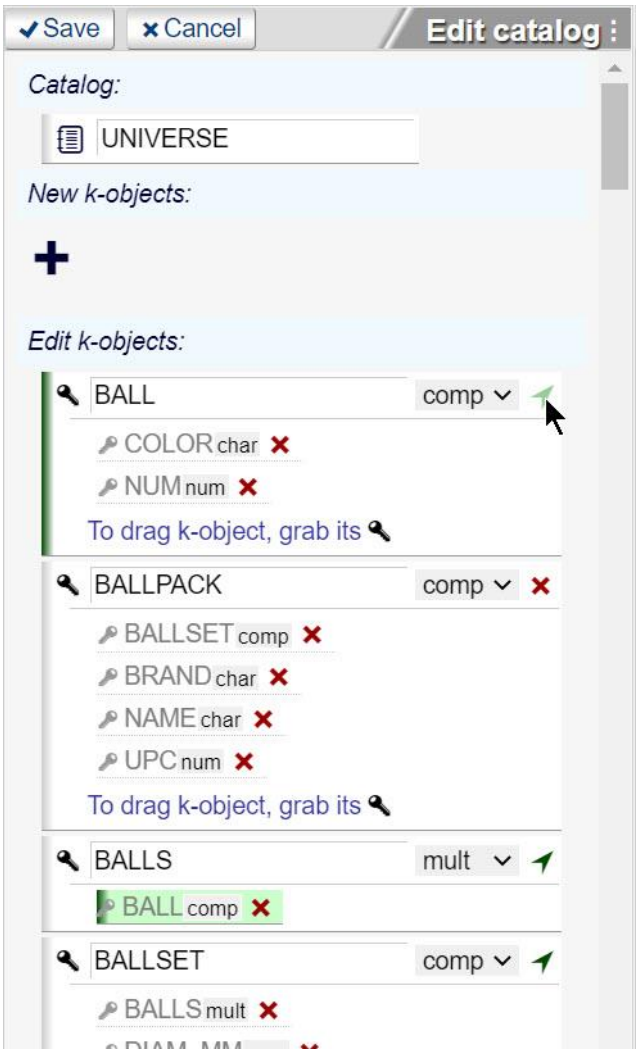
NAVIGATING THE K-OBJECT HIERARCHY

Not all k-objects can be dropped in the fashion just described. A k-object that is a constituent element of another k-object is flagged with a dark green icon (which becomes light green when cursor hovers over it) that indicates that it can be moved, but not dropped.

To see every use of this k-object in the catalog, click on the green icon.



Scrolling through the catalog, you see the k-object highlighted in green:

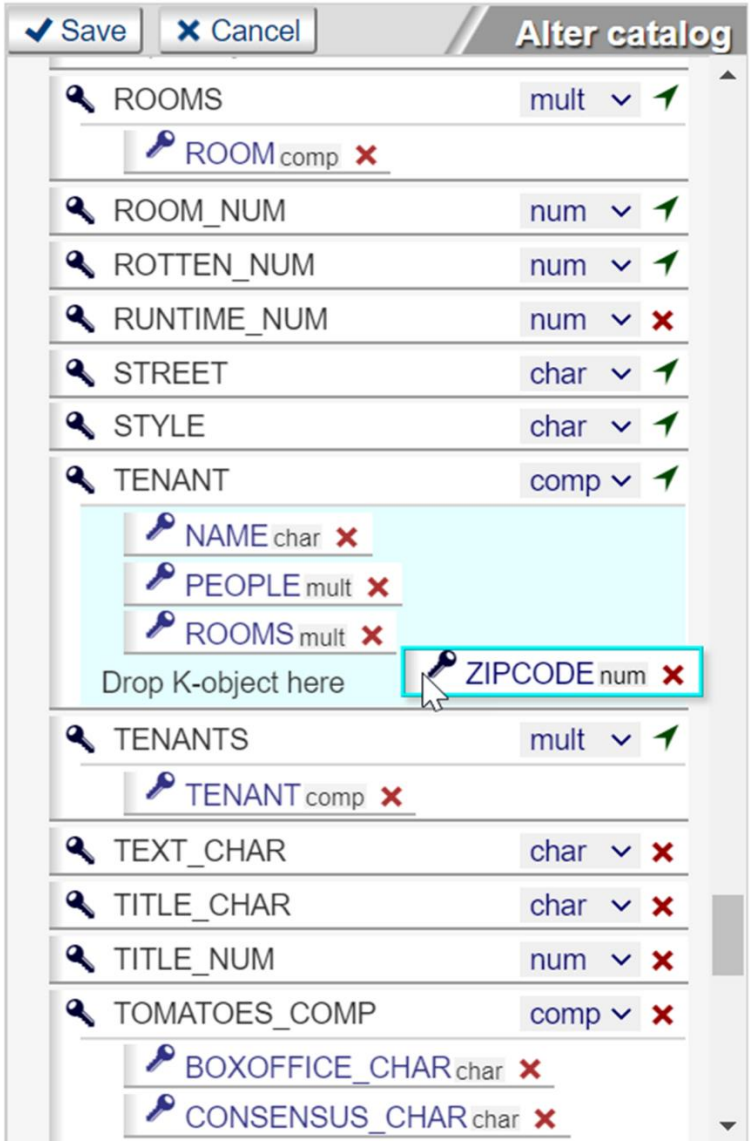


FYI: Should you still desire to drop that k-object from the catalog, you would need to first drop it from the composite (*comp*) or multi-composite (*mult*) k-objects that possess it.

MOVE A K-OBJECT

You can re-organize your catalog, using drag-and-drop to add k-objects to composite or multi-composite k-objects.

The screenshot below depicts adding the k-object ZIPCODE to the composite k-object TENANT by dragging ZIPCODE to the drop zone for TENANT (look for words *Drop k-object here*):



To remove a k-object from a composite, such as for TOMATOES_COMP in the screenshot above, click its red icon. You cannot drop via drag-and-drop. Note that this action does not remove the k-object from the catalog, only removes it from the composite definition.

2.3 TOOLS

KeySQL Studio has the capability to import and export data in a variety of formats.

Both Import and Export offer a rich functionality, hence are described in their own chapters.

FYI: Given that KeySQL Studio runs in your browser, and reads or writes to text files, there are limits as to the quantity of data that can be handled in a single import or export operation. For frequent or large data transfers, or to transfer data directly to KeySQL Server from an application or database server, consider asking your IT organization to investigate use of KeySQL Server drivers, which are available for the most popular programming languages.

IMPORT

Please refer to the **3. Import** chapter.

EXPORT

Please refer to the **4. Export** chapter.

TABS

Please refer to the **Session Save and Restore** section of the **1. Getting Acquainted** chapter.

3 IMPORT

KeySQL Studio makes it easy to populate your KeySQL. You can import from the following data formats:

- KeySQL
- CSV
- JSON, including BSON
- MongoDB
- XML

3.1 IMPORT KEYSQL STATEMENTS

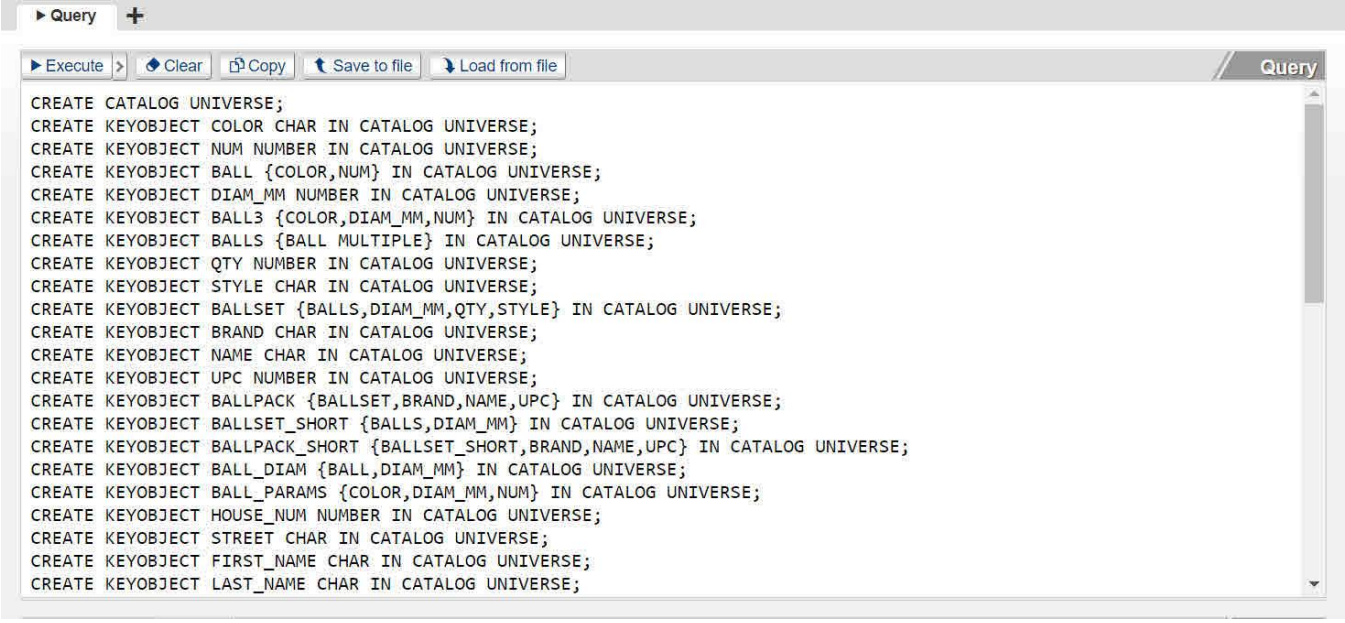
The **KeySQL Import** menu enables you to:

- Create a Catalog populated with k-objects
- Create a Store
- Populate a Store with data generated by KeySQL INSERT statements

Selecting **KeySQL** causes a file dialog to appear. Once you select a file, the content appears in the Execution pane Query text area for you to preview and potentially revise. Note that if the KeySQL statements are too numerous to display in the text area, KeySQL Studio will instead show a pop-up that asks you to confirm execution of this file.

3.1.1 CREATE A CATALOG EXAMPLE

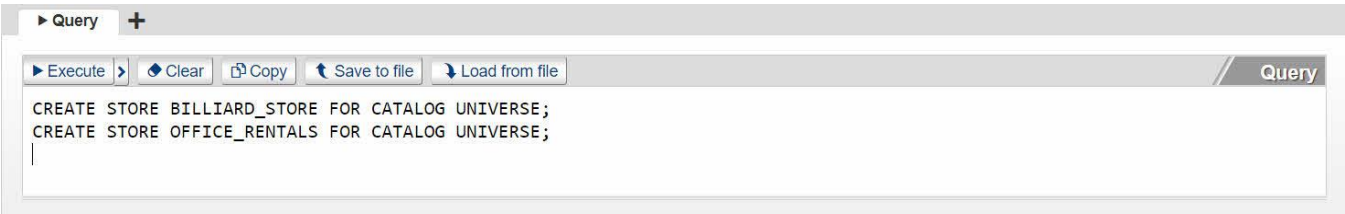
To execute these statements, click the **Execute** button. This will create a catalog named Universe.



The screenshot above depicts statements imported from file “catalog_UNIVERSE.ksql“ which was generated by Export of the Universe catalog.

3.1.2 Create a Store Example

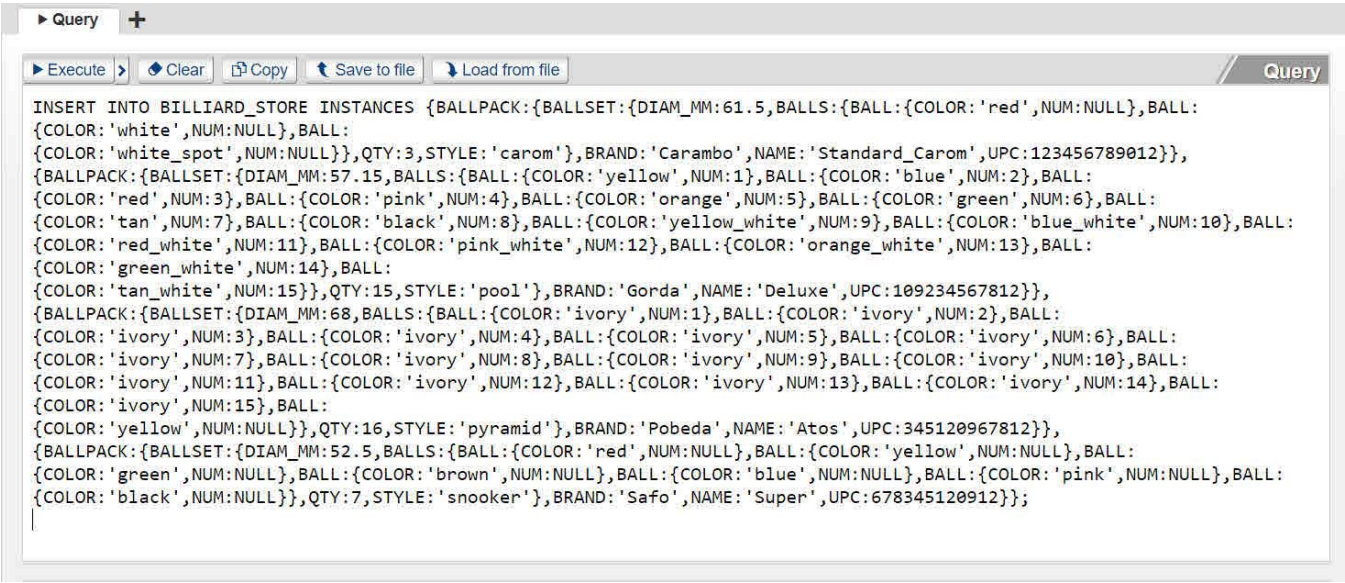
To execute these statements, click the **Execute** button. This will create a store named BILLIARD_STORE, and another store named OFFICE_RENTALS.



The screenshot above depicts statements imported from file “stores_UNIVERSE.ksql“ which was generated by Export of the Universe stores.

3.1.3 Populate a Store Example

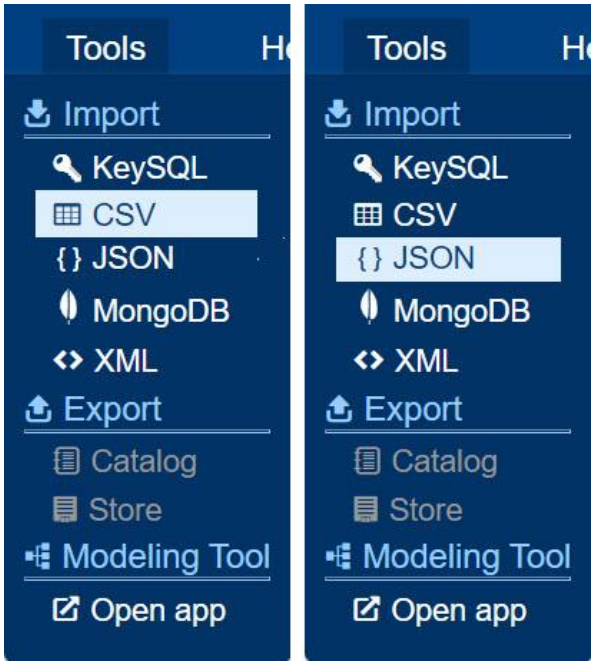
To execute these statements, click the **Execute** button. This will populate a store named BILLIARD_STORE with data. Each INSERT statement creates an instance (data) in the BILLIARD_STORE.



The screenshot above depicts statements imported from a file “store_BILLIARD_STORE.ksql“ which was generated by Export of the BILLIARD_STORE store instances (data).

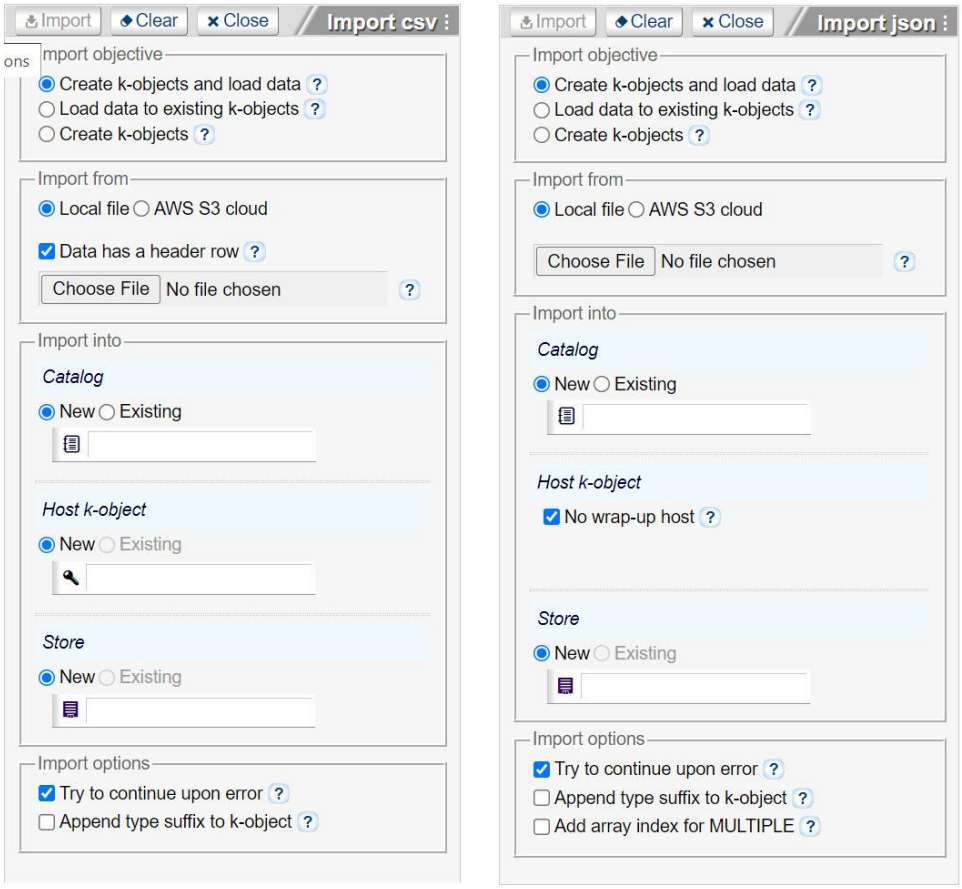
3.2 IMPORT CSV AND JSON

Data is frequently provided by CSV or JSON files. The procedure for importing these two formats is similar, so this section will share the similarities. Subsequent sections will provide a step-by-step example of CSV import, followed by JSON import. It’s recommended to read these two sections in order, first CSV and then JSON.



3.2.1 Getting Familiar with Import Pane

The **Import csv** pane is shown on the left, and the **Import json** pane on the right:



Selecting **CSV** or **JSON** displays the **Import from file** pane for that flavor of data. Use this pane to select a data file to import and set the conditions for import. In the side-by-side comparison below, with the CSV pane on the left, and the JSON pane on the right, you can see a few differences:

- CSV checkbox for having a header row
- JSON radio button to specify selection of JSON schema file rather than a data file

A quick explanation of the Import pane is provided immediately below, followed by import examples.

Please note that KeySQL Studio will display a helpful tip when you hover your cursor over the question mark:

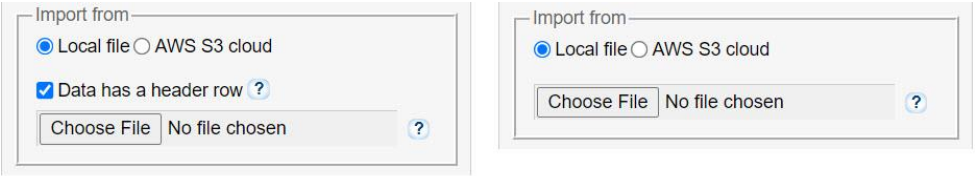
3.2.1.1 FORM BUTTONS

The three buttons at the top of the pane have the following purpose:

Icon	Button	Purpose	Always available?
	Import	Import data now	Available when all input fields are filled
	Clear	Clears the pane, restoring the original pristine form	Yes
X	Close	Closes the pane	Yes

3.2.1.2 PICK DATA FILE

The Choose file button lets you navigate to the data file for import.



3.2.1.3 SET IMPORT SCENARIO - CATALOG

Specify whether you wish to import into a New catalog or an Existing catalog.

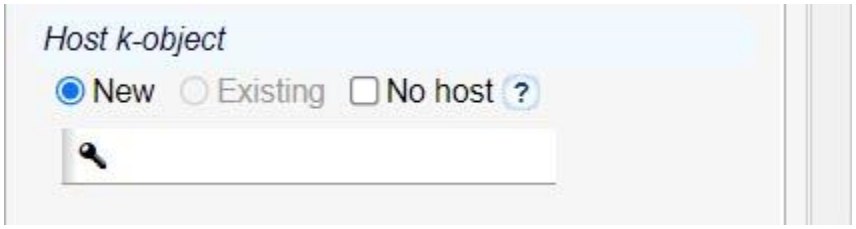
Should you toggle the radio button to **Existing**, KeySQL Studio will display existing Catalogs for you to pick among.

3.2.1.4 SET IMPORT SCENARIO - STORE

Specify whether you wish to import into a New catalog or an Existing store.

Should you toggle the radio button to **Existing**, KeySQL Studio will display existing Stores, for the already selected Catalog, for you to pick among.

3.2.1.5 SET IMPORT SCENARIO - HOST K-OBJECT



For most imports, the filename is automatically used as the Host k-object. Feel free to shorten the Host k-object to a familiar abbreviation. The naming rules are the same as for other k-objects; it must begin with a letter.

WHAT IS A HOST K-OBJECT?

The **host k-object** for a given store is any k-object an instance of which is inserted into the store using the KeySQL Insert statement. It can contain or *host* the instances of other k-objects from the base catalog, hence the name.

An explanation of the checkbox **No wrap-up host**, an advanced feature, is provided in **3.3 Advanced Topics**.

3.2.2 CSV IMPORT EXAMPLE

In this example two CSV files will be imported to a new store in the HOUSEHOLD_FINANCE catalog. This catalog is designated to hold credit card information that you download:

CSV File	Content	Sample Data
CreditCardAmExBill.csv	Once a month bill summary	Closing Date, Payment Due Date, Reward Points, New Charges, Fees, New Balance 11/8/2020, 12/5/2020, 23749, 1548.20, 0, 1548.20
CreditCardAmExCharges.csv	Individual transactions, such as charges and credits	Transaction Date, Vendor, Card, Method, Amount 11/19/2020, INSTACART SAN FRANCISCO, JANICE VOLLMANN - 02019, ,119.64 11/7/2020, APPLE.COM/BILINTERNET CHARGE CA, BILL VOLLMANN - 03004, Paid for with Apple Pay, 0.99

Note that both CSV files have a *header* as the first row. KeySQL Studio uses this header row to name the k-objects it will create to represent this data.

FYI: If your CSV file contains data without a header row, uncheck the **has header** check box. The Import process will automatically create k-objects named COLUMN_1, COLUMN_2 etc.

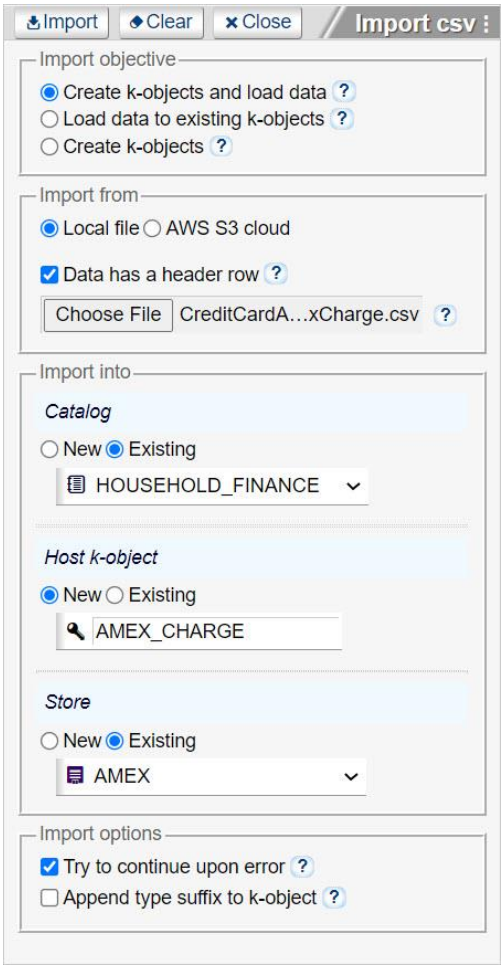
3.2.2.1 DOING THE FIRST IMPORT

In the Import from file pane below you will see the settings that match the story given above.

Three details to note:

- There is a check mark for the **data has a header row** option. Most CSV files provide a first row with field names, the situation for our file, hence the check mark.

- Click the **Choose File** button to navigate to the data file. Note that the file picker initially displays files with a CSV extension, but you can easily change this to display all file extensions.
- Upon selection of file *CreditCardAmExBill.csv* the Host k-object was automatically populated with *CreditCardAmExBill_CSV*. This was manually edited to the more readable AMEX_BILL.



To begin the import, click the **Import** button.

While the import is running you will see a circular progress indicator with an estimated percent complete:

Upon completion of this import, you will see a completion status pop-up:

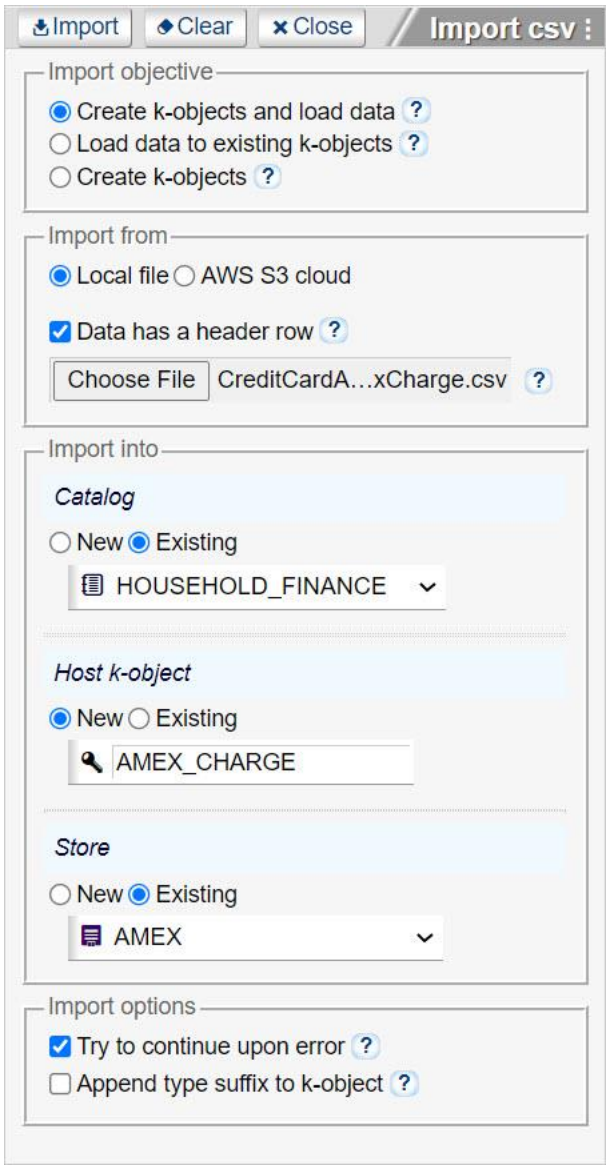
Click OK to close this dialogue.

You can proceed to your next import by either clicking the Clear button or using the file picker.

3.2.2.2 DOING THE SECOND IMPORT

This time you will select the *CreditCardAmExCharges.csv* and edit the Host k-object to just AMEX_CHARGE.

As we are adding this import to the same store, for the catalog selection you click the Existing button and select HOUSEHOLD_FINANCE from the dropdown. For the store selection you also click the Existing button and select AMEX.



To begin the import, click the **Import** button.

Upon completion of the second import, you can click the Close button, which shuts the Import pane.

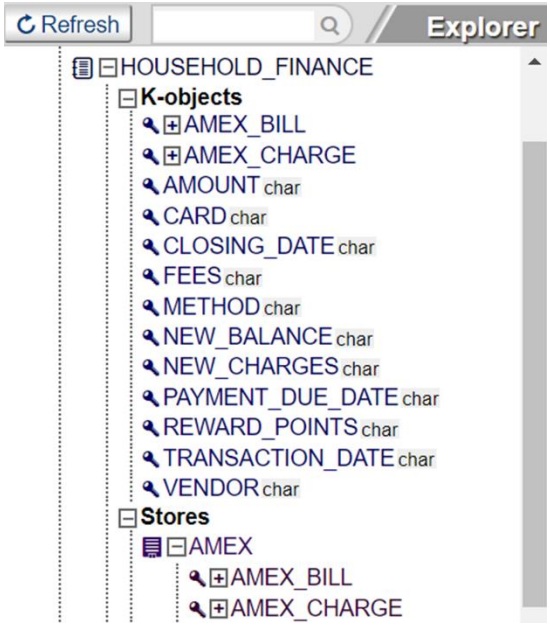
FYI: The CSV Import routine for an existing store matches CSV field name to the store host k-object name. Accordingly, an import will be successful, even if the columns are in a different order.

Only CSV columns whose field name matches the store host k-object are imported. Any columns that do not match are ignored. All hosted k-objects, for which there was no import data, are assigned the null value.

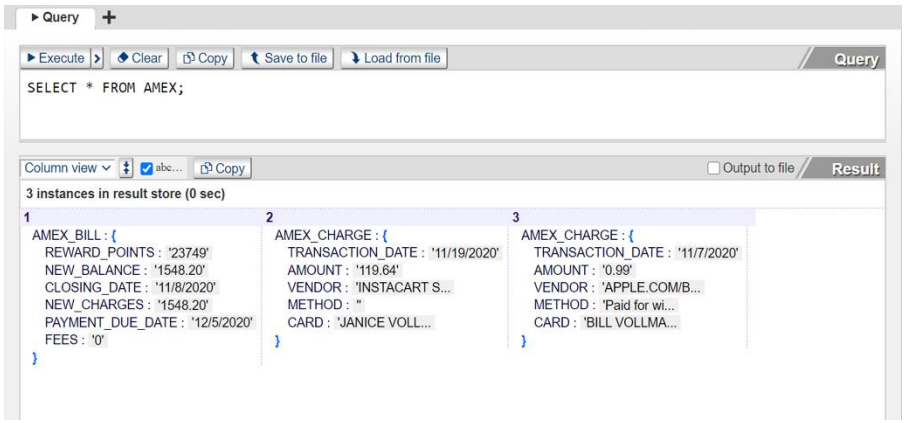
3.2.2.3 EXAMINING THE IMPORT IN EXPLORER

The two Host k-objects are visible in the store AMEX:

Upon expanding the Host k-objects you see that a k-object was created for each column in the CSV field:



Executing a SELECT query shows the value for each k-object for these three instances, one “bill” and two “charges”:

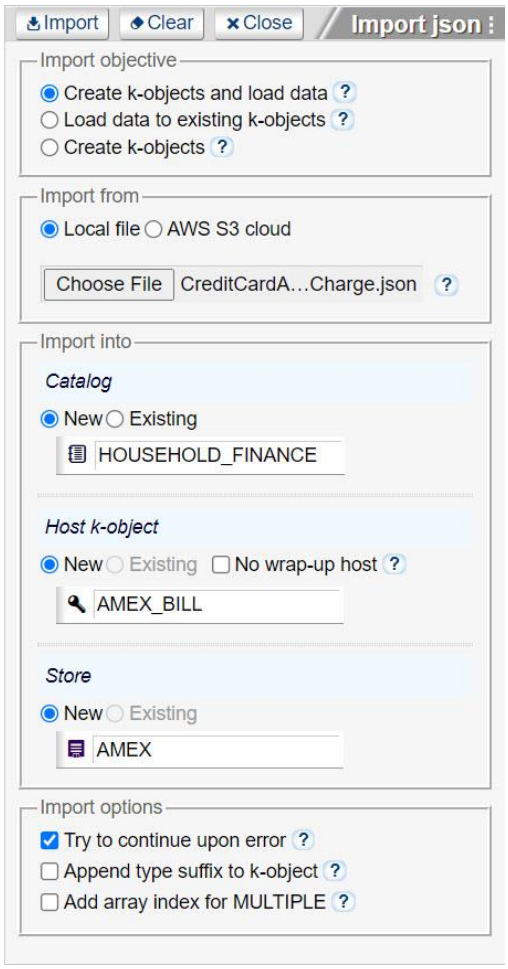


Upon close examination of the screenshot above, you will notice that all the data values are character type data, including data that is clearly date or number values. This is an inherent limitation of CSV import, a limitation that does not exist for JSON/BSON import, as will be shown in the JSON Import Example.

3.2.3 JSON IMPORT EXAMPLE

This section presents examples of JSON Import that illustrate the richness in import features beyond that available with the CSV Import described in the preceding section.

The screenshot below depicts a filled-in Import pane just before clicking the **Import** button to initiate the import process:



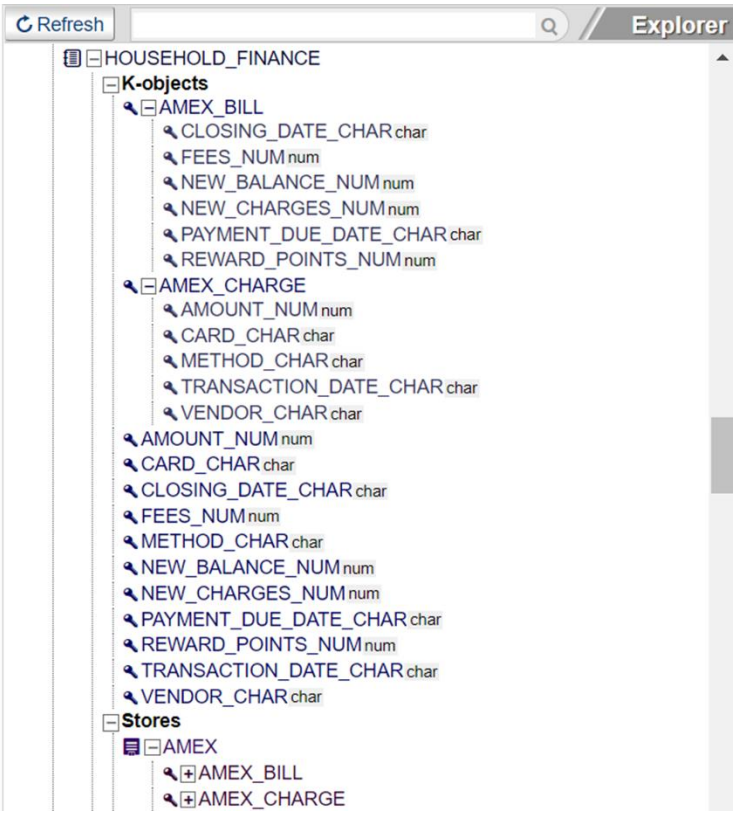
This example features file **CreditCardAmExCharge.json** which contains data in JSON format. This is the same data seen in CreditCardAmExCharge.csv that was featured in the CSV Import Example:

```
[
        {
                "Transaction Date": "11/19/2020",
                "Vendor": "INSTACART SAN FRANCISCO,JANICE VOLLMANN - 02019",
                "Card": "JANICE VOLLMANN - 02019",
                "Method": null,
                "Amount": 119.64
        },
        {
                "Transaction Date": "11/7/2020",
                "Vendor": "INSTACART SAN FRANCISCO,JANICE VOLLMANN - 02019",
                "Card": "APPLE.COM/BILINTERNET CHARGE CA,BILL VOLLMANN - 03004",
                "Method": "Paid for with Apple Pay",
                "Amount": 0.99
        }
]
```


If you were to compare Explorer for a JSON data import against a CSV import you will notice differences in how the values are interpreted:

	CSV Import	JSON/BSON Import
Data Types	**char** (character string) only	**char (character string)** **num** (numeric value including money) special value of **null** where data is missing
K-object Name	Taken from the header row.	Taken from the JSON field label plus addition of the underline character (_) followed by data type

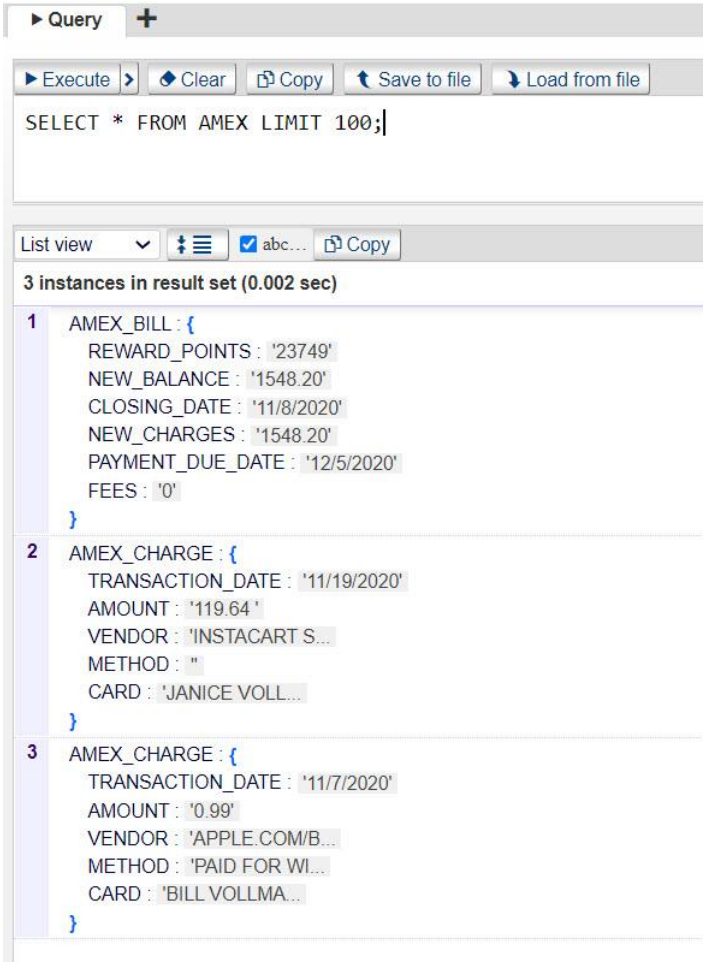
Looking at Explorer, you will see that about half of the k-object names end with *CHAR*, and the other half with NUM, depending on whether the data is character or numeric in type.



Teaser alert: In the next section you will learn how to give your k-objects a more regular name, without the _CHAR or _NUM. The explanation will include a pleasant surprise for why KeySQL Studio Import uses this naming convention.

Performing a SELECT query reveals that the data stored in KeySQL is the same as seen for the CSV Import with three subtle but important differences:

- An amount value, when saved to a numeric k-object, is not surrounded by a single quote
- A missing value will appear as NULL for a num k-object
- A missing value will appear as either NULL or ‘’ for a char k-object, depending on whether the import data was missing, or an empty string. If missing, then NULL will appear. If an empty string, then two single quotes with nothing in between will appear.



This note is for readers who are new to JSON data and curious as to how a JSON file is constructed:

- each instance begins with an open brace { and ends with a close brace }
- instances are separated by a comma
- within an instance there are data elements separated by a comma
- when streaming data, an array (demarcated by opening and closing brackets) of instances is common:
  [{"amount":23 , "date":"01/23/21"} , {"amount":34 , "date":"01/24/21"}]

Each **data element** (which is mapped to a k-object) has three parts:

- label in double quotes (e.g. “Transaction Date”)
- colon separator
- value

The **value** can be any of the following:

- character string such as "11/7/2020” or "Paid for with Apple Pay"
- number such as 0.99
- null which indicates there is no value

Unfortunately, the JSON standard lacks a date type, hence dates are represented as a character string such as “11/7/2020”. Consequently, many importers, including KeySQL Studio, leave such dates as a character string to avoid potential conversion errors.

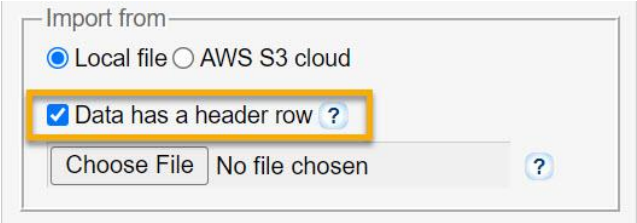
3.2.3.1 EDIT K-OBJECTS PRIOR TO LOADING

Caution: This feature is recommended only for advanced users, and for one-time imports.

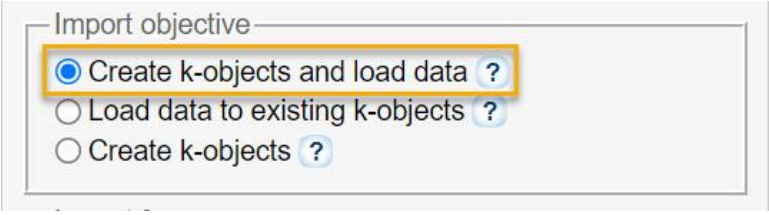
It is best to limit revisions to changing only a k-object name, and best to avoid touching the data type even though the data type can be edited.

The CSV and JSON examples already presented featured an automated import, where the k-objects were taken from a scan of the field names in the data file.

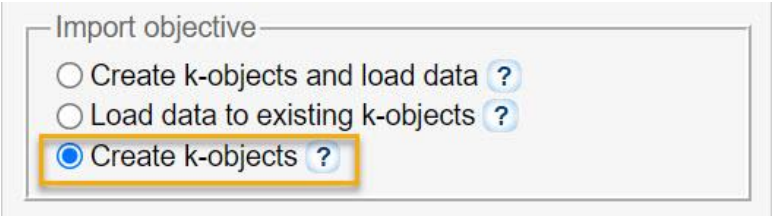
For CSV files, the k-object names are typically taken from CSV header row due to default setting shown here:



For JSON files, the k-object names are taken from the JSON field labels due to the default setting shown here:

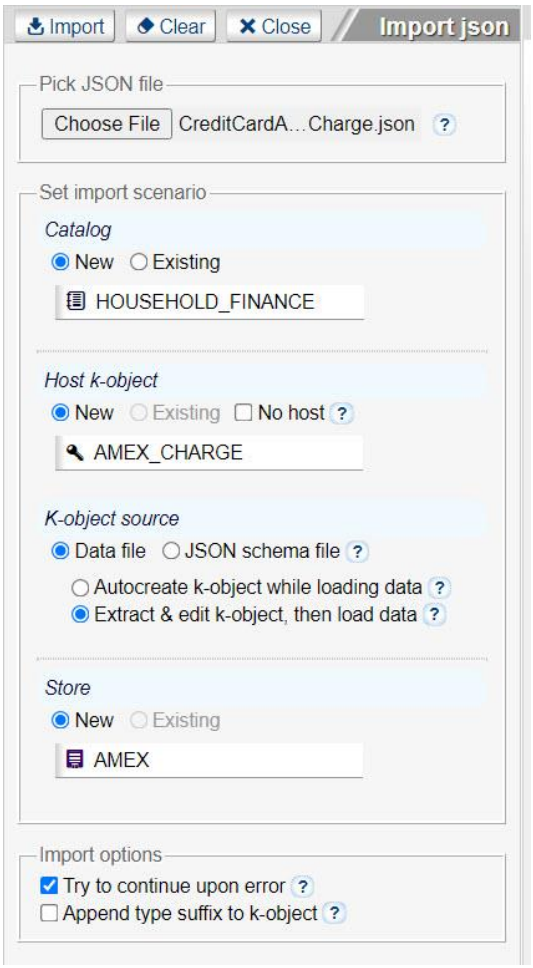


Selecting the *Create k-objects*, shown below, enables you to review and possibly change the k-object data type before their addition to the catalog:

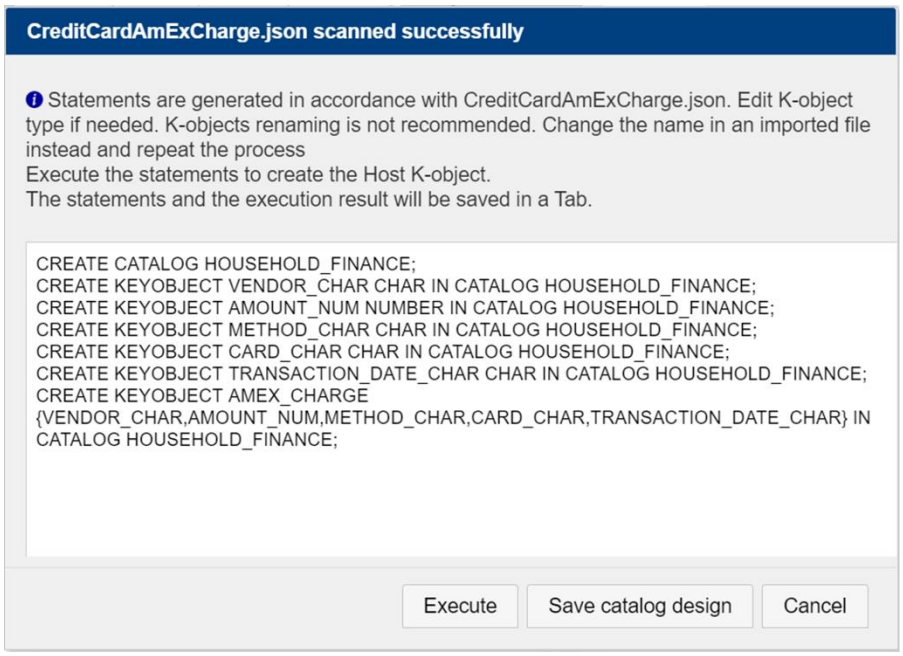


A second step, *Load data to existing k-objects*, will be necessary to load the data.

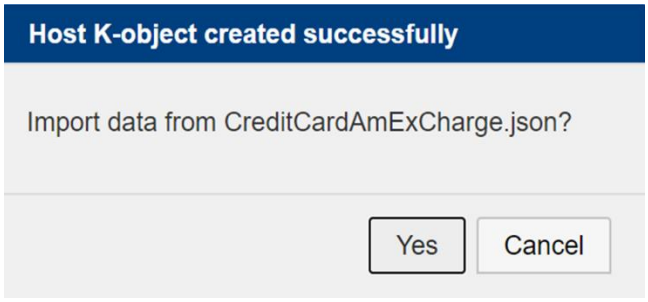
Below is an example of the **Import json pane** that shows the complete setup for creating k-objects (but not yet loading data):



When the Import button is clicked, a window appears which reveals the KeySQL statements before they are executed. This gives you the opportunity to revise the k-object names, such as to change VENDOR_CHAR to VENDOR, and AMOUNT_NUM to AMOUNT:



The usual path is to click the **Execute** button. You will then see a pop-up asking whether to continue to the second step, importing the data:



Selecting **Yes** will import your data.

Selecting **Cancel** will stop at this point, creating a catalog with k-object definitions, but not importing data.

Saving the Catalog Design

Rather than execute the KeySQL statements, or cancel import before loading data, there is a third choice, to save your KeySQL statements for later use.

The table below provides a fuller explanation of these three choices:

Button	Purpose	Detail
Execute	Execute the statements immediately.	Upon success in adding these k-objects to the catalog, the data is automatically ingested.
Save catalog design	Save the KeySQL statements, including any revisions just made, to a text file for future use. Nothing is executed; you immediately returned to the Import pane.	To reuse the saved statements, open the file in a text editor, and paste them into a fresh Import pane textarea, replacing the statements already there.
Cancel	Nothing is executed; you immediately returned to the Import pane.	Any changes you made to the KeySQL statements will be lost.

Caution: Do not change the data type or make other structural changes to these statements. Additional functionality, such as changing the data type, may be offered in a future release of KeySQL Studio Import, but at this time will cause a program error.

3.2.4 BSON Import Example

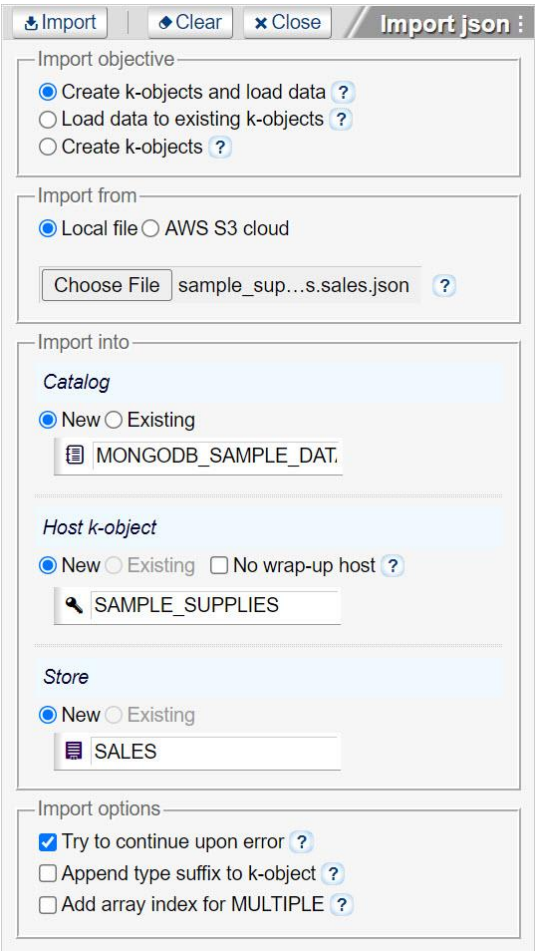
This section presents an example of BSON Import. It's a popular data format for exchanging data between document databases such as MongoDB. The beauty of the BSON data format is that there is a nice variety of data type, including both number and date, as well as arrays. When you use KeySQL Studio's Import facility you create k-objects of compatible types, namely num and date, and mult.

3.2.4.1 WHAT IS BSON?

The name "BSON" is based on the term JSON and stands for "Binary JSON". It is a binary form for representing simple or complex data structures including associative arrays (also known as name-value pairs), integer indexed arrays, and a suite of fundamental scalar types. BSON originated in 2009 at MongoDB. Several scalar data types are of specific interest to MongoDB and the format is used both as a data storage and network transfer format for the MongoDB database, but it can be used independently outside of MongoDB. (from https://en.wikipedia.org/wiki/BSON)

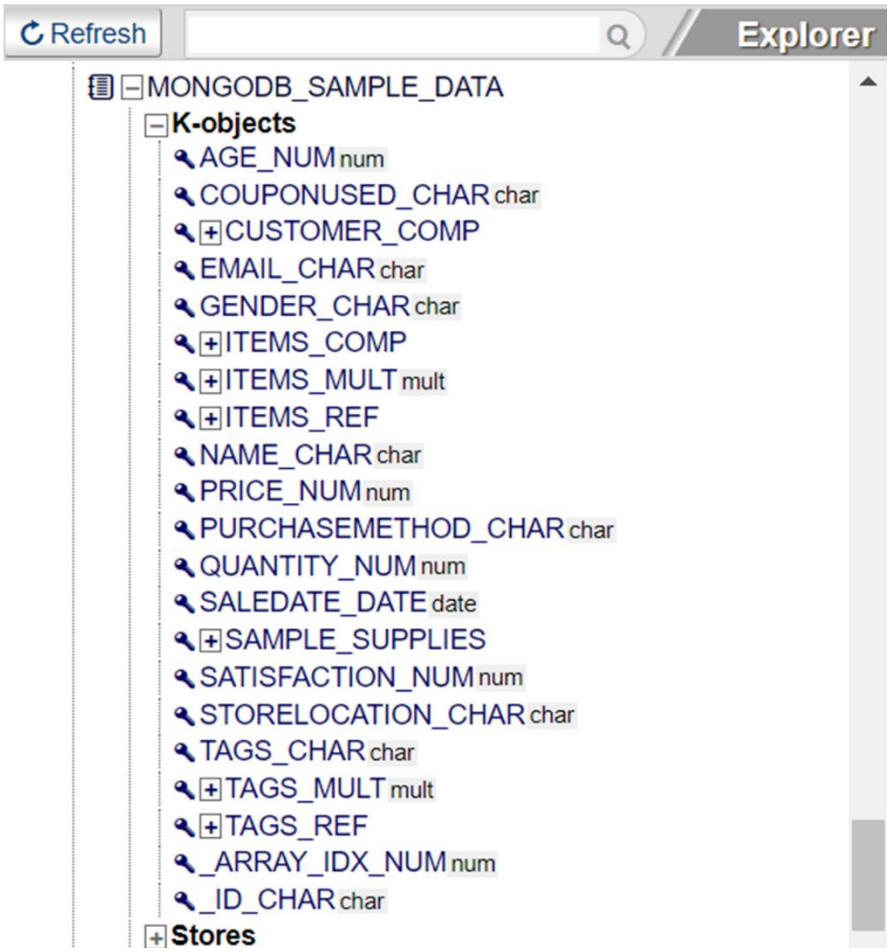
This example illustrates import of [sample data made available by MongoDB, Inc] for their customers to load into their MongoDB(R) Atlas cluster. As the data is stored in BSON format, it can be imported by a wide range of document friendly databases including KeySQL Server.

The screenshot below depicts a filled-in Import pane just before clicking the **Import** button to initiate the import process. Note that the Import pane is the same for BSON and JSON. KeySQL Studio Import accommodates either format:



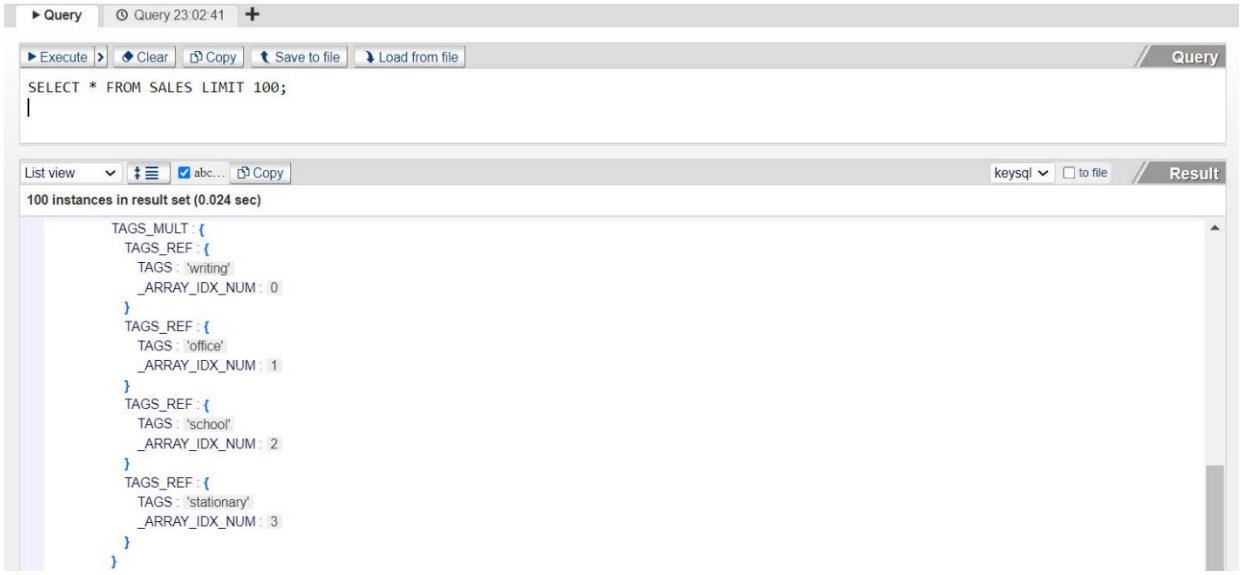
The screenshot below shows the successful import. SQL Studio faithfully replicated the complex document design. Note the preservation of data types:

- Numeric data (num k-object) such as AGE_NUM
- Character string data (char k-object) such as COUPONUSED_CHAR
- Date data (date k-object) such as SALEDATE_DATE



Also notice that KeySQL supports arrays, such as seen for TAGS_MULT which contains any number (from none to an impossibly large number) of tags for the given product.

The screenshot below shows an exemplary TAGS_MULT as shown in the SQL Studio Result section:



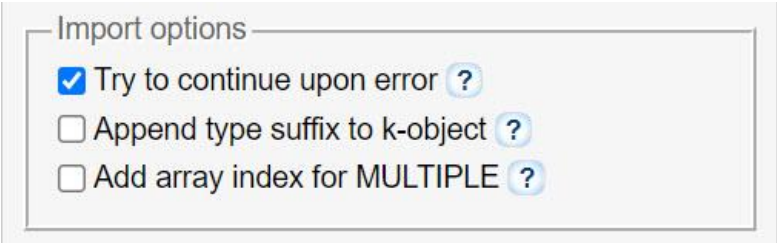
The BSON data file contained an array of these three labels, in this order, which represents the physical order that these labels were saved in the original MongoDB ® collection:

- school
- travel
- kids

Depending on the Import Options, when the KeySQL Studio imports data and recognizes an array, an _ARRAY_IDX_NUM k-object will automatically be included to capture the physical order. This is described in the next section.

3.2.5 Import Options

The default import options are fine for most situations:



Note that the options vary according to type of file. For example, CSV and XML lack **Add array index for MULTIPLE.** The complete set of options is listed below:

Import Option	Purpose	Detail
Try to continue upon error	When checked (the default setting), continue the data file import even though a data import error has occurred. All errors will be reported upon completion of the import. When unchecked, the import process will stop upon the first error encountered.	Should this be checked, when an error is encountered KeySQL Import will attempt to load as much of the document/record as possible, as well as continue to import subsequent document/records. A null value will be written where k-objects could not be populated with values from the data file.
Append type suffix to k-object	When left unchecked (the default setting) a single k-object is created for each element of the document/record you import. As each k-object has a defined type (e.g. number or char), attempting to import a mix of will result in an error. When checked, Import will create one or more k-objects for each element, to accommodate each data type encountered. If you know that your data contains a mix of data types, you need to check this option. An example of using this option is shown in the	If using the "Autocreate k-object while loading data" to create the catalog and store, an error will result in neither being created. This is true even when "Try to continue upon error" is checked, as that controls data import only _after_ the catalog has been created. Here is an example of the behavior when the Append type option is checked, to allow for a mix of data types. Let's assume you have a data element called "quantity". If Import sees a mix of number and char, Import will segregate our data, creating a k-object quantity_num that holds values such as 15, 200.5 and so forth. Import will also create a k-object quantity_char that holds values such as "two", "none", "" and "no idea".

	section below *Import of JSON Field with a Mix of Data Value Type*	
Add array index for MULTIPLE	When checked, when a MULTIPLE is loaded with array data, _ARRAY_IDX_NUM is automatically created. It is an integer k-object that captures the array index.	

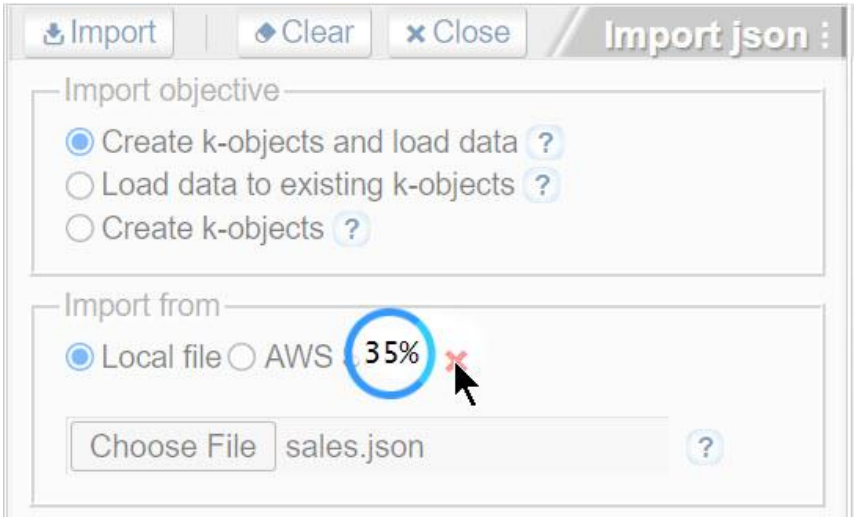
3.3 ADVANCED TOPICS

This section addresses topics that become useful as your gain experience with KeySQL Studio Import.

- Interrupting Long Imports
- Import of JSON Field with a Mix of Data Value Type
- Import of Date Field
- Array Import
- Import with JSON Schema File
- Import Limitations

3.3.1 INTERRUPTING LONG IMPORTS

While your import is underway, you will see a blue spinner with a percentage estimate for completion:



Should you wish to terminate the import, click the red X.

A possible side-effect is interrupting your connection to KeySQL Server. If so, a dialogue prompts you to reconnect.

3.3.2 Import of JSON Field with a Mix of Data Value Type

KeySQL Studio import has the nice ability to tolerate a mix of data types. Consider the four instance JSON example below, and compare the **Amount** value:

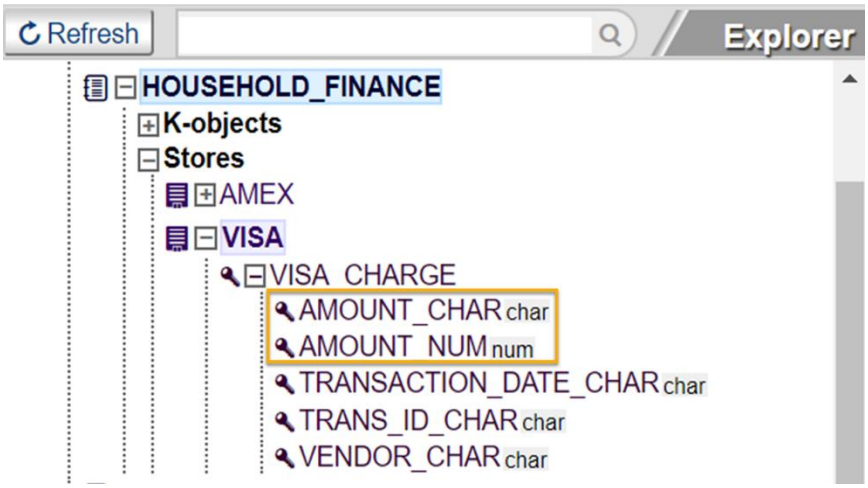
- JSON number values representing $ purchases: 6.00, 70.00 and 7.36
- JSON string value representing a Euro purchase: €32.00

```
[
        {
                "Transaction Date": "2020-11-07",
                "Vendor": "PayByPhone - SF",
                "Trans ID": "RE3829-SF",
                "Amount": 6.00
        },
        {
                "Transaction Date": "2020-11-08",
                "Vendor": "SFMTA*Street Cleaning",
                "Trans ID": "20383939331",
                "Amount": 70.00
        },
        {
                "Transaction Date": "2020-11-12",
                "Vendor": "JoesIceCream-Geary",
```



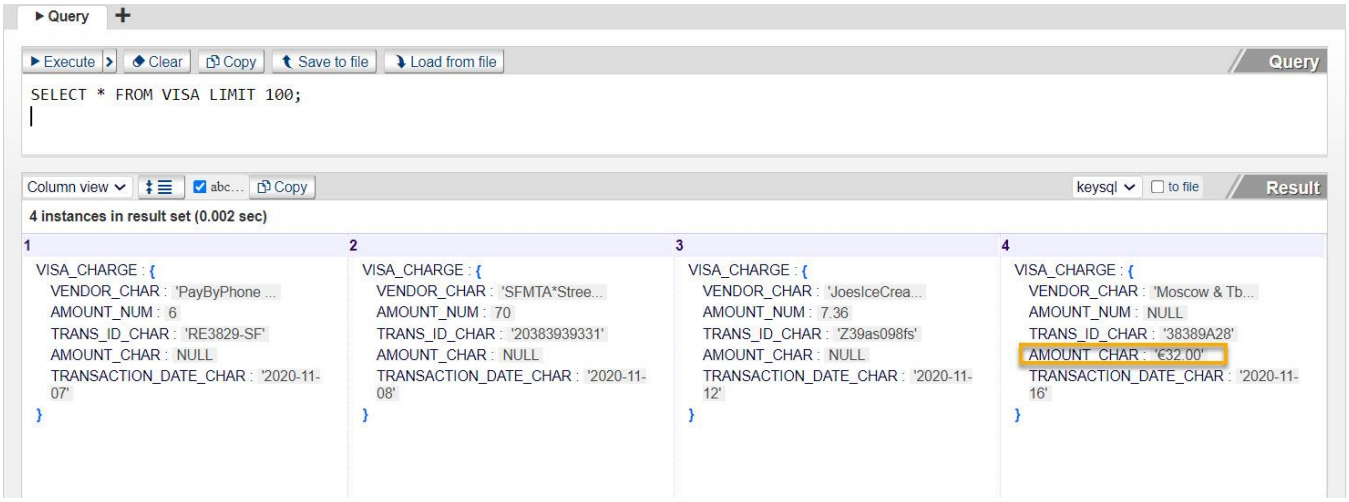
```
                "Trans ID": "Z39as098fs",
                "Amount": 7.36
        },
        {
                "Transaction Date": "2020-11-16",
                "Vendor": "Moscow & Tbilisi Bak",
                "Trans ID": "38389A28",
                "Amount": "€32.00"
        }
]
```

Upon import, note that two Amount k-objects were created, one **char** and one **num** type:



As a reminder, Import will automatically generate multiple k-objects as seen above, when you select the checkbox *Append type suffix to k-object*.

Examining the store, you can see how Import automatically segregated the char data `"€32.00"` from the number data:



This behavior is useful:

- You can calculate a dollar total with SELECT SUM(AMOUNT_NUM) FROM VISA;
- You can easily spot data exceptions (e.g. `"€32.00"`) and handle them at your leisure with simple KeySQL

KeySQL queries illustrate these benefits.

GET SUM OF THE $ VALUE

COUNT THE EXCEPTIONS

SHOW THE EXCEPTIONS

3.3.3 IMPORT OF DATE FIELD

When importing data to an existing store, where the target k-object has been defined as the **date** data type, KeySQL Studio Import will extract a date value when your data is in an easily recognized date or datetime formats:

Format	Purpose	Example
yyyy-MM-dd	Simple date	2022-02-27
yyyy-MM-dd HH:mm:ss	Simple data and time	2022-02-27 08:53:00
yyyy-MM-dd HH:mm:ss.SSSZ	Date and time with an optional fractional seconds	2022-02-27 08:53:00.100 100 milliseconds (1/10 second) into the 53rd minute
The formats below, with 'T' demarcation, are not yet supported by Import		

yyyy-MM-dd'T'HH:mm:ss	Simple data and time This variant features demarcation with 'T' rather than space	2022-02-27T08:53:00
yyyy-MM-dd'T'HH:mm:ss.SSSZ	Date and time with an optional fractional seconds This variant features demarcation with 'T' rather than space	2022-02-27T08:53:00.100 100 milliseconds (1/10 second) into the 53rd minute

FYI: These formats match the most common formats defined by the MongoDB Extended JSON (v2) standard which implements `ISO-8601 Date/Time Format` that's based on the proposed ITEF RFCC 3339 standard.

Note that date/time has a maximum time precision of milliseconds. If there is no fractional part, the SSSZ component can be omitted.

3.3.4 Array Import

The section above "BSON Import Example" shows an exemplary TAGS_MULT which is reproduced here:

The data file contains the array [“school”,”travel”,”kids”]. When KeySQL Studio Import ingests this array, it automatically adds the attribute **_ARRAY_IDX_NUM**. The numbering, beginning with 0, indicates the *physical order* that the tags appear.

Knowing the physical order can be useful when analyzing eCommerce or survey data. For example, you may import BSON data from a MongoDB based classroom enrollment system that captures a student’s preference in course. The tag order given by **_ARRAY_IDX_NUM** provides essential context, namely preference ranking.

3.3.5 IMPORT WITH JSON SCHEMA FILE

When JSON files are provided to you, they may be accompanied by a **JSON Schema** file. A JSON file is a lean data specification, containing one or more keys and a value or values for each. A more precise definition is that a JSON Scheme is a proposed IETF standard that contains metadata that potentially provides a description, the data type and even an example or validation rules (e.g. minimum value) for the JSON keys.

KeySQL Studio Import can process a JSON Schema file, creating KeySQL statements that, when executed in Studio, create KeySQL catalog k-objects for the data file.

FYI: Recall how you can conveniently and automatically add k-objects to the catalog via the “Create k-objects and load data” feature through inference based on a data file. There are several advantages to using a JSON Schema file, should a matching one exist for your data files:

a) the “Create and load” is based on the first data file you import, and only analyzes data in that first file. In contrast, a JSON Schema file by design should describe the data format across an *entire family* of import files. For example, the JSON Schema file may describe data elements that appear in later generated import files and not in the initial one scanned by Studio.
b) The schema file specifies data types, while Import must guess based on the data content, such as whether data that looks like “123” should be imported as a character string or a number.
c) The schema file assists Studio in validating the data during import. For example, should the schema file identify a data element as numeric, and a date value appears instead, Studio will know to tag this as an error.

3.3.5.1 SIMPLE JSON SCHEMA FILE

To get familiar with use of a JSON Schema file we will begin with a simple JSON data file and a simple schema file that explains the format of data in that data file.

Simple data file:

```
{
  "productId": 1,
  "productName": "A green door",
```



```
  "price": 12.50,
  "tags": [ "home", "green" ]
}
```

And the matching “simple” schema file (it really is simple, once you see the individual sections explained below):

Simple schema file:

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/product.schema.json",
  "title": "Product",
  "description": "A product from Acme's catalog",
  "type": "object",
  "properties": {
    "productId": {
      "description": "The unique identifier for a product",
      "type": "integer"
    },
    "productName": {
      "description": "Name of the product",
      "type": "string"
    },
    "price": {
      "description": "The price of the product",
      "type": "number",
      "exclusiveMinimum": 0
    },
    "tags": {
      "description": "Tags for the product",
      "type": "array",
      "items": {
        "type": "string"
      }
    }
  },
  "required": [ "productId", "productName", "price" ]
}
```

TOP DESCRIPTION SECTION

The top section is “boiler plate” that provides a general description of the JSON schema files, and you can pretty ignore it:

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/product.schema.json",
  "title": "Product",
  "description": "A product from Acme's catalog",
  "type": "object",
```


MIDDLE PROPERTIES SECTION

The middle “properties” section is most interesting. It lists the “properties” for each JSON key:

- productID
- productName
- price
- tags

The KeySQL Studio Import will translate each key specification seen here into a KeySQL k-object.

```
"properties": {
  "productId": {
    "description": "The unique identifier for a product",
    "type": "integer"
  },
  "productName": {
    "description": "Name of the product",
    "type": "string"
  },
  "price": {
    "description": "The price of the product",
    "type": "number",
    "exclusiveMinimum": 0
  },
  "tags": {
    "description": "Tags for the product",
    "type": "array",
    "items": {
      "type": "string"
    }
  }
},
```

Note the type associated with each key:

JSON Key	Type	KeySQL Type
`productid`	integer	number
`productName`	string	char
`Price`	number	number
`Tags`	array of string	MULT of char

As you can see above, the JSON Schema file lists attributes such as *description*, *type* and more. In the case of **price** there is an optional attribute *exclusiveMinimum* which provides a sensible data validation check that price cannot have a negative value:

"exclusiveMinimum": 0

Caution: Optional attributes such as “exclusiveMinimum” are hints, not iron clad rules. As KeySQL Studio is an analytical tool, not a transaction processing system, this attribute is ignored. All data – even “bad data” with a negative price – will be imported.

BOTTOM REQUIRED PROPERTIES SECTION

The bottom section is short, but vital: It indicates which properties must exist:

```
  "required": [ "productId", "productName", "price" ]
}
```

Caution: KeySQL Studio ignores the required property.

3.3.5.2 MORE COMPLEX JSON SCHEMA FILE

The code snippet below introduces additional complexity:

- **tags** property has two additional optional attributes: minItems and uniqueItems
- optional properties that characterize date keys added to the schema file: introDate and updateDatetime

```
 "tags": {
   "description": "Tags for the product",
   "type": "array",
   "items": {
     "type": "string"
   },
   "minItems": 1,
   "uniqueItems": true
 },
     "introDate": {
   "description": "Date of product introduction",
   "type": "string",
   "format": "date"
     },
     "updateDatetime": {
   "description": "Record last update datetime",
   "example": "2020-12-31T12:03:05.250",
   "type": "string",
```



```
"format": "date"
  }
```

Attributes of property *tags:*

Property Attribute	Value	Meaning
minItems	1	At least one tag must exist. According to this setting, KeySQL Import creates a MULT object.
uniqueItems	true	No duplicate values allowed in the tag array. KeySQL ignores this setting.

Attributes of properties introDate and updateDatetime:

Property Attribute	Value	Meaning
example	"2020-12-31T12:03:05.250" (updateDatetime only)	Helps explain the *key* but providing a typical *value*
format	"date"	The Schema file specification does not provide a “date” data type. So a hint is typically given by including a *format* attribute with the value of “date”.

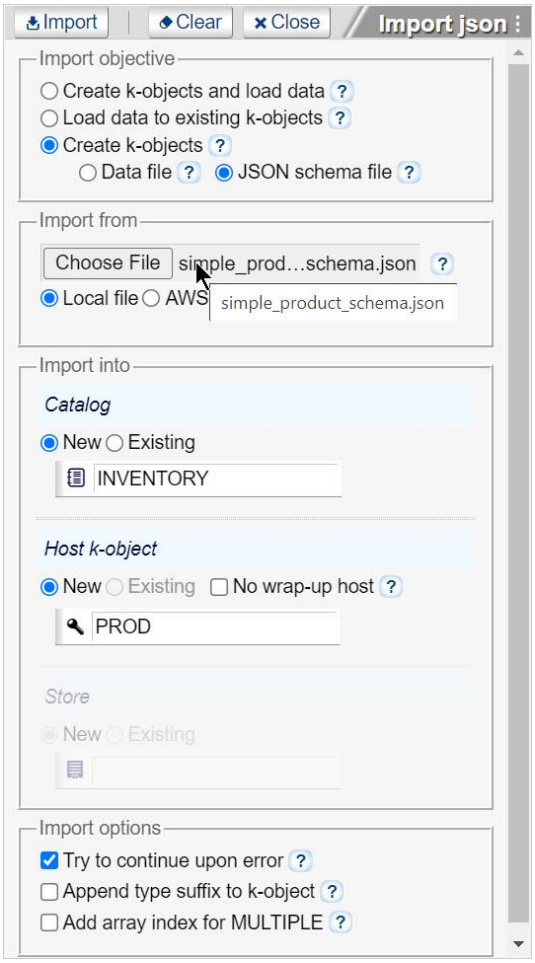
FYI: Currently KeySQL Studio Import does not recognize hint format=”date”

3.3.5.3 EXAMPLE OF IMPORT WITH SIMPLE JSON SCHEMA FILE

This section illustrates an import done with the same data file described above in “Simple JSON Schema File” but with the additional use of a JSON schema to validate the data.

JSON SCHEMA FILE SETUP

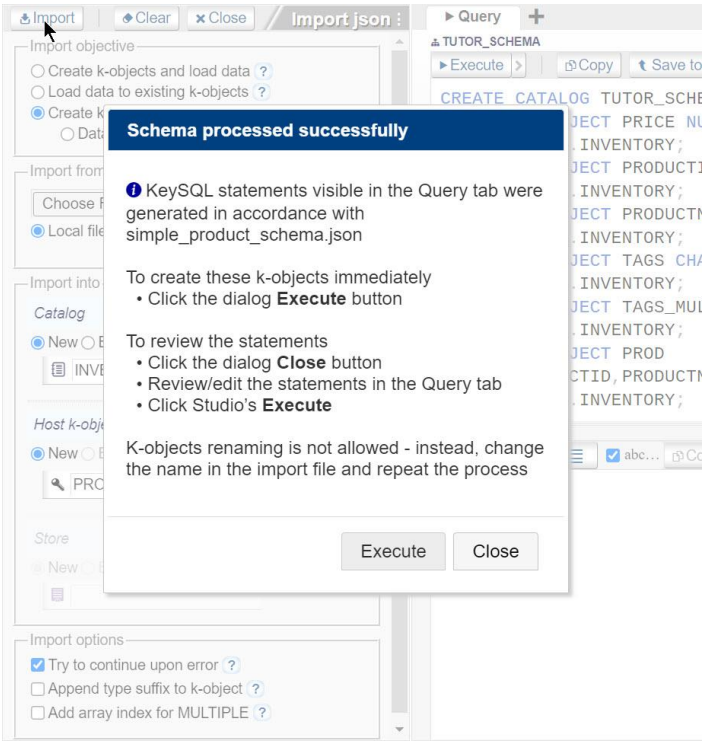
The setup is shown below:



Note that you need not specify a store. We use the JSON Schema file to only add catalog k-objects that define data we plan to store. At a later stage, when you proceed to import data into a store, you will need to specify a store.

JSON SCHEMA FILE PROCESSING

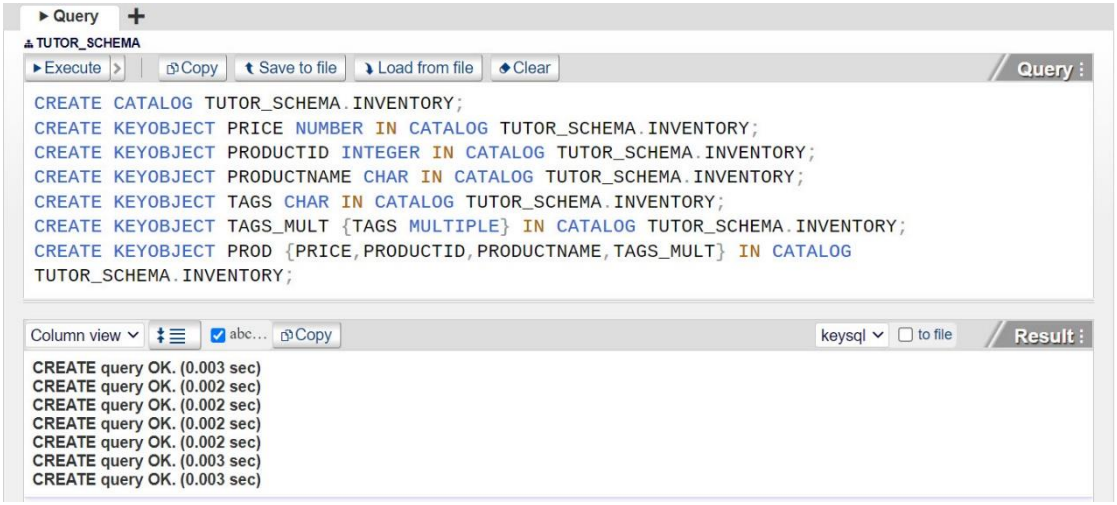
When the **Process** button is clicked, KeySQL Studio Import composes KeySQL statements that generate a KeySQL catalog compatible with the data format given by the JSON Schema file. The statements are displayed in the Schema pane where they can be reviewed before execution:



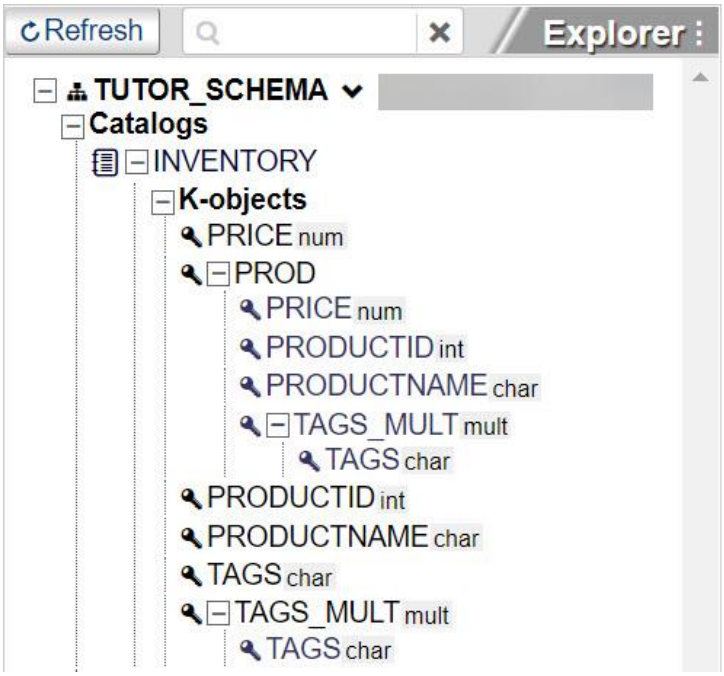
You have two courses of action:

- **Execute** the statements, immediately adding these k-objects to a catalog
- **Cancel** which returns you to the Import pane, taking no action for now, but allowing you to save and later execute the statements

Upon execution, the KeySQL statements are copied to Execution pane Query section and automatically executed:



When fully expanded, you can see how the catalog resembles the JSON Schema file:

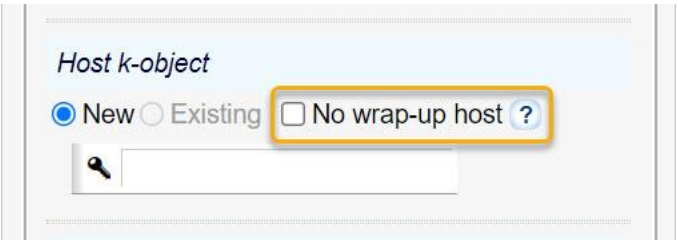


At this point, JSON data files compatible with the JSON Schema file can be imported via the “Load data to existing k-objects” option.

3.3.6 No Wrap-up host Option

The **No wrap-up host** advanced option allows for the straightforward import of JSON that already has a host k-object, in which case this option should be checked:

Note that when unchecked, the familiar default setting, you must specify the host k-object:



The three data scenarios for selecting the **No wrap-up host** setting are the import of files where there is:

- single instance
- single instance comprising multiple documents of the same host ("msg")
- array of instances with distinct host (e.g. "KWD", "BHD", "OMR", "JOD")

Examples for all three are provided below.

3.3.6.1 SINGLE INSTANCE

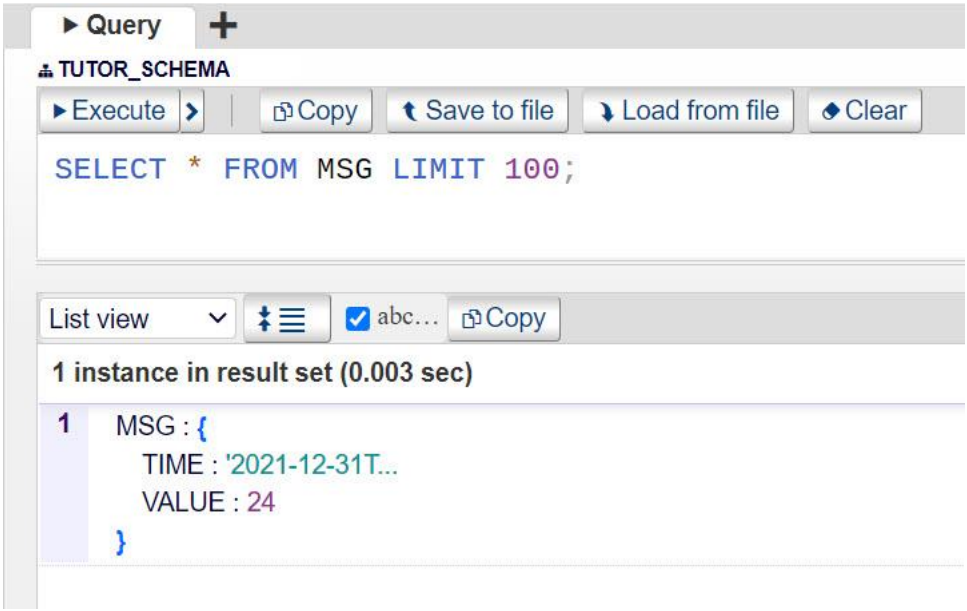
Consider a JSON file with a single instance containing a single document:

```
{
        "MSG":{
                "TIME":"2021-12-31T23:00:00Z",
                "VALUE":24.0
```



```
        }
}
```

By checking **No wrap-up host** you instruct Import to create an instance with “msg”, provided in the JSON data, serving as the host object:

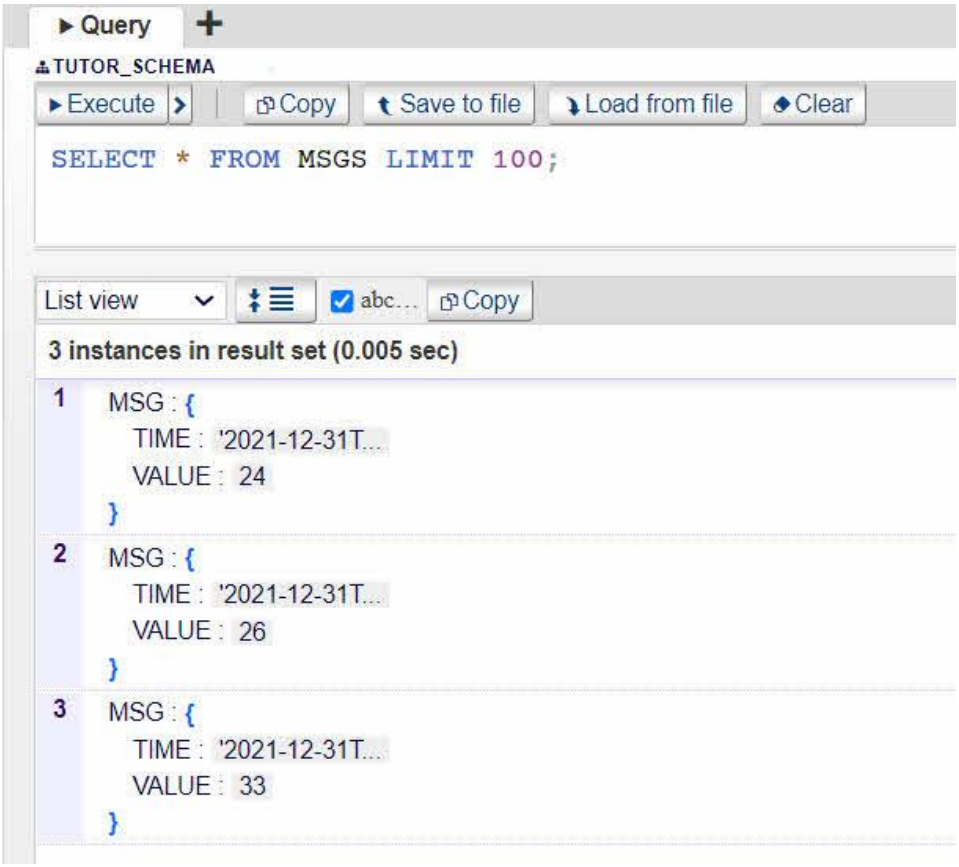


3.3.6.2 SINGLE INSTANCE COMPRISING MULTIPLE DOCUMENTS OF THE SAME HOST

Consider a JSON file with a single instance comprising multiple documents of the same host (e.g., "msg"):


```
[{
	"MSG": {
		"TIME": "2021-12-31T23:01:00Z",
		"VALUE": 24
	}
}, {
	"MSG": {
		"TIME": "2021-12-31T23:02:00Z",
		"VALUE": 26
	}
}, {
	"MSG": {
		"TIME": "2021-12-31T23:03:00Z",
		"VALUE": 33
	}
}]
```

By checking **No wrap-up host** you instruct Import to create an instance for each of the “msg” array elements, where “msg” serves as the host object:



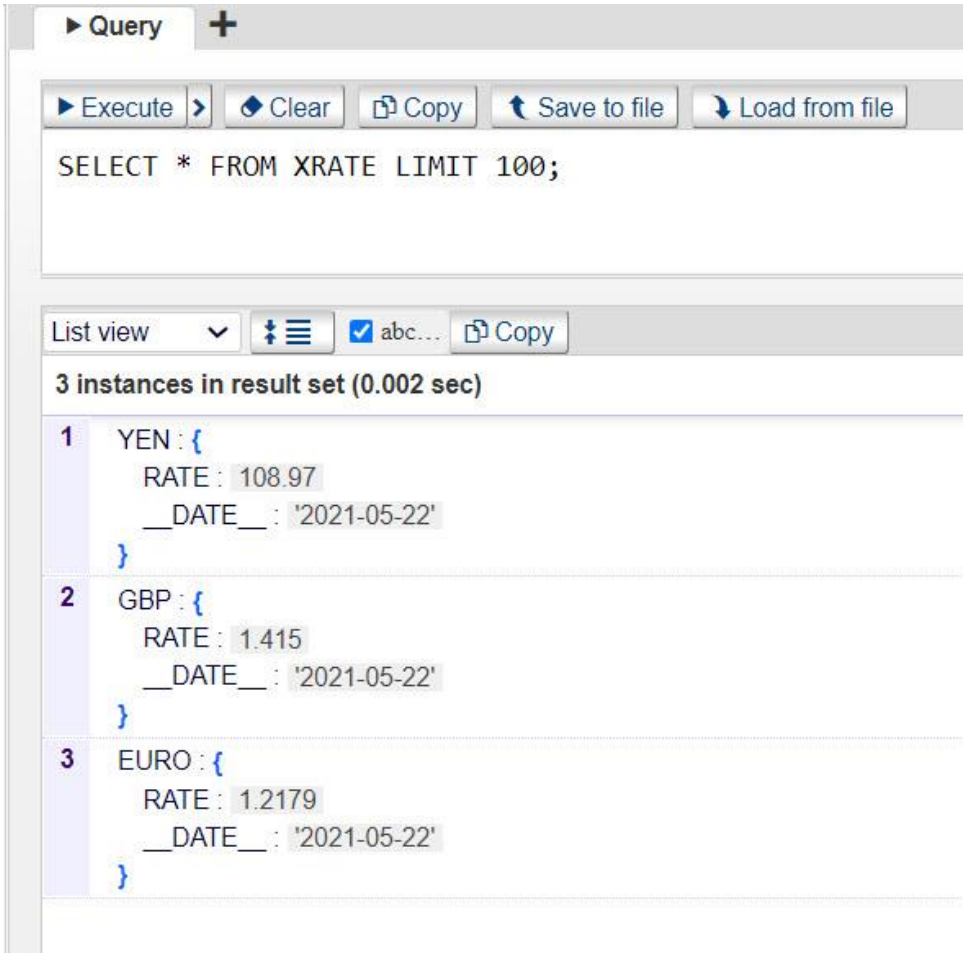
3.3.6.3 ARRAY OF INSTANCES WITH DISTINCT HOST

```
[{
        "YEN": {
              "DATE": "2021-05-22",
              "RATE": 108.97
        }
},
{
        "GBP": {
              "DATE": "2021-05-22",
              "RATE": 1.415
        }
},
{
        "EURO": {
```



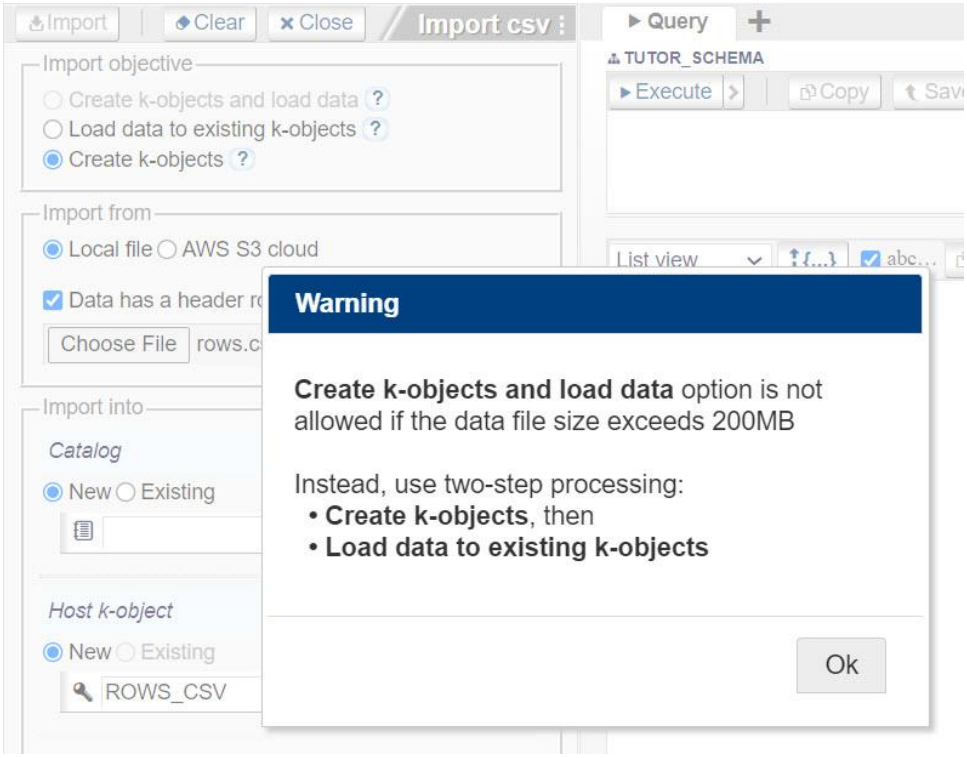
```
            "DATE": "2021-05-22",
            "RATE": 1.2179
        }
}]
```

By checking **No wrap-up host** you instruct Import to create an instance for each of the unique array elements:



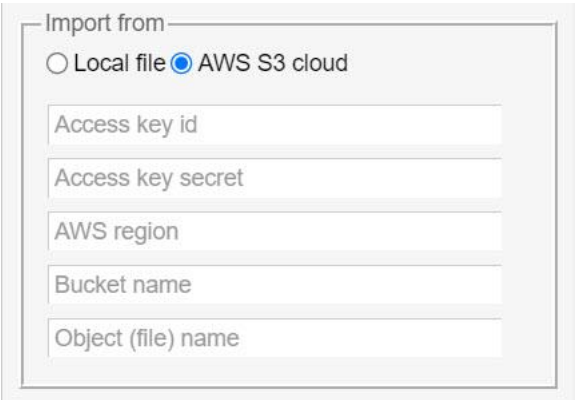
3.3.7 Import Limitations

Import using the convenient “Create k-objects and load data” option is available for files up to 200MB in size. When Import encounters a file larger than 200MB, it displays an advisory message that suggests the alternative two-step import:



3.3.8 AWS S3 AS IMPORT SOURCE

In addition to picking a file on your computer's local file system, you can also choose on the AWS S3 cloud file system. Selecting radio button AWS S3 cloud displays a form for accessing S3:



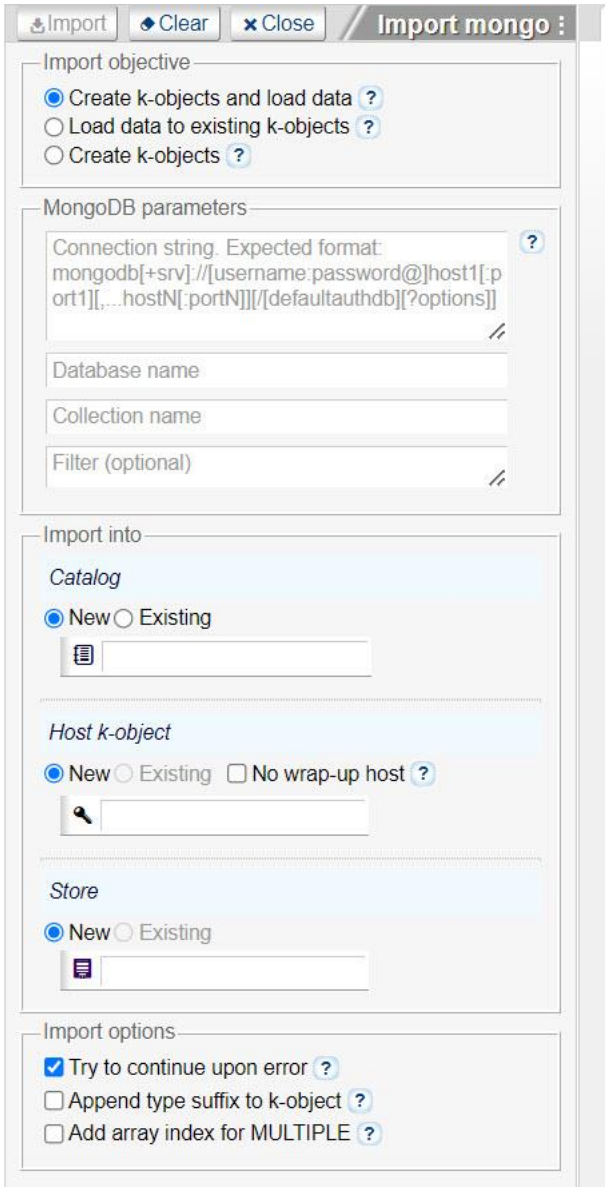
3.3.9 IMPORT OF COMPRESSED FILES

Studio's KeySQL, CSV, JSON, and XML import can digest archive files in both .ZIP and .GZ file format. Additional file formats will be added in the future.

3.4 IMPORT MONGODB

Studio can import data directly from a MongoDB database, including from MongoDB hosted in the cloud by Mongo Atlas.

3.4.1 GETTING FAMILIAR WITH MONGODB IMPORT PANE



This direct import is faster and easier than a two-step process, where a MongoDB collection is saved as BSON file, and that file is imported via Studio's JSON Import.

3.4.1.1 MONGODB PARAMETERS

The unique aspect of MongoDB import is populating the MongoDB parameters group. You need to specify:

- Connection string, which points to the MongoDB cluster and includes a username and password

- MongoDB Database name
- MongoDB Collection name
- Filter, an optional Mongo query language filter

COMPOSE THE CONNECTION STRING

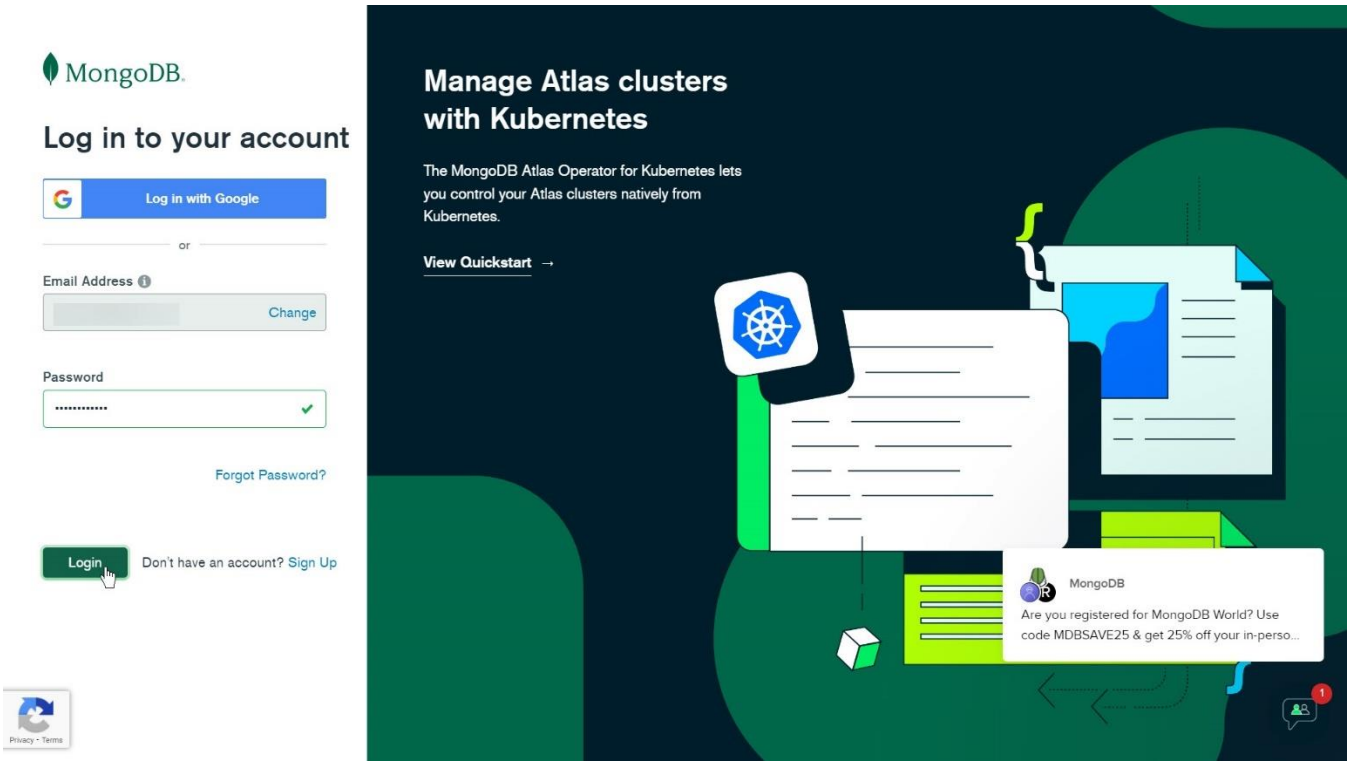
Here are the steps to compose the connection string, using an example for MongoDB Atlas. The initial steps are to fetch the connection string, which resembles this one from Atlas in the Azure cloud:

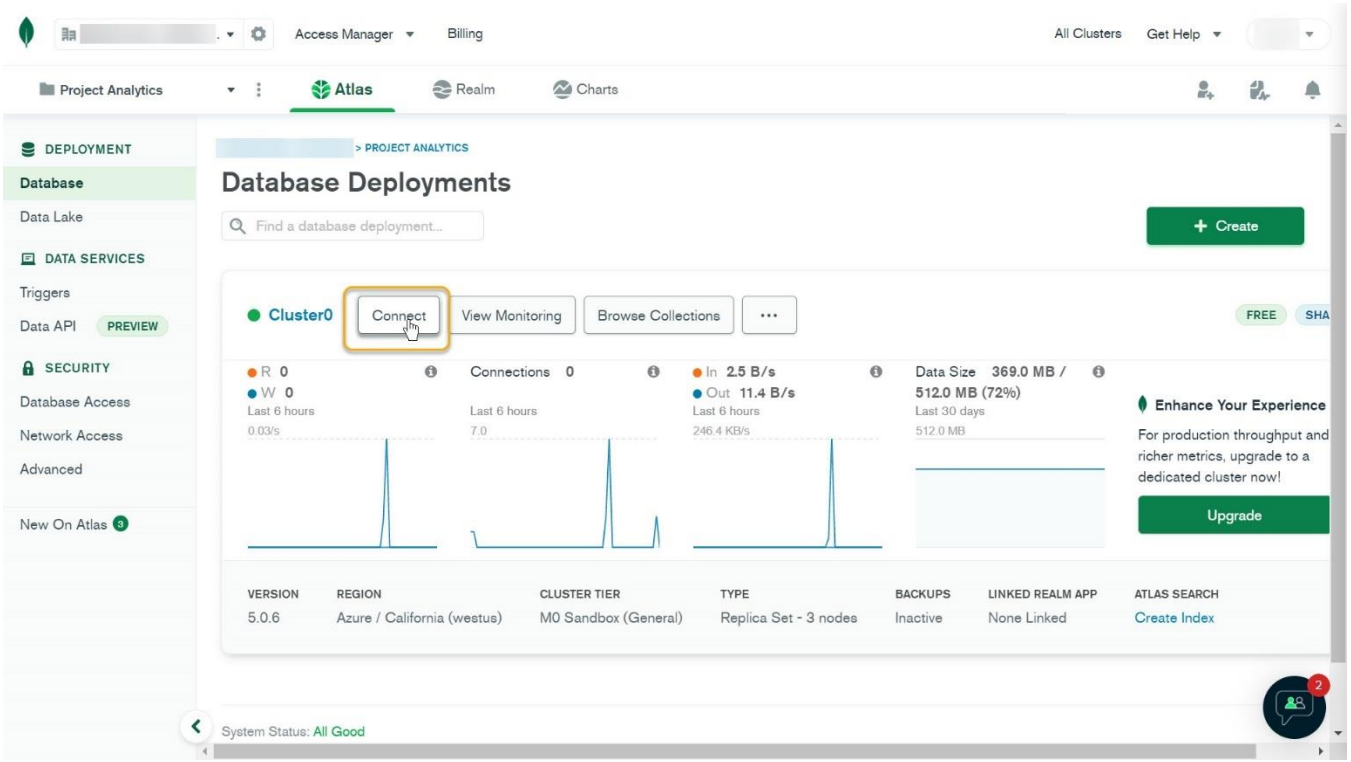
mongodb+srv://<username>:<password>@cluster0.8zsr4.azure.mongodb.net/test

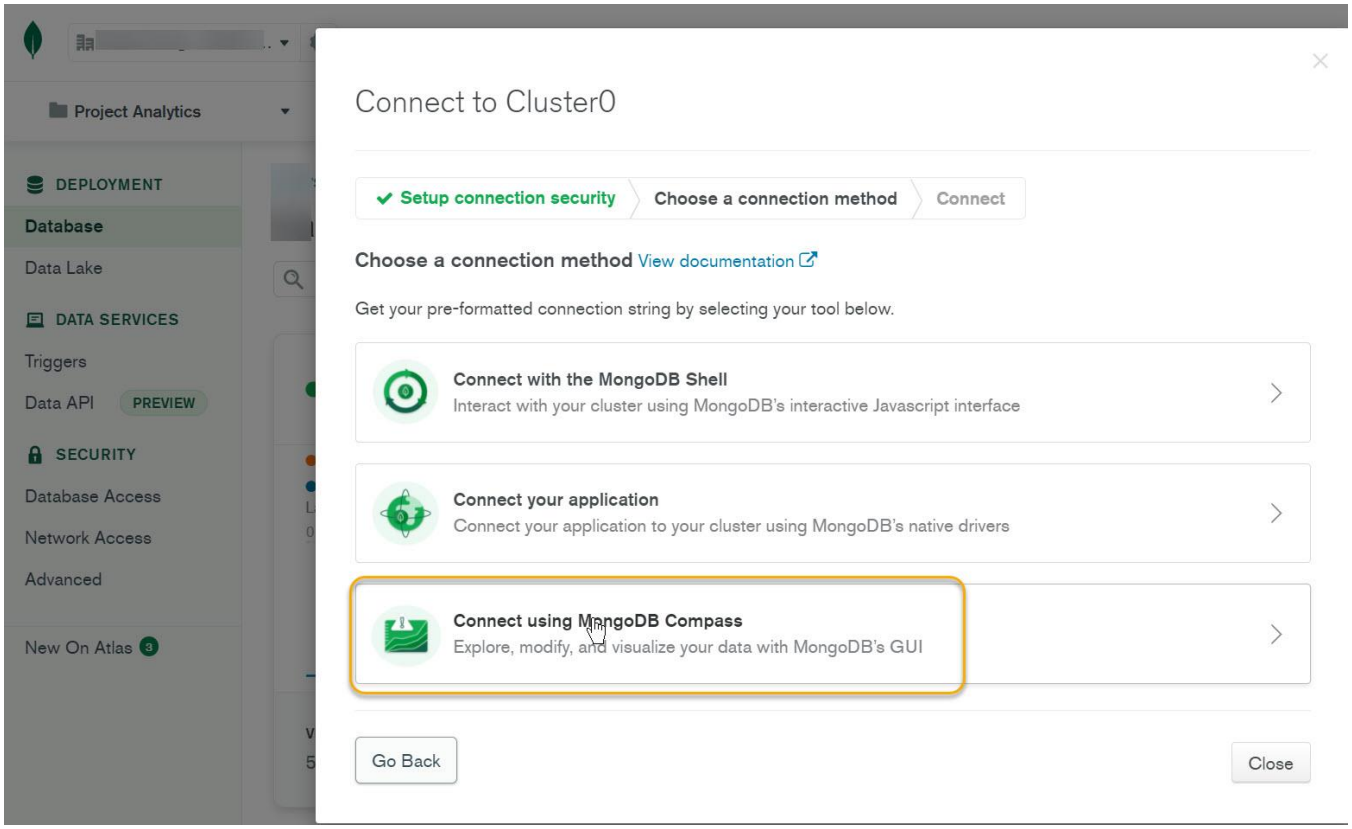
Screen shots for steps 1-4 follow:

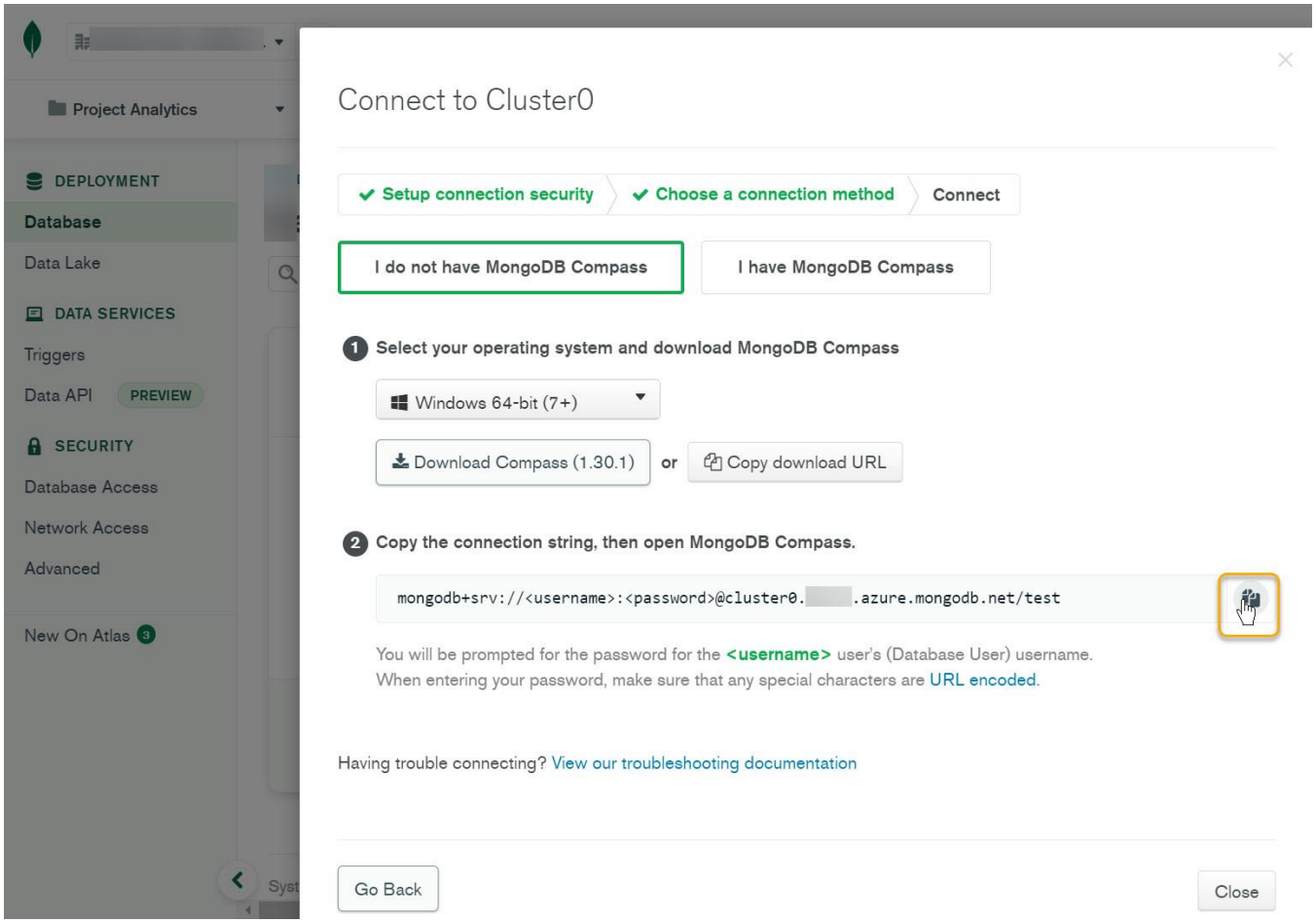
1. Log into **cloud.mongodb.com**
2. On the Database Deployments screen, select **Connect**
3. On the first Connect to Cluster screen, select **Connect Using MongoDB Compass**
4. On the second Connect to Cluster screen, select the copy icon for (2) Copy the connection string, then open MongoDB Compass
5. Replace <username> with your Mongo username
6. Replace <password> with your Mongo password

Steps 1-4:



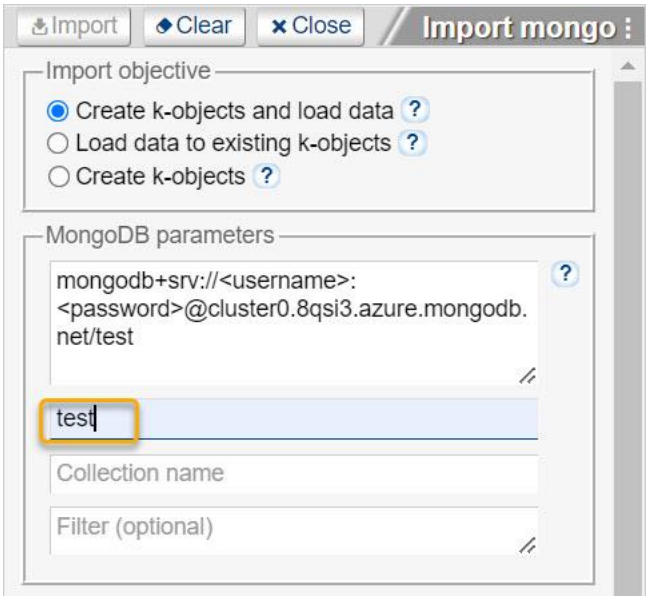






POPULATING THE MONGODB PARAMETERS GROUP

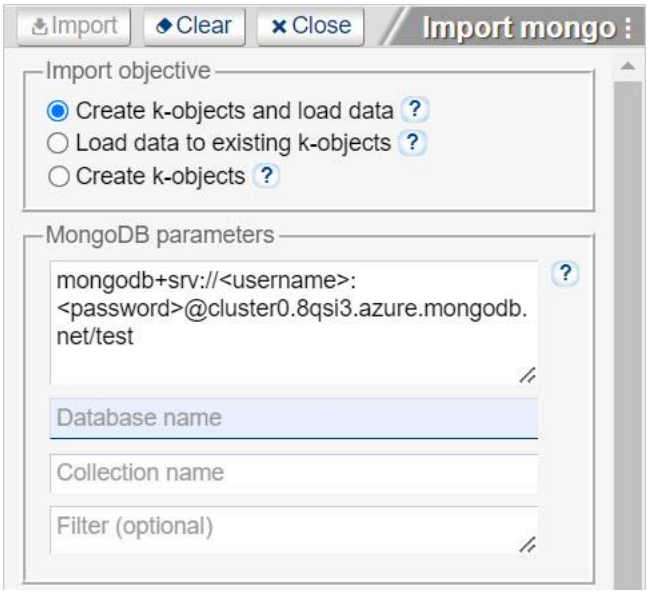
The screenshot below shows the situation immediately after a connection string has been pasted:



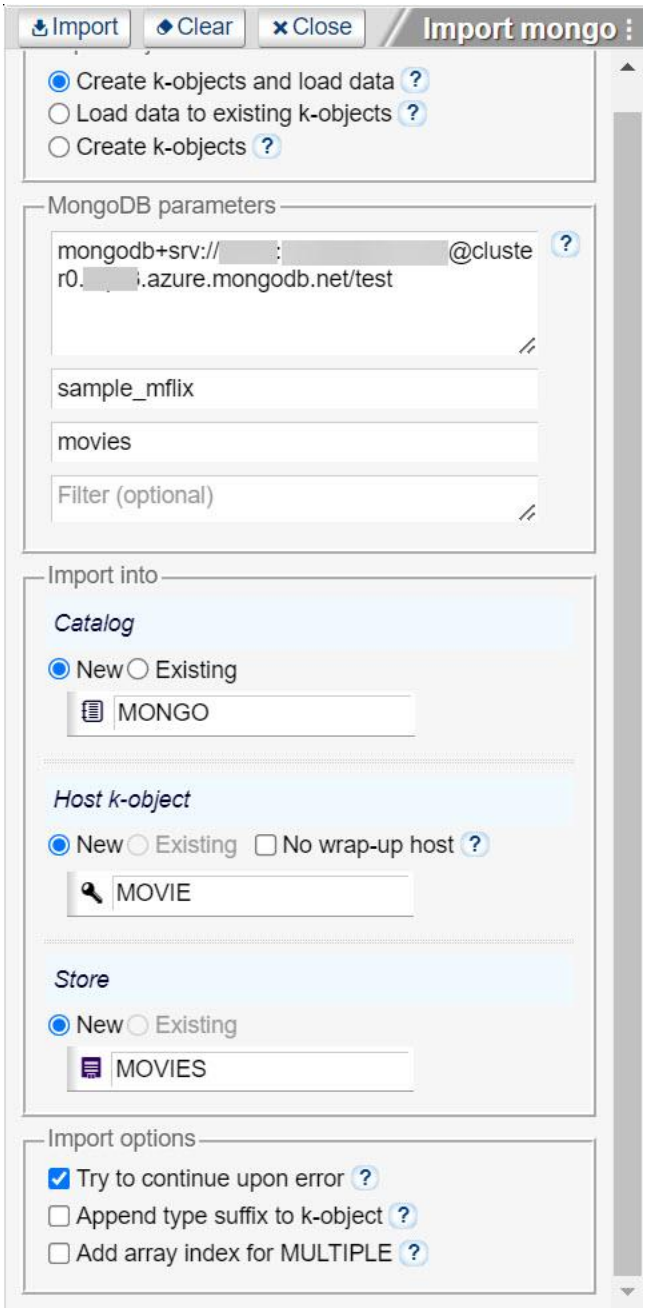
The <username> and <password> still need to be replaced with your values.

Note the word **test** in the 2nd input field. The Import form assumes that the database name at the end of the string (/test) is where your database documents are found, and automatically populates the **Database name** input field with this value. In the case of MongoDB Atlas, the database name at the end of the strong is used for authenticating credentials, not storing your data.

When using Atlas, you should clear the Database name field (as shown in the screenshot below) and then specify the name of the database where your collection is found:



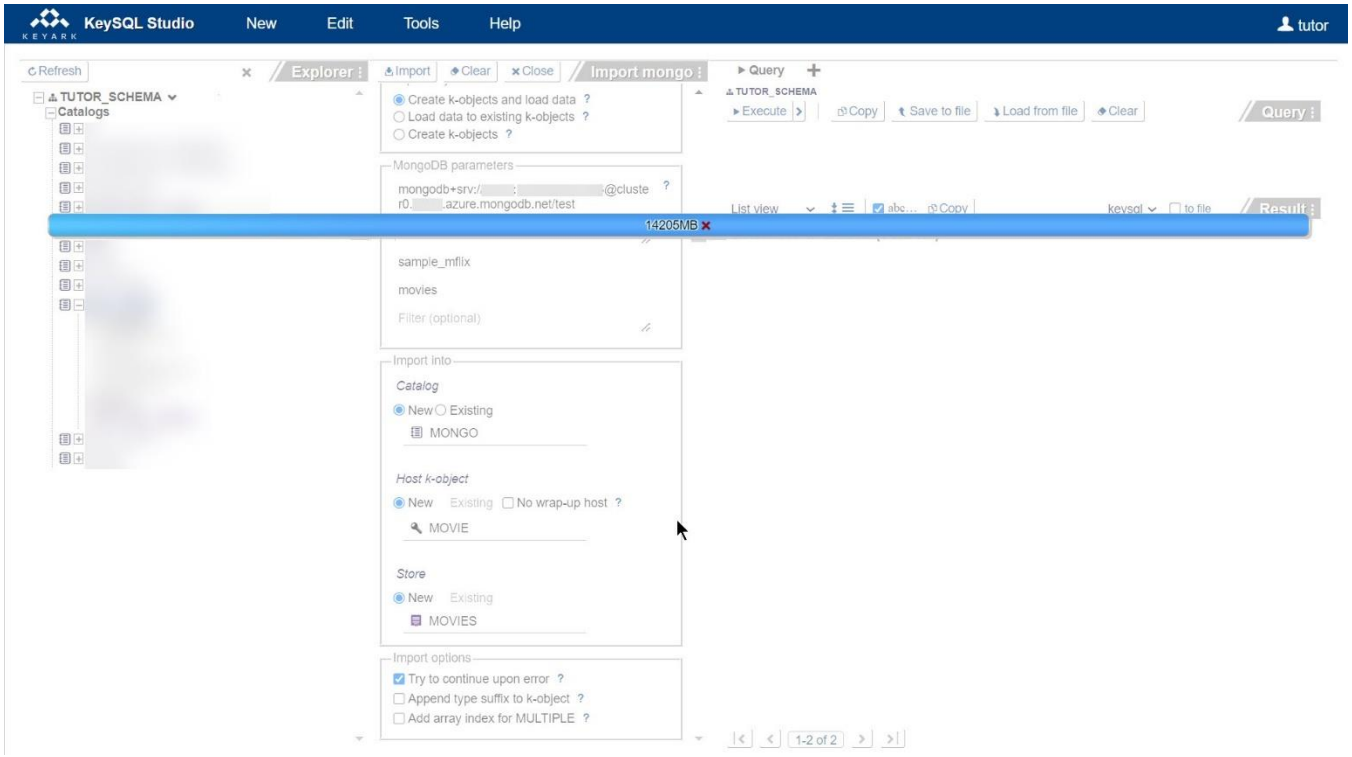
The next screenshot shows a fully populated Import mongo form, which is setup to import the **movies** collection from the **sample_mflix** sample database:



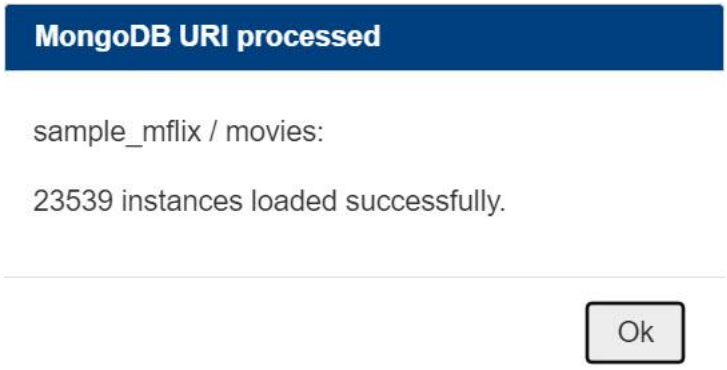
As a reminder, the MongoDB database name and collection name are case sensitive.

EXECUTING THE MONGODB IMPORT

When you click the Import button, Studio displays a blue banner which shows the progress. To interrupt the import, click the red x in the middle of this banner:



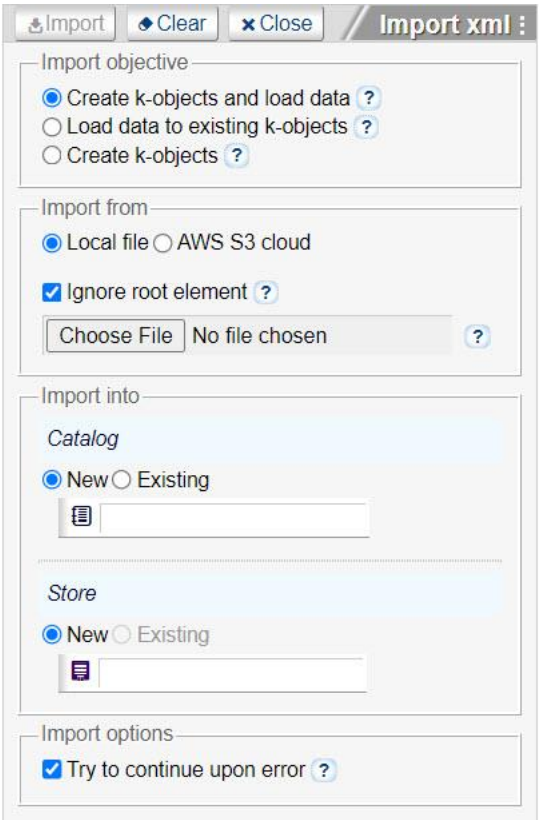
When the import finishes, Import displays a status screen:



3.5 IMPORT XML

Studio imports data in the popular XML format.

3.5.1 Getting Familiar with XML Import Pane



Unique to XML Import is the option **Ignore root element**. An example of where it is desirable to ignore the root element is shown in the following sample XML file, where we desire to ignore <CATALOG> and use <CD> as the host object:

```
<?xml version="1.0" encoding="UTF-8"?>
<CATALOG>
     <CD>
          <TITLE>Empire Burlesque</TITLE>
          <ARTIST>Bob Dylan</ARTIST>
          <COUNTRY>USA</COUNTRY>
     </CD>
     <CD>
          <TITLE>Empire Burlesque</TITLE>
         <ARTIST>Bob Dylan</ARTIST>
         <COUNTRY>USA</COUNTRY>
     </CD>
</CATALOG>
```

Deselect this option if the XML root element should serve as the host k-object.

Note: The <?xml > element, as shown above, is not mandatory.

There are a few limitations for XML Import:

- Files in excess of 200MB must be imported in a two-step process, same as for JSON file import.
- The data type is inferred; Import does not recognize either XML DTD or an XML Schema file.
- XML attributes are imported but elements are ignored. For example, the element **id="p101"** will be ignored:

```
<note id="p101">
    <to>Chris</to>
    <from>Rock</from>
    <heading>Reminder</heading>
    <body>Don't forget to bring wine this weekend!</body>
</note>
```


4 EXPORT

KeySQL Studio export commands can be executed by utilizing either:

- Tools menu
- Explorer context menu

4.1 NATURE OF EXPORTED FILES

KeySQL Studio exports two types of information:

- Catalog and store definition statements that define (and recreate) catalogs, stores, and k-objects that comprise these
- Store data, that is data you would use to populate a store

4.1.1 CATALOG AND STORE DEFINITION

For both user interfaces, begin by using Explorer to first select a catalog, and optionally a store.

Upon selection of catalog, you can export (save to a text file) statements that create the catalog and/or the stores in that catalog, and all the k-objects that comprise the catalog. These ready to execute statements are called KeySQL statements and are specific to KeySQL Server.

4.1.2 STORE DATA

Upon selection of a store, you can export data for that store to a file. You can choose from three formats:

- Export to KeySQL, which results in ready to execute KeySQL statements, specific to KeySQL Server
- Export to JSON, which results in a data file in JSON format
- Export to CSV, which results in a data file in CSV format

The CSV option does not appear if the data is too complex for flattening.

4.1.2.1 CHOOSING AMONG THE DATA FORMATS

A natural question is which file type is preferable for the store data export. Here is guidance:

File Type	Complexity Handled	Fidelity of Information Preservation	Universality
KeySQL	Very High	Very High	Import/Export among KeySQL Servers only
JSON	High	Medium There is no date format defined for JSON	JSON is ingested by most DBMS today, and some applications.
CSV	Low complexity Cannot express data where a field has multiple values (such as a phone field with several phone numbers); hierarchies are flattened.	Poor Loss of data types	Both Microsoft Excel® and virtually any software application that supports import, can ingest CSV files.

4.2 EXPORT COMMAND USER INTERFACE

For your convenience the same export commands are accessible by your choice of path:

- Tools Menu
- Explorer Context Menu

Both are explained in the sections that follow.

FYI: Depending on your web browser setting for downloads, when you go to save a file either a standard dialog will pop-up, letting you pick the file name, or the file will be immediately written to your download folder and given a default filename.

4.2.1 EXPORT VIA TOOLS MENU

The Tools Export menu visually confirms your selection of catalog and store in Explorer, displaying the catalog and store name (if one is selected), and available menu options. In this screenshot the catalog UNIVERSE is selected, and the store BILLIARD_STORE is also selected:

4.2.2 Export via Explorer Context Menu

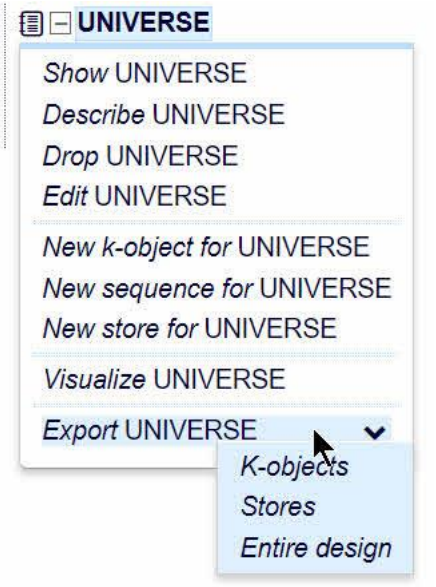
The Explorer context menu shows the export file format options specific to the catalog or store you select:

4.2.3 Dynamic Menu Options for Data

The screenshot above shows that for BILLIARD_STORE your options are **to KeySQL** and **to JSON** but not **to CSV**. CSV export is available only if the data can be written in CSV format. Should the k-object selection include the MULT data type, or multiple host k-objects, the data cannot be flattened, and consequently CSV export is not allowed and accordingly not shown in the menu.

FYI: Should a store be empty, none of these menu options will appear.

4.3 CATALOG AND STORE DEFINITION EXPORT



When a catalog is selected, KeySQL Studio can create a file containing KeySQL statements for recreating the catalog and store design, as shown above, and explained in this table:

Export Menu Choice	**Purpose**
Catalog <catalog-name>	
K-objects	Export KeySQL statements that define the k-objects that constitute that catalog
Sequences	Export KeySQL statements that define sequences (which apply to k-objects). This menu choice appears only if sequences exist for this catalog.
Stores	Export KeySQL statements that define the stores for that catalog, and any constraints and hosts that may exist for those stores
Entire design	KeySQL statements for the entire design including everything provided by the export for k-objects, sequences, and stores

4.4 STORE DATA EXPORT

Upon selection, KeySQL Studio will create a file containing statements that hold data for populating stores and sharing data with other applications:

Export Menu Choice	**Nature of Commands**	**Always available?**

Store <store-name>		
to KeySQL	KeySQL statements for inserting data into a store	Yes
to JSON	JSON file that can be imported by document databases and many applications	Yes
to CSV	CSV file that can be imported by most any application including Microsoft Excel (R)	Only when the data can be viewed in rows and columns as seen in a spreadsheet. If the data is too complex, such as repeating (multi-composite) values or contains multiple host k-objects, this menu option will be hidden.

FYI: Should a store be empty, none of the three menu options will appear.

4.5 EXPORT EXAMPLES

4.5.1 CATALOG AND STORE DEFINITION EXPORT – EXAMPLE

Below is a sample of two KeySQL statement files (given a file extension of KSQL) that were created by an export. The first defines the catalog. The second defines the stores in that catalog. These files can serve as either a backup of your catalog and store definition, or to create similar ones.

Command	Filename	Purpose	File Content
Catalog Universe k-objects	catalog_UNIVERSE.ksql	Create the catalog and add definitions of KeySQL objects hosted in that catalog	CREATE CATALOG UNIVERSE; CREATE KEYOBJECT COLOR CHAR IN CATALOG UNIVERSE; CREATE KEYOBJECT NUM NUMBER IN CATALOG UNIVERSE; CREATE KEYOBJECT BALL {COLOR,NUM} IN CATALOG UNIVERSE; CREATE KEYOBJECT BALLS {BALL MULTIPLE} IN CATALOG UNIVERSE; *– trimmed –*

Catalog Universe Stores	catalog_stores_UNIVERSE.ksql	Create stores in the catalog that can host k-object instances (data)	CREATE STORE BILLIARD_STORE FOR CATALOG UNIVERSE; CREATE STORE OFFICE_RENTALS FOR CATALOG UNIVERSE;

4.5.2 Data Export - Examples

4.5.2.1 KeySQL Data Export - Example

Below is a sample of a KeySQL file export that can serve to back up your store data:

In the table below is shown a sample of KeySQL data manipulation statements (the INSERT) that are found in a backup file, which can be used to repopulate a store (in this case BILLIARD_STORE) with data.

Command	Filename	Purpose	File Content (extract)

Store Billiard_Store	stores_UNIVERSE.ksql	Create KeySQL Insert statements that populate a store with k-object values	INSERT INTO BILLIARD_STORE INSTANCES {
			BALLPACK : {
			BALLSET : {
			DIAM_MM : 61.5,
			BALLS : {
			BALL : {
			COLOR : 'red',
			NUM : NULL
			},
			BALL : {
			COLOR : 'white',
			NUM : NULL
			},
			BALL : {
			COLOR : 'white_spot',
			NUM : NULL
			}
			},
			QTY : 3,
			STYLE : 'carom'
			},
			BRAND : 'Carambo',
			NAME : 'Standard_Carom',
			UPC : 123456789012
			}
			},
			
			– trimmed --

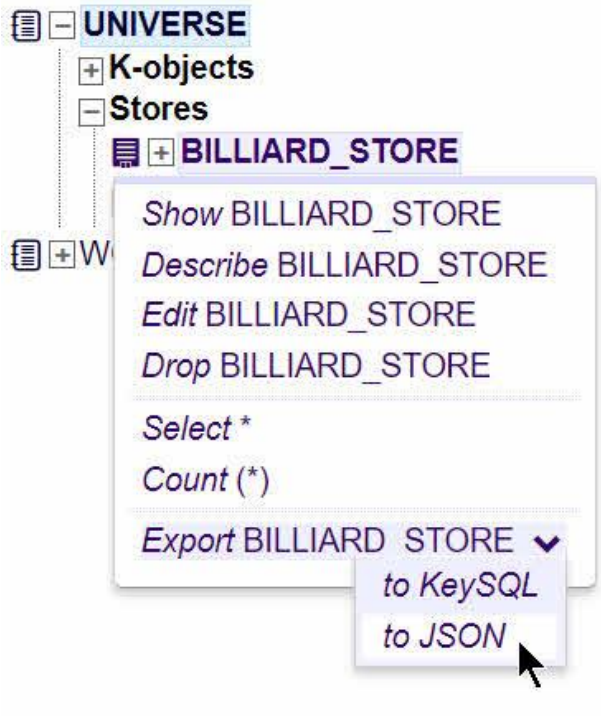
FYI. The store data export file begins with a CREATE STORE statement as shown here for a tiny export file:

CREATE STORE MSG FOR CATALOG MSG;
INSERT INTO MSG INSTANCES {MSG:{TIME:'2021-12-31T23:01:00Z',VALUE:24}}

The catalog definition export file also contains a CREATE STORE statement. Should your goal be to restore data, and the store already exists in the catalog, you will get an error message. In this situation, drop the store from the catalog and rerun the store data export file.

4.5.2.2 JSON DATA EXPORT - EXAMPLE

Below is a sample of a JSON document export that can be used to transport data to another database system via the universal JSON file format:



4.5.2.3 CSV DATA EXPORT – EXAMPLE

Below is a sample of a CSV document export that can be used to transport data to Microsoft Excel (R) or another application.

Note how the **to CSV** export menu option is available when the store contains only flat data as shown here:

```
AMOUNT,TRANSACTION_DATE,TRANS_ID,VENDOR
6,"11/7/2020","RE3829-SF","PayByPhone - SF"
70,"11/8/2020","20383939331","SFMTA*Street Cleaning"
7.36,"11/12/2020","Z39as098fs","JoesIceCream-Geary"
32,"11/16/2020",,"Moscow & Tbilisi Bak"
```

Here are a few details on how KeySQL Studio exports in CSV format:

- A header row is written based on the store's k-object names
- Data stored with a k-object data type of NUMBER are exported as numbers
- Data stored as a null value is written as seen in the last row, where the TRANSACTION_DATE is null, and accordingly there are two commas with nothing in between

4.6 FILENAME CONVENTION

4.6.1 CATALOG EXPORT

Export Menu	Filename

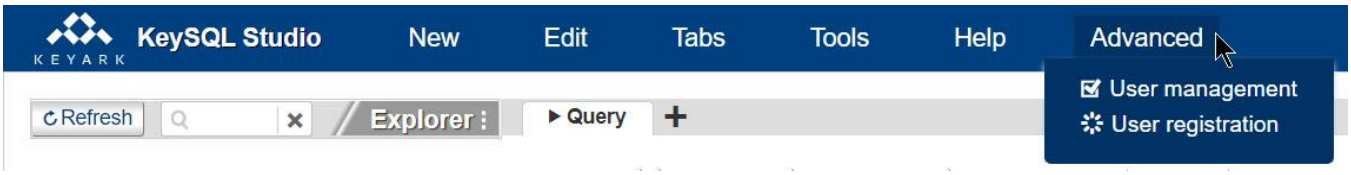
K-objects	<SCHEMA_NAME>_CATALOG NAME>_kobjects.ksql
Sequences	<SCHEMA_NAME>_CATALOG NAME>_sequences.ksql
Stores	<SCHEMA_NAME>_CATALOG NAME>_stores.ksql
Design	<SCHEMA_NAME>_ CATALOG NAME>_design.ksql

4.6.2 Store Data Export

Export Menu	**Filename**
KeySQL	<SCHEMA NAME>_store_<STORE NAME>.ksql
JSON	<SCHEMA NAME>_store_<STORE NAME>.json
CSV	<SCHEMA NAME>_store_<STORE NAME>.csv

5 ADMIN ADVANCED MENU

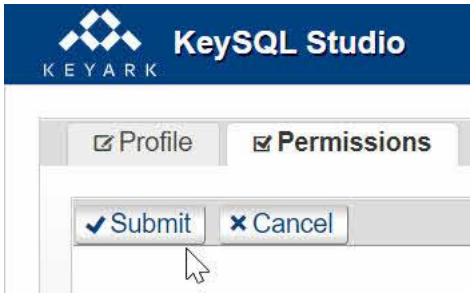
Only users with administrator privileges will see an additional **Advanced** menu option to the right of **Help**:



This provides two additional areas for KeySQL Server user administration:

- User management (manage existing user accounts)
- User registration (create and setup new user account)

After making changes, make certain to click **Submit** (or Cancel):

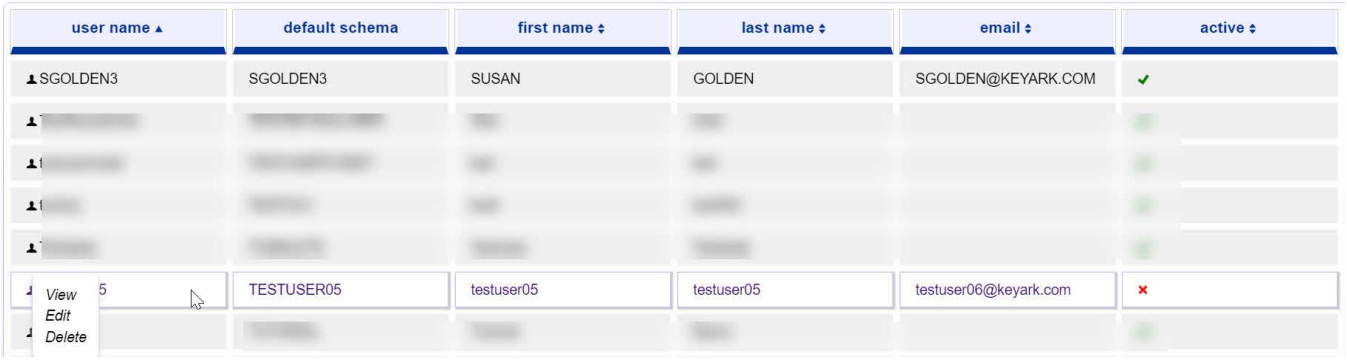


FYI: To save your work, the proper sequence is:

1. create (or alter) Profile
2. create (or alter) Permissions
3. click the **Submit** button

5.1 USER MANAGEMENT

You can view, alter and delete existing user accounts by selecting User Management from the Advanced menu. The list of existing user accounts is displayed:



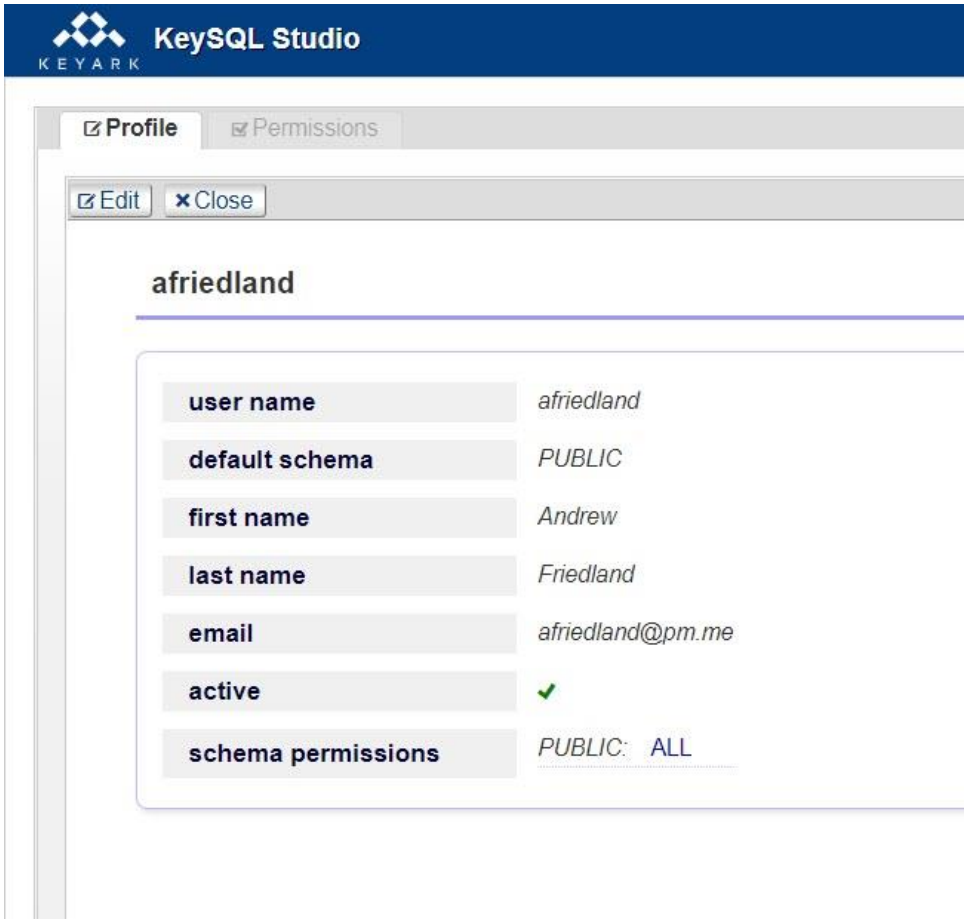
In the example above, there is one inactive user and many active users.

To view the Profile and Permissions for a user, left click on the row. This causes a small context menu to appear on that row. The equivalent buttons on the top become live as well. You can use either to accomplish your task:

5.1.1 VIEW/EDIT/DELETE

5.1.1.1 VIEW

Select **View** displays the user's Profile in a view only mode, so there is no risk of accidental changes:

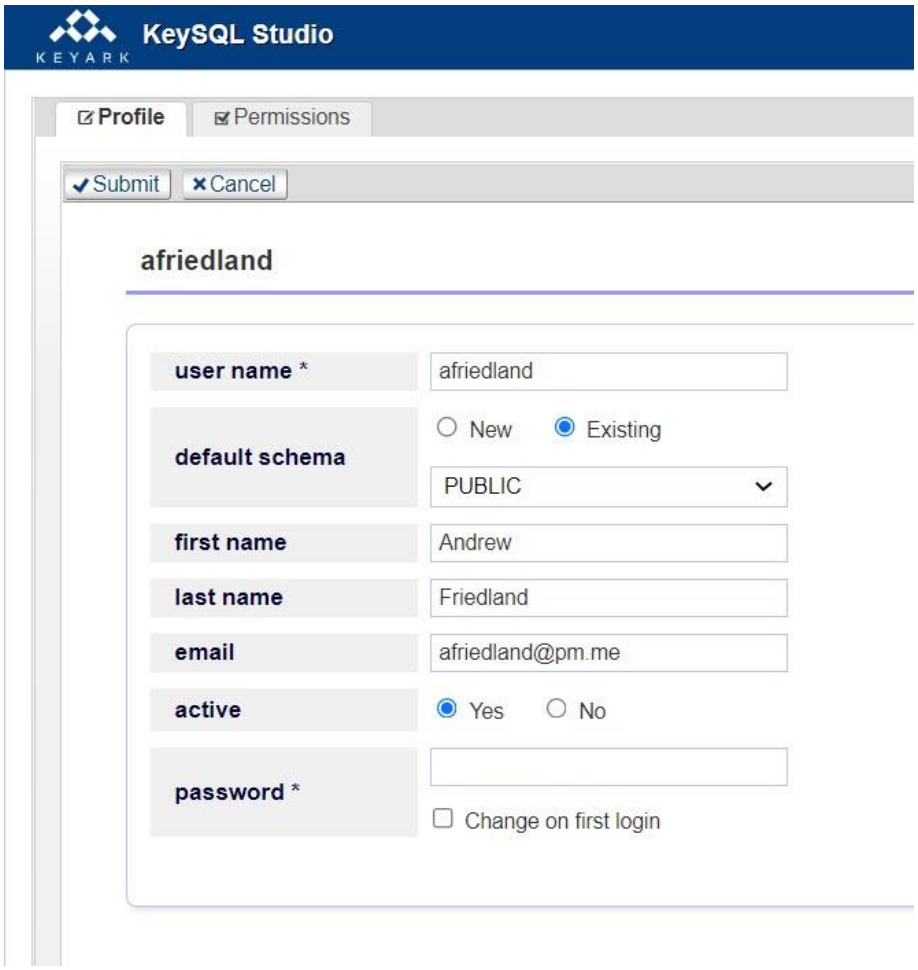


The profile fields are described in detail, section **5.2 User Registration**.

To edit, click the **Edit** button.

5.1.1.2 EDIT

When in Edit mode, the password related fields are additionally exposed:



Upon completion of any changes, click Submit to save, or Cancel to quit without changes.

5.1.1.3 DELETE

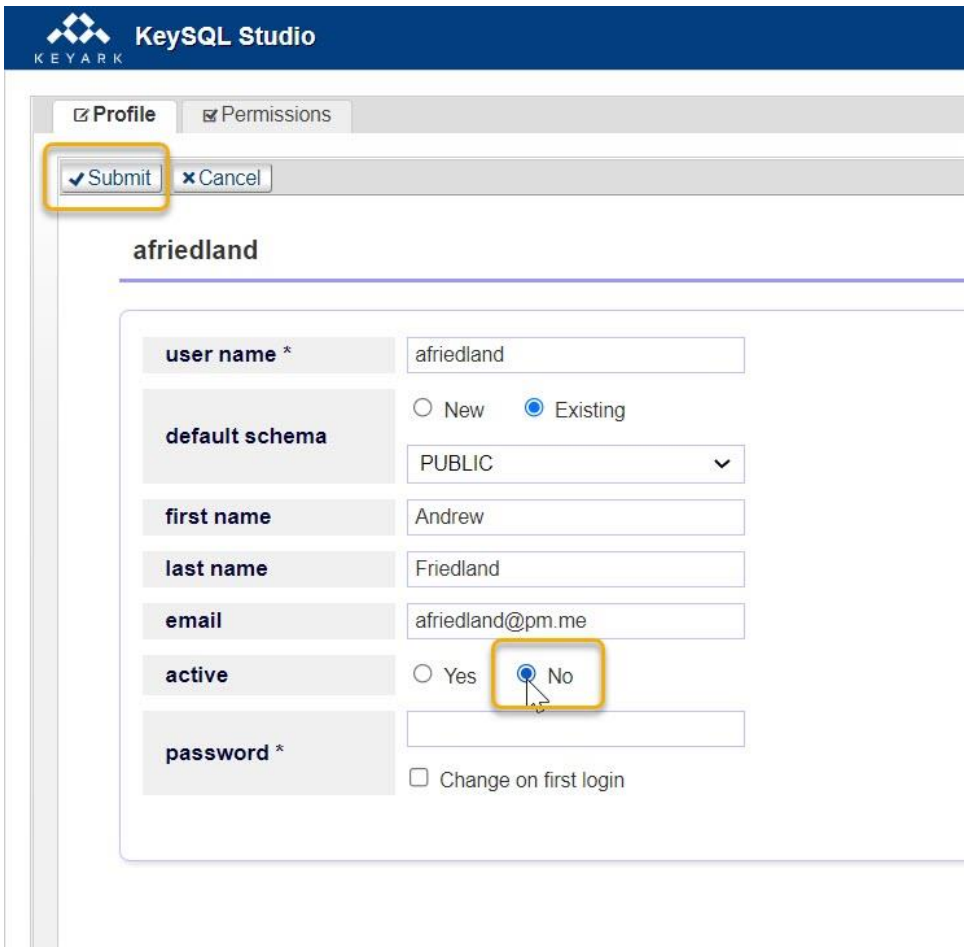
Deleting a user is a two step process, to avoid accidents:

- Edit the profile, marking user as inactive
- Give command to delete

These steps are shown for user **afriendland**.

Click on that user's row, and select Edit:

Click **No** on the **active** radio button, and then **Submit**:



Now that this user is inactive, select **Delete**:

Note that the Delete button, and the corresponding Delete context menu option, are live only for inactive users.

FYI: Schema's owned by a user are _not_ deleted when the user account is deleted. When you delete a user there is no risk of losing their data.

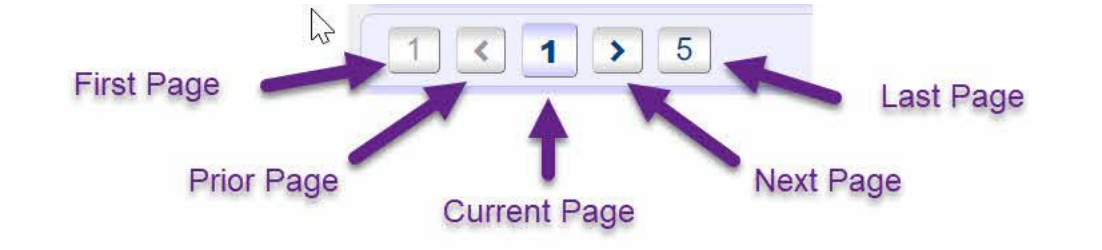
5.1.2 NAVIGATION

There are two ways to navigate through a long list of users:

- Scroll
- Find

5.1.2.1 SCROLL

You can quickly scroll through pages of users by utilizing the scroll control at the bottom left of the window:

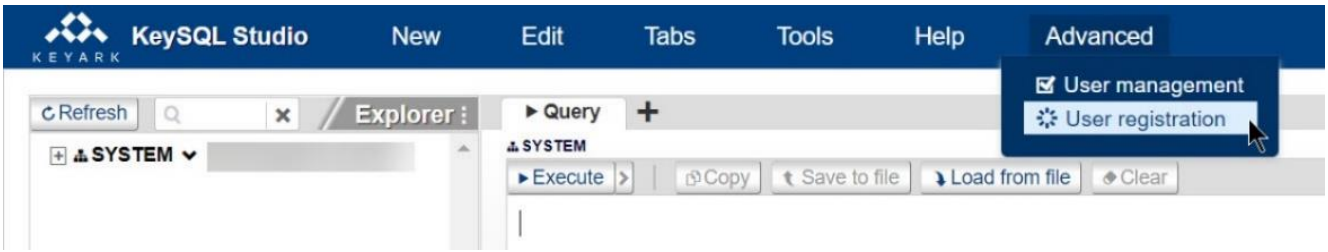


5.1.2.2 FIND

You can filter the list of user names by utilizing the Find textbox, which does either an exact or partial string match:

5.2 USER REGISTRATION

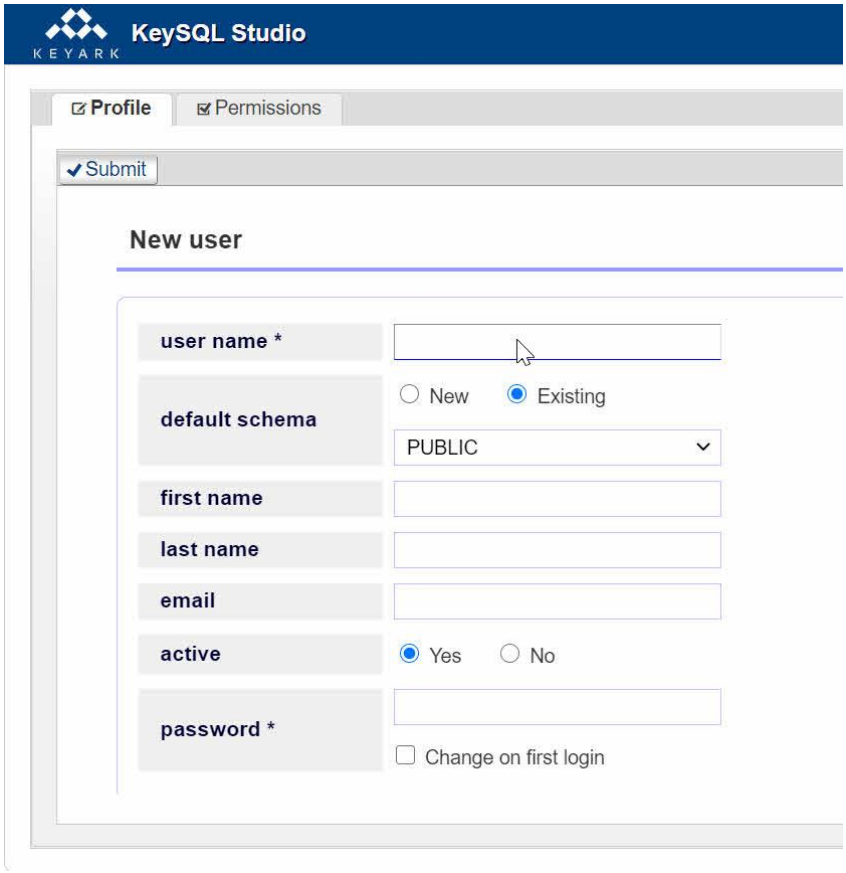
You begin the process of creating a new user account by selecting User registration from the Advanced menu:



Describe the new user by filling in the Profile tab. Then fill in the Permissions tab, and finally click the Submit button. These steps are described below.

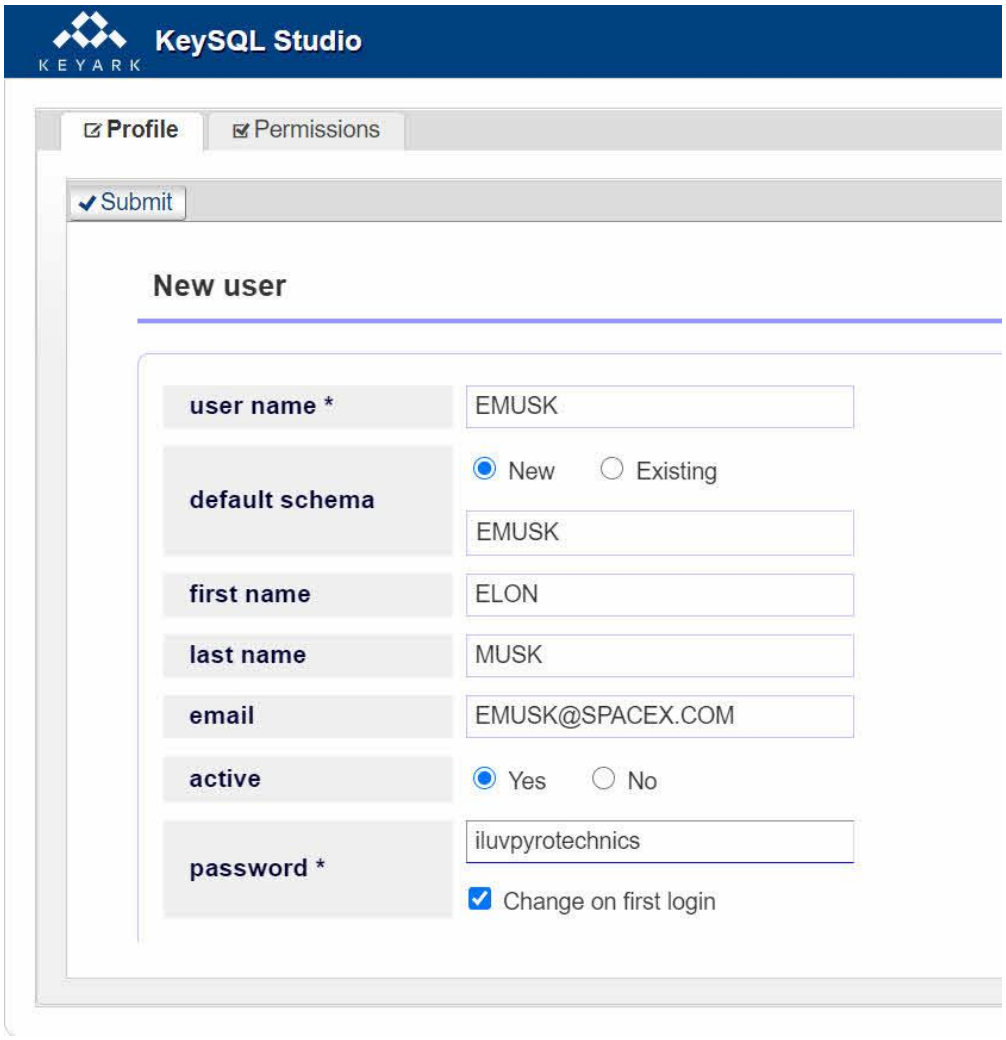
5.2.1 PROFILE TAB

Shown below is the initial state of the Profile tab. For a new user select the New button and fill in the fields:



.

The screenshot below shows an exemplary submission for an exceptional employee:



Important: If you click the New radio button, specifying a new schema, **you must proceed to the Permissions screen** to assign permissions to that schema, as will be shown below. You must specify the permissions *before* clicking the Submit button. This step both creates the new schema and assigns permissions.

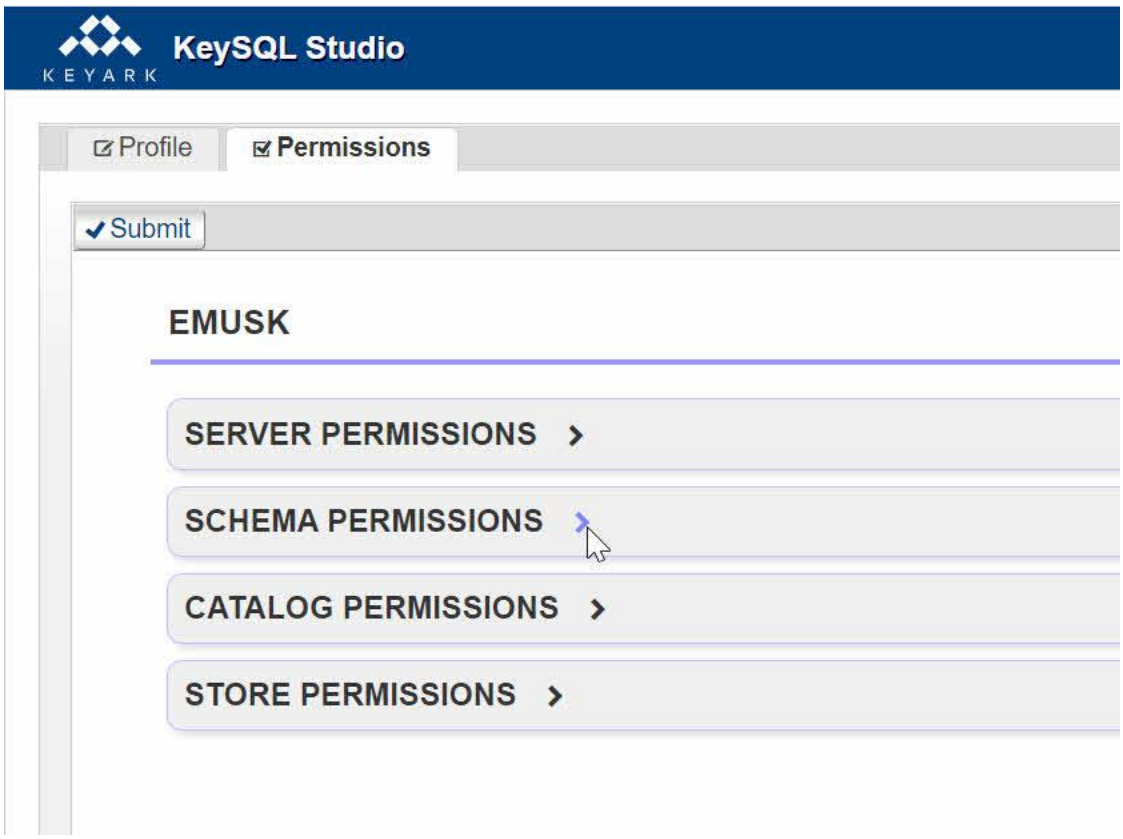
USER PROFILE FIELD EXPLANATION

user name	Typical user name without spaces. Capitalization is preserved, and shown on KeySQL Studio, but in this form the user name is displayed in all capitals. KeySQL Server is *insensitive* to user name capitalization during log-in.

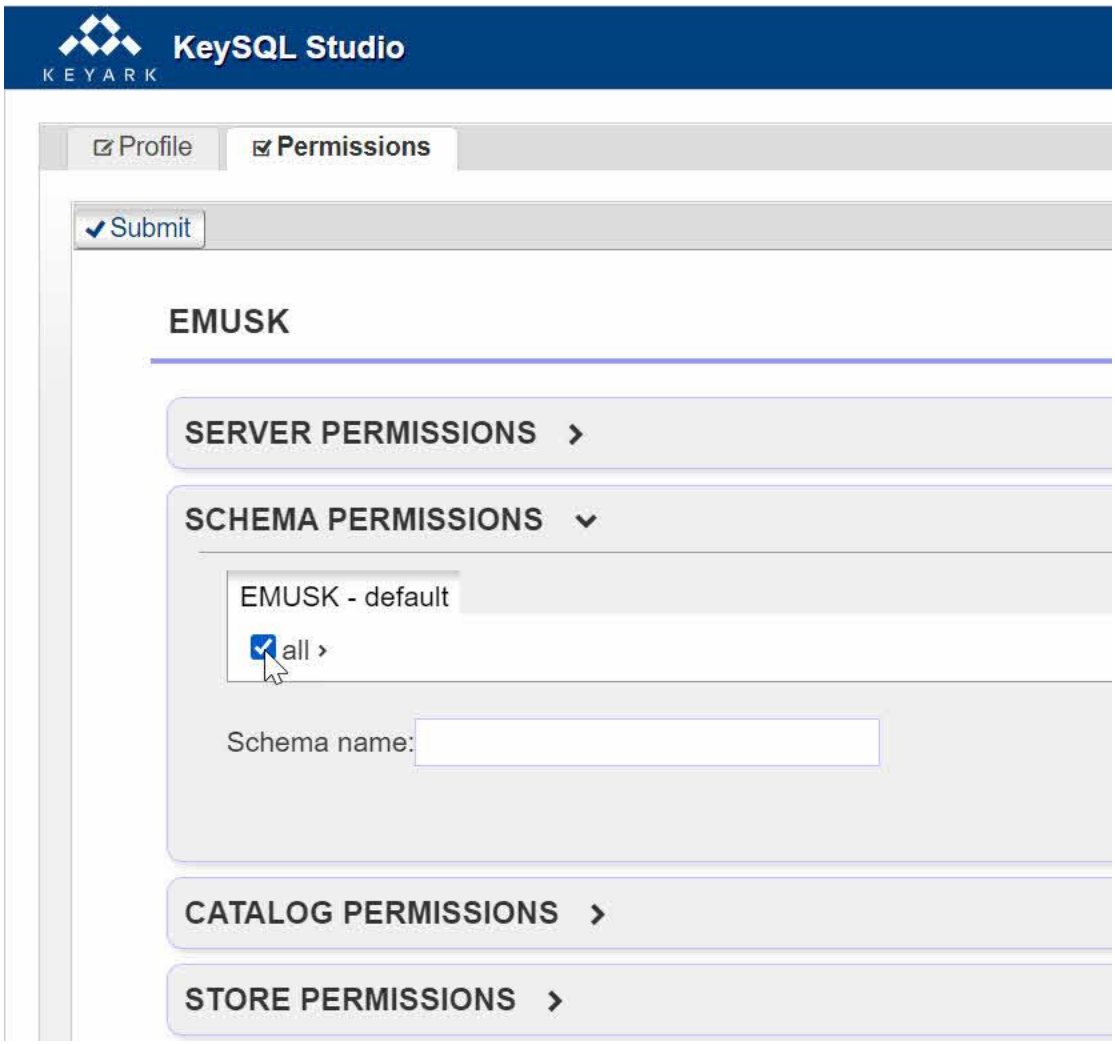
default schema	When the user first logs into KeySQL Studio, Explorer shows the catalogs and stores found in this schema. This field must be populated for a user to log in. The default schema is where user created catalogs and stores, and related data is placed, unless KeySQL command specify another schema for which the user has permission to access. Many organizations create a new schema for each new user, with a name same or like their username. Other organizations share an existing schema such as PUBLIC to be used as the default schema. It's also possible to do a mix of the two approaches.
first name	User's first name
last name	User's last name
email	Optional; can be used for account related notifications.
active	Controls whether this user account is active. If set to No, the user can no longer access KeySQL Server using this account. Generally, set to Yes for new users. But also, possible to set this to No, and then later (such as when user begins the position) to make this active.
Password Change on first login	The initial account password can be entered here. Note that KeySQL Server is *sensitive* to password capitalization. Typically, when a password is specified, the **Change on first login** checkbox is also checked. When done so, the user is forced to enter a confidential password upon their first login.

5.2.2 Permissions Tab

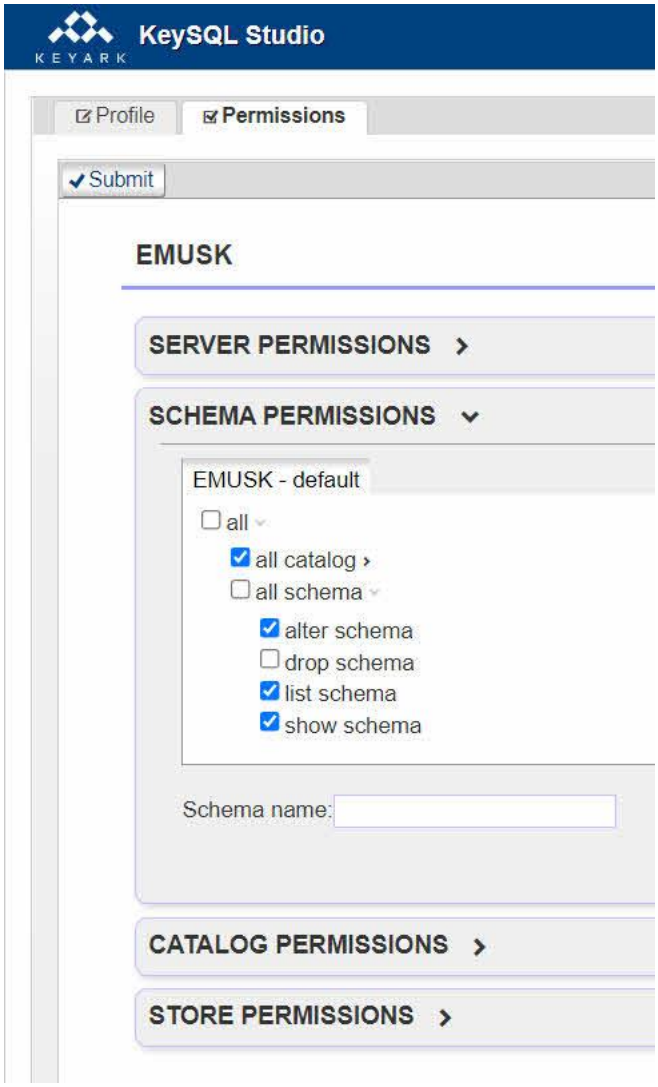
The typical new user set up requires specification of at least one default schema, as specified in the Profile tab. When the Permissions tab is displayed, the Schema (and other) Permissions are not yet visible:



Click on the › symbol to the right of the Schema Permissions to see the current settings. Once displayed you must click the **all** checkbox, or a set of specific permissions:



By clicking the › symbol to the right of all, you can be selective on the permissions. The process continues through the tree of permissions. As can be seen below, a common scenario is to grant all permissions but (accidental) deletion of the schema via the drop command:



To provide permissions to additional schema, click into the **Schema name** textbox. A list of all existing schemas will appear, and you can select among these.

FYI: The KeySQL Definitive Guide explains the schema, catalog and store permissions. In addition, the guide provides command line equivalents for the commands shown in this section of the KeySQL Studio Guide, plus many more.

6 APPENDICES

6.1 HOTKEYS

6.1.1 ACTIVE TAB HOTKEYS

Key Combination	Command
ALT+Q	Execute **Q**uery (and other) statements in the current tab, same as clicking the Execute button.
ALT+C	**C**lear both the Query and Result sections for the current tab, same as clicking the Clear button
ALT+H	Show request **H**istory (ESC to close)
ALT+X	Close current tab (except initial ‘Query’ one)

6.1.2 PAGE HOTKEYS

Key Combination	Command
ALT+R	**R**efresh Tree
SHIFT+→; SHIFT+←; SHIFT+↓; SHIFT+↑	Use the cursor keys to resize the Explorer pane or Execution pane sections: query section and result section.

To resize a pane, click on the section name: Explorer, Query, Result and then use the SHIFT + cursor key combinations. Upon clicking the name, the – Aa + control also appears, by which you can change the font size.

FYI: Resize hotkeys are not tracked if you're inside of an input element. This helps to avoid conflict with normal user typing.

6.2 OBJECT NAMING CONVENTION

There are just a few rules:

- K-objects names are automatically capitalized
- They must begin with A..Z or an underscore (_) character
- Beyond the first character, any letter or number or underscore can be used

6.3 K-OBJECT TYPES

KeySQL Data Type	Explorer Abbreviation	Explanation	Sample Values
char	char	A single data type for representing either a single character or a string.	“Z” “Bolshoi” ”Model 3” ”123” (as distinct from the number 123)
integer	int	A single data type for efficient storage of an integer value. It is ideal as a key value and is used for sequences.	123 -20000 123456789
number	num	A single data type for representing integer, decimal and real values.	123.456 -20000 2E+25
date	date	A single data type that represents both date and date-time values. KeySQL implements the ISO-8601 Date/Time Format that’s based on the proposed ITEF RFCC 3339 standard. Refer to the KeySQL Definitive Guide for more detail.	2022-02-27 2022-02-27T08:53:00.100 100 milliseconds (1/10 second) into the 53rd minute

composite	comp	Container for a set of other k-objects	
multi-composite	mult	Container that allows for multiple occurrences of a single k-object, be it an elementary one: char, number, date, or a composite, or another multi-composite.	