

# Qvu Data Service

## Help

Last Updated: 09/13/2023

# Table of Contents

Overview.....	4
Administration.....	4
System Settings.....	5
Repository Backup.....	6
Add/Edit a Datasource.....	6
Datasource Entry Fields.....	8
Datasource Table Settings.....	9
Table Column Settings.....	10
Table Foreign Key Settings.....	12
Custom Foreign Keys.....	13
Testing the Database Connection.....	16
Add/Edit a Role.....	17
Add/Edit a User.....	18
Add/Edit a Document Group.....	20
Query Design.....	21
Column Select Pane.....	22
Table Tree Icon Descriptions.....	24
Root Table.....	24
Root View.....	24
Imported Foreign Key Table (parent).....	24
Exported Foreign Key Table (child).....	24
Root Table with Column Selections.....	25
Root View with Column Selections.....	25
Imported Table with Column Selections.....	25
Exported Table with Column Selections.....	25
Imported Table with Inner Join (joins default to outer).....	25
Imported Table with Inner Join and Column Selections.....	25
Exported Table with Inner Join (join defaults to outer).....	25
Exported Table with Inner Join and Column Selections.....	25
Column.....	25
Primary Key Column.....	26
Setting Join Type.....	26
Related Table Display.....	26
Query Design Data Tab.....	27
Data Column Configuration Description.....	28
Data Column Configuration Panel Icons.....	28
Column Details Icon.....	29
Duplicate Column Icon.....	29
Copy Table Alias and Name to Clipboard.....	29
Move Column Position.....	30
Data Column Entry Fields.....	30
Query Design Filter Tab.....	30
Query Design SQL Tab.....	33
SQL Tab SQL Pane.....	33
Filter Values Prompts.....	34
Saving a Query Document.....	35
Loading a Saved Query Document.....	36
REST API.....	38
Custom Security.....	39

Qvu Repository.....	40
Qvu Repository config Folder.....	41
Qvu Repository documents Folder.....	41
Qvu Repository help Folder.....	41
Qvu Repository logs Folder.....	42
Language Support.....	42

# Overview

Qvu Data Service is an ad-hoc query and api data service design tool that allows users to create and save query designs in a user-friendly, web-based UI. Qvu Data Service provides REST API endpoints for users and applications to execute saved query documents and return results in tabular or json formatted result sets. Qvu Data Service provides role-based datasource, table column and document group access control and supports both Basic and OIDC authentication.

## Administration

The Qvu Admin tab allows users with the administration role to add and configure datasources, users, roles and document groups as well as the desired authentication scheme. After initializing the Qvu repository (see qvu-gettingstarted.pdf), start Qvu and login as the admin user – you should see the screen below:



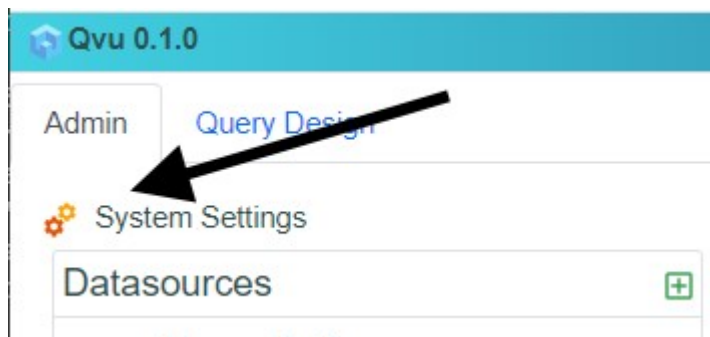
The initial admin tab will have no datasources, 2 default roles – **administrator** and **query designer** and one user – **admin**.

## System Settings

System Settings is used to configure Qvu Data Service authentication. There are 3 choices:

1. Basic Authentication
2. OIDC
3. Custom Authentication

Click the System Settings icon in the Admin tab to display the System Settings dialog:



System Settings

×

Default Security Type

basic

▼

Basic

Oidc

❓ Custom Security Service

Cancel

Save

Select the desired Default Security Type and complete the associated entry fields. If **basic** is selected then no other action is required. Security objects will be stored locally in the configured Qvu repository. If you wish to use a custom security implementation then a Java class must be entered in the Custom Security Service field. See the [Custom Security](#) section in this document for more information on this topic.

If **oidc** is chosen as the Default Security Type complete the required entries on the Oidc tab

System Settings

×

Default Security Type

oidc

Basic

Oidc

❓ \*Issuer Location URL

❓ \*Client ID

❓ \*Client Secret

❓ Admin Role Mapping

☐ Use Email for User Id

\*indicates required field

Cancel

Save

Issuer Location URL, Client ID and Client Secret are standard OIDC entries associated with your identity provider.

The Admin Role Mapping is used to map incoming OIDC role claims to the Qvu **administrator** role. Add a comma-delimited list of roles in this field if desired.

## Repository Backup

Click the Repository Backup icon to backup the repository:



A backup file target folder is specified in the <repository-folder>/config/application.properties file. This defaults to <repository-folder>/backups but can be changed (must restart the server).

```
backup.folder=c:/dev/qvu/backups
```

The backup file format is qvu-backup-yyyyMMddhhmmss.zip.

# Add/Edit a Datasource

To add a datasource, click the add icon on the Datasources control:

The Create new datasource dialog should display:

Create new datasource

✕

\*Database Type:

MySQL

▼

\*Name:

MySQL

Microsoft SQL Server

Oracle

PostgreSQL

Description:

\*JDBC Url:

\*JDBC Driver:

\*Schema

\*User Name:

\*Password:

② \*Max Imported Key Depth:

2

② \*Max Exported Key Depth:

4

Connection Timeout:

Idle Timeout:

Max Life Time:

Max Pool Size:

② Role Access:

Select roles...▼

\*indicates required field

Table Settings

Custom Foreign Keys

Test Connection

Cancel

Create

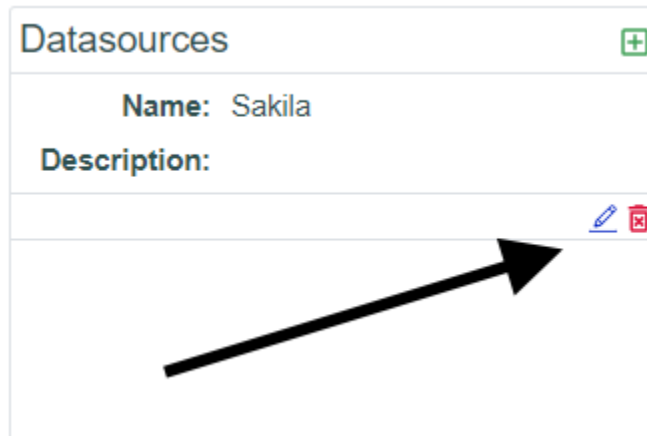
Select the desired database type and fill in the required entries. The table below describes each entry field:

## Datasource Entry Fields

Database Type	MySQL, Microsoft SQL Server, Oracle or PostgreSQL
Name	This is the datasource name which will show up in the Datasource selection drop down and be associated with the query document. The name must be unique
Description	A description of this datasource
JDBC Url	The database-specific JDBC URL to connect to the database. When specifying a Url for MySQL be sure to include <code>nullDatabaseMeansCurrent=true</code> , for Microsoft SQL Server you may need to include <code>encrypt=true;trustServerCertificate=true</code>
JDBC Driver	The JDBC Driver java class name
Schema	The database schema for this datasource
User Name	Database user name to use to connect to the database
Password	User password ti use to connect to the database
Max Imported Key Depth	The max depth to recurse through the imported (parent table) foreign key definitions when building table relationships.
Max Exported Key Depth	The depth max depth to recurse through the exported (child table) foreign key definitions when build table relationships
Connection Timeout	the maximum number of milliseconds to wait for a database connection checkout. Defaults to 30000 (30 seconds)
Idle Timeout	Maximum amount of time (in milliseconds) that a connection is allowed to sit idle in the pool. Defaults to 600000 (10 minutes)
Max Life Time	Maximum lifetime (in milliseconds) of a connection in the pool. An in-use connection will never be retired, only when it is closed will it then be removed. Defaults to 1800000 (30 minutes)
Max Pool Size	Maximum size that the connection pool is allowed to reach, including both idle and in-use connections. Basically this value will determine the maximum number of actual connections to the database backend. Defaults to 10.
Role Access	Roles required to access this datasource – no selections indicates datasource is available to all users

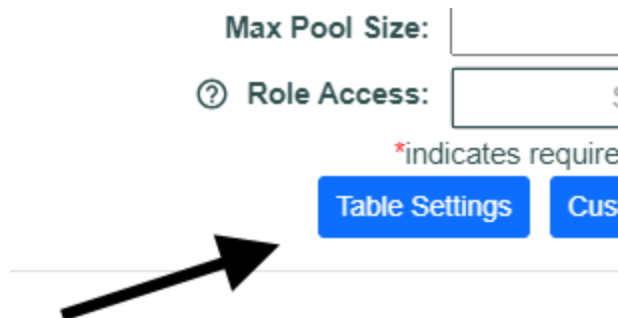
To edit/delete a datasource click the Edit/Delete icon





## Datasource Table Settings

A user can customize how datasource tables and columns are displayed in the UI and control user access to tables and columns by role assignment. To modify these setting click the Table Settings button to display the Table Settings dialog::



The Table Settings dialog will display a list of tables associated with the datasource. For each table the following fields can be set to handle the way the table is presented to the user in the UI:

?
Table Settings - Sakila
X

Table: actor

Display Name:

Roles:

Select roles... ▼

☐ Hide

Column Settings

Table: actor\_info

Display Name:

Roles:

Select roles... ▼

☐ Hide

Column Settings

Table: address

Display Name:

Cancel

Save

Display Name	A user friendly name can be entered that will be used in the UI whenever the table is presented to the end user
Roles	Roles can be associated with the table and only users that are members of the selected roles will be able to see the table in the UI. No selection means the table is available to all users.
Hide	When checked the table will not be displayed in the UI for any user.

## Table Column Settings

Click the Column Settings button on any table in the Table Settings dialog to configure how the associated columns are displayed in the UI for the table. When clicked the Column Settings dialog will display:

? Column Settings - actor
 ×

Column: actor\_id

Display Name:

Roles:
Select roles...

☐ Hide

Column: first\_name

Display Name:

Roles:
Select roles...

☐ Hide

Column: last\_name

Display Name:

Roles:
Select roles...

☐ Hide

Column: last\_update

Display Name:

Roles:
Select roles...

☐ Hide

Column:

Display Name:

Roles:
Select roles...

☐ Hide

Cancel

Save

The Column Settings dialog will display a list of columns associated with the table. For each column the following fields can be set to handle the way the column is presented to the user in the UI:

Display Name	A user friendly name can be entered that will be used in the UI whenever the column is presented to the end user
Roles	Roles can be associated with the column and only users that are members of the selected roles will be able to see the column in the UI. No selection means the column is available to all users.
Hide	When checked the column will not be displayed in the UI for any user.

## Table Foreign Key Settings

The table foreign key settings allow an administrator to customize how foreign key tables are displayed in the data select tree. The foreign key settings also provides a means to set the field name for the child data field of foreign key data when retrieving data as an object graph. To add foreign key settings click the Foreign Key Settings button for the desired table in the Table Settings dialog to display the Foreign key settings dialog:

**Table:** payment

**Display Name:**

**Roles:**

☐ Hide

**Column Settings** **Foreign Key Settings**

**Table:** rental

### Foreign Key Settings - payment

**Name:** fk\_payment\_customer  
**Type:** imported  
**To Table:** customer  
**Column Mappings:** customer\_id->customer\_id  
**Display Name:**   
**Field Name:**

**Name:** fk\_payment\_rental  
**Type:** imported  
**To Table:** rental  
**Column Mappings:** rental\_id->rental\_id  
**Display Name:**   
**Field Name:**

**Name:** fk\_payment\_staff  
**Type:** imported

**Cancel** **Save**

Enter the desired Display Name and/or Field Name for the selected table foreign key relationship. By default the foreign key is displayed as:

`<foreign-table-name>: fromColumn1→toColumn1, fromColumn2→toColumn2...`

You can enter a \$t in the Display Name to include the foreign table in the Display Name. You can add \$c to include the foreign key column mappings in the Display Name.

## Custom Foreign Keys

Configured database foreign keys are used by Qvu to define the table relationships for query design. In cases where a desired foreign key does not exist in the database Qvu supports defining pseudo foreign key definitions. To create a pseudo foreign key click the Custom Foreign Key button to display the Custom Foreign Key dialog

The screenshot shows the 'Custom Foreign Keys' dialog box for the 'Sakila' database. The dialog has a title bar with a question mark icon and a close button. Inside, there are five required fields: 'Name', 'Source Table', 'Target Table', 'Source Columns', and 'Target Columns'. Each field is preceded by an asterisk. 'Source Table' and 'Target Table' are dropdown menus. 'Source Columns' and 'Target Columns' have a grid icon to the left of the input field. Below these fields is a checkbox labeled 'Imported Keys'. At the bottom right, there are three buttons: 'Add' (highlighted with a blue border), 'Cancel', and 'Save'. A legend at the top left of the dialog area states '\*indicates required field'.

Enter a unique name for the foreign key, select the source and target tables from the associated drop downs:

?

Custom Foreign Keys - Sakila

X

\*Name:

newfk

\*Source Table:

film\_category

▼

\*Target Table:

film

▼

\*Source Columns:

actor

address

category

city

country

customer

film

film\_actor

film\_category

film\_text

inventory

language

payment

rental

staff

store

actor\_info

customer\_list

film\_list

\*Target Columns:

\*indicates required field

Add

Cancel

Save

Now choose source and target foreign key columns. When choosing the foreign key columns you will be presented with a multi-select control. Selection order is import. The columns will added as a comma-delimited list in the order selected, make sure you choose the associated columns in the correct order to map the table relationship correctly.

?

Column select for customer

×

<input checked="" type="checkbox"/> customer_id	
<input type="checkbox"/> store_id	
<input type="checkbox"/> first_name	
<input type="checkbox"/> last_name	
<input type="checkbox"/> email	
<input type="checkbox"/> address_id	
<input type="checkbox"/> active	
<input type="checkbox"/> create_date	
<input type="checkbox"/> last_update	

Cancel

Save

For exported foreign tables a user can enter a raw comparison value in the text entry field if desired as shown below:

? Column select for payment
×

<input type="checkbox"/> payment_id	
<input checked="" type="checkbox"/> customer_id	
<input checked="" type="checkbox"/> staff_id	= 2
<input type="checkbox"/> rental_id	
<input type="checkbox"/> amount	
<input type="checkbox"/> payment_date	
<input type="checkbox"/> last_update	

Cancel
Save

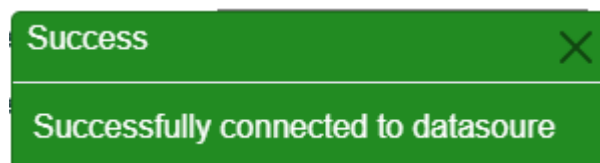
This will produce a join clause similar to this:

```
FROM `customer` `t0`
    left outer join `payment` `t1` ON
        (`t1`.`customer_id` = `t0`.`customer_id` and
`t1`.`staff_id` = 2)
WHERE
```

Check the Imported Key checkbox if this relationship is to a parent table – leave blank if it is to a child table.

## Testing the Database Connection

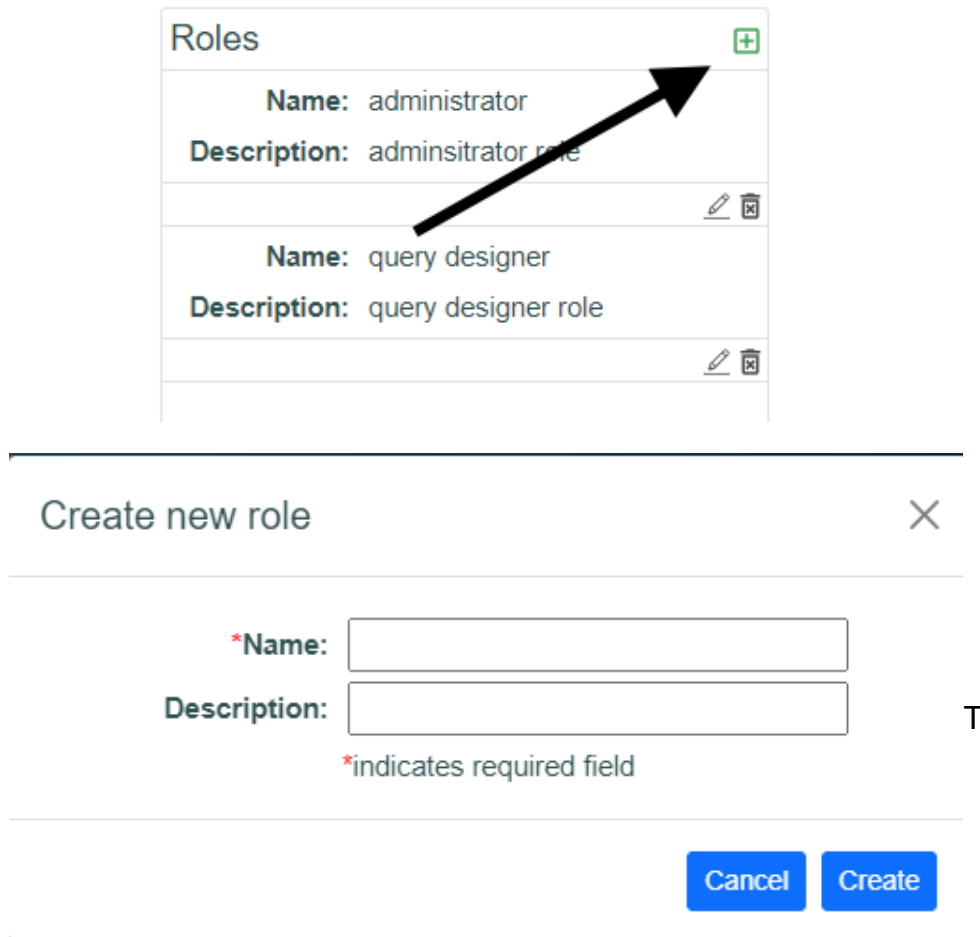
Once the datasource entries are complete you can test the database connection by clicking the Test Connection button. If the connection succeeds you will receive a success message:









## Add/Edit a Role

To add a new role click the Add icon to display the Create new Role dialog::



The image shows a 'Roles' table with two entries: 'administrator' and 'query designer'. A black arrow points from the 'Add' icon (a green square with a white plus sign) in the top right corner of the table to the 'Create new role' dialog box. The dialog box has a title bar 'Create new role' with a close button (X). It contains two input fields: '\*Name:' and 'Description:'. The '\*Name:' field is required, as indicated by the asterisk and the note '\*indicates required field' below the fields. There are 'Cancel' and 'Create' buttons at the bottom right of the dialog.

Roles		+
<b>Name:</b> administrator	<b>Description:</b> adminisitrator role	
		 
<b>Name:</b> query designer	<b>Description:</b> query designer role	
		 

Create new role

X

\*Name:

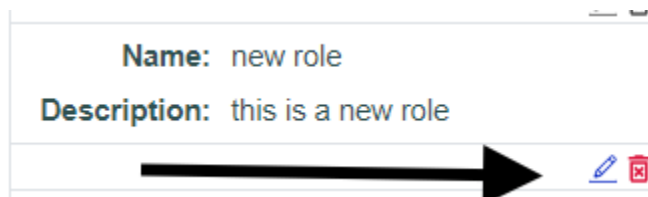
Description:

\*indicates required field



Cancel

Create






To edit/delete a role click the Edit/Delete icon:



The image shows a single row in the 'Roles' table with the name 'new role' and description 'this is a new role'. A black arrow points from the bottom of the row to the edit and delete icons (a blue pencil and a red trash can) located at the bottom right of the row.

<b>Name:</b> new role	<b>Description:</b> this is a new role	
		 

There are 2 canned system roles that cannot be edited or deleted – **administrator** and **query designer**. On these roles the edit/delete icons are disabled.

Roles		
<b>Name:</b> administrator	<b>Description:</b> adminsitator role	 
<b>Name:</b> query designer	<b>Description:</b> query designer role	 

## Add/Edit a User

Qvu Users are handled a bit differently than other security-related entities. If Qvu is configured to use the local Qvu repository for authentication then a user can be added, edited and deleted.

If authentication is configured to use OIDC, when an authenticated user does not exist in the local Qvu repository that user will be added for user information purposes – authentication will be handled by the OIDC identity provider. A role mapping can be setup to map incoming role claims to the administration role if desired – see the System Settings section for more information on this topic.

If the custom authentication plugin is enabled then the custom authentication service is responsible for providing user information and expected roles (administrator and query designer). The user will be read only in the Qvu application.

There is a canned system user – Admin – which cannot be edited or deleted.

To add a new user when Qvu is configured to use local repository authentication click the Add icon to display the Create new User dialog:

Users

User ID: admin

First Name:

Last Name:

Create new user

\*User ID:

\*Password:

First Name:

Last Name:

Email:

Roles:

\*indicates required field

Cancel

Create

Complete the required and entries and save the user. To assign roles to the user, click the Roles drop down and select the desired roles:

User ID: testuser

First Name: John

Last Name: Doe

**Create new user** [X]

\*User ID: newuser

\*Password: .....

First Name: John

Last Name: Doe

Email: jdoe@email.com

Roles: Role(s) selected [X] ^

Search

- ☒ administrator
- ☒ query designer

Create

User passwords are stored locally as a md5 hash of the entered passwords. See the [Qvu Repository](#) section for more information.

To edit/delete a user when enabled click the Edit/Delete icon:

## Add/Edit a Document Group

When query documents are saved they must be assigned a document group. A canned, default document group **user** is always available and cannot be edited or deleted. Documents in the user document group can only be seen by the current user. Additional document groups can be created as desired.

To add and new Document Group click the Add icon to display the Create new Group dialog:

**Document Groups** [+]

Name: user  
Description: current user document group

Name: general  
Description: default document group

[Edit] [Delete]

Create new group



**\*Name:**

--

**Description:**

--

### Roles:

Select roles... 

\*indicates required field

Cancel

Create

Enter the required fields – the name must be unique. Roles can be assigned to a document group. Only users that are members of the selected roles will be able to see the documents in the group. If no role is selected, all users can access the documents in the group.

To edit/delete a document group click the Edit/Delete icon:

**Name:** new group  
**Description:** this is a new group

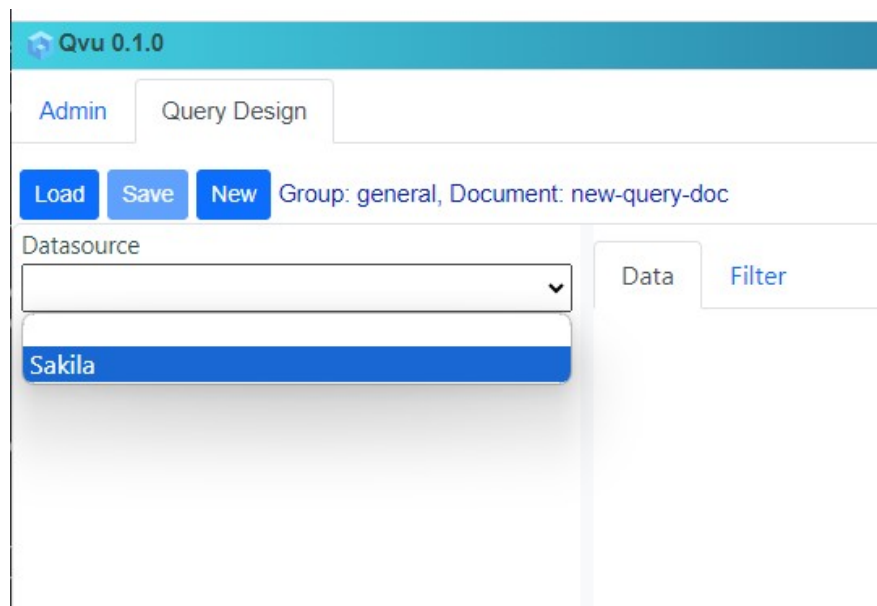
If a document group is deleted the document group folder is backed up to the configured backup folder then the group folder and any documents are deleted. A query document can be assigned a new document group. See [Saving a Query Document](#) for more information on this topic.

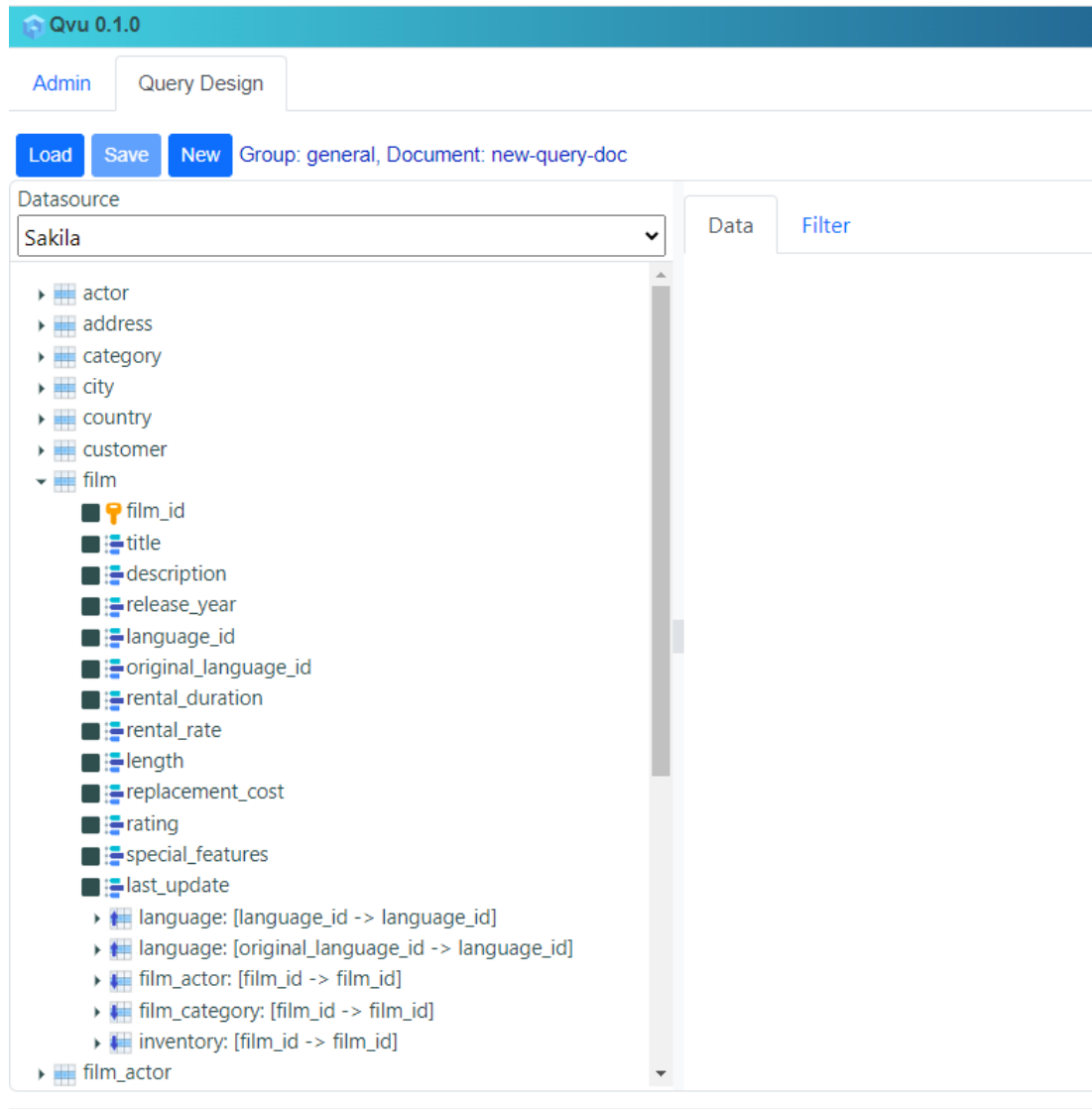
# Query Design

The Qvu UI is built on React and provides a rich, user-friendly interface for query design. Query design is based on a root table selection from a selected datasource. Defined table relationships associated with the selected root table are displayed in a tree view that allows users to select data columns as desired.

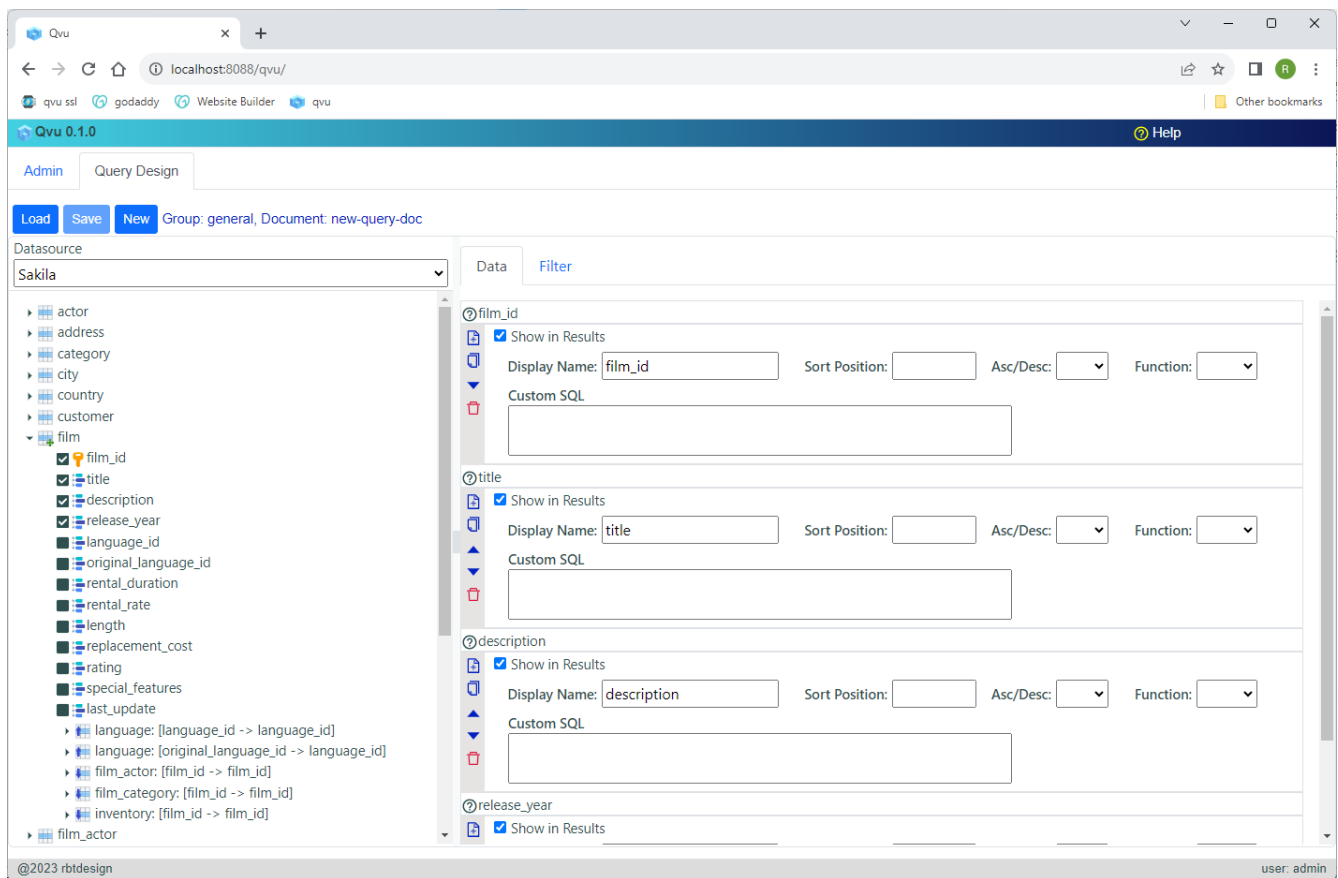
## Column Select Pane

From the Qvu Query Design tab select the desired datasource to display the table hierarchy tree:





Tables and columns displayed in the tree are based on the role settings discussed in the [Administration](#) section above. The user can now select the desired columns. Column selection order determines the initial order of the result set columns. This can be changed in the Data tab on the right split pane. As columns are selected the Data tab will populate with column configuration panes.



## Table Tree Icon Descriptions

Below is described the various icons you see in the table tree:

### ***Root Table***



### Root View



***Imported Foreign Key Table (parent)***



### Exported Foreign Key Table (child)





***Root Table with Column Selections***



***Root View with Column Selections***



***Imported Table with Column Selections***



***Exported Table with Column Selections***



***Imported Table with Inner Join (joins default to outer)***



***Imported Table with Inner Join and Column Selections***



***Exported Table with Inner Join (join defaults to outer)***



***Exported Table with Inner Join and Column Selections***



***Column***

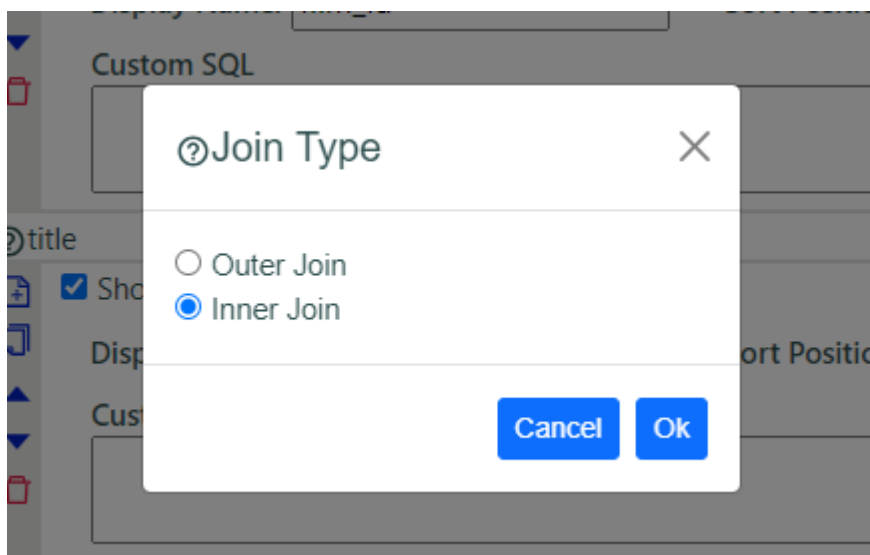
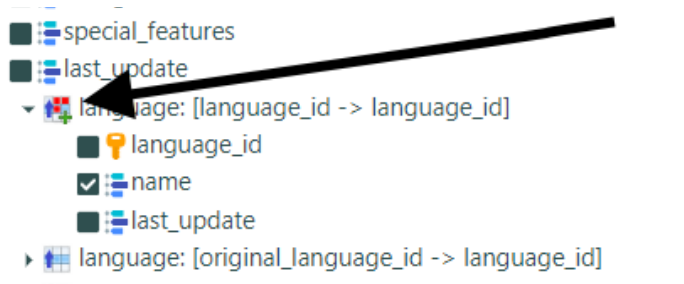


## Primary Key Column



## Setting Join Type

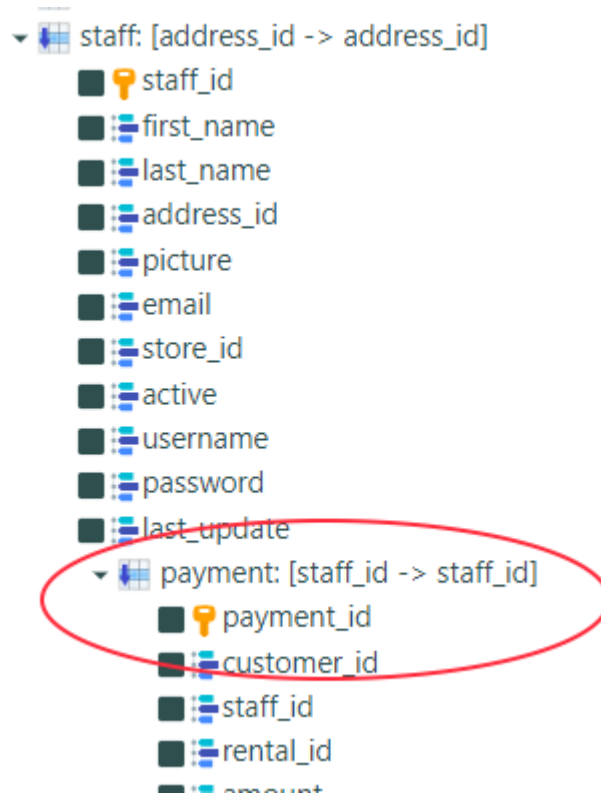
Right click on a related table to display the Join Type dialog and select the desired join type:



## Related Table Display

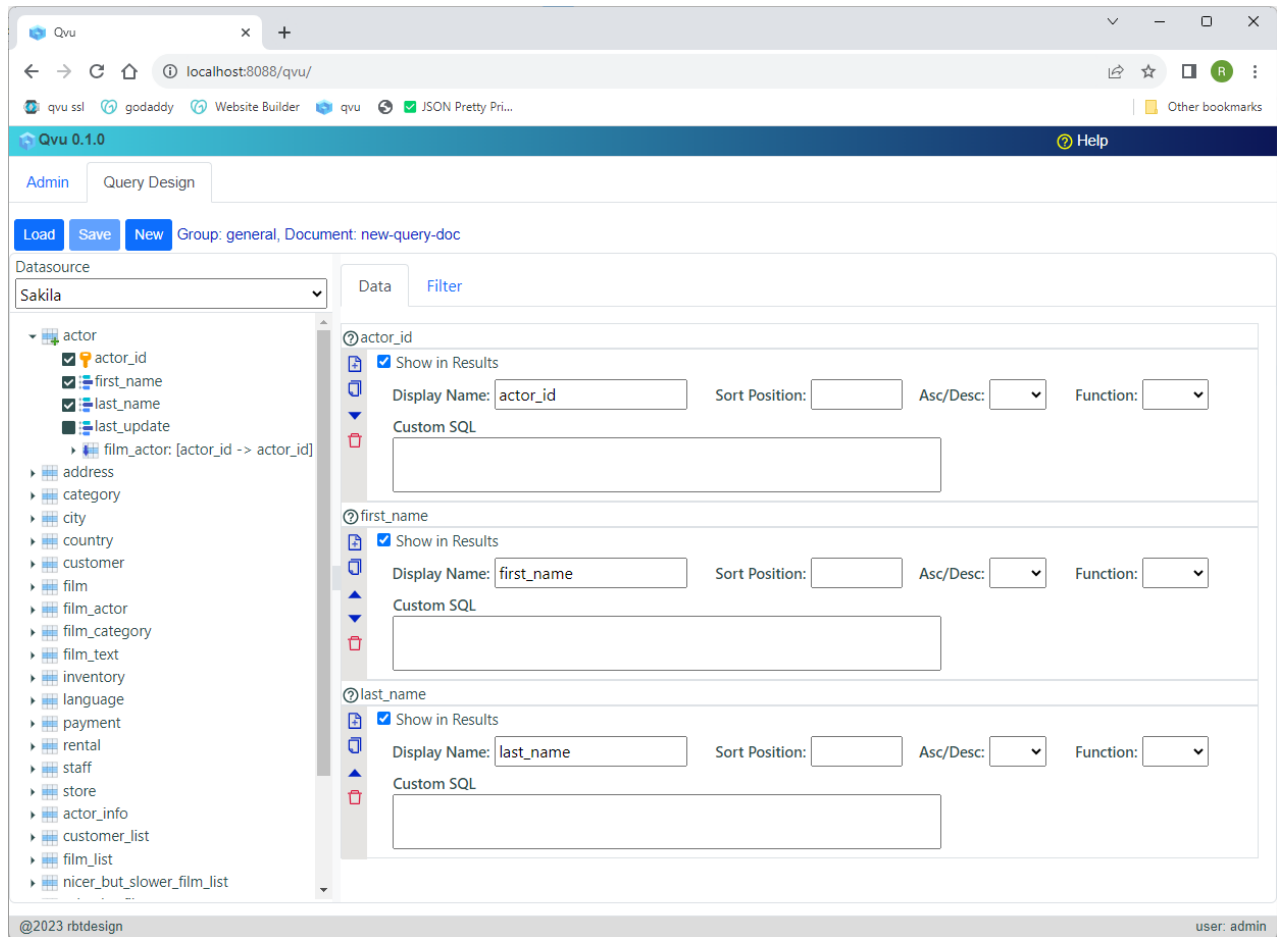
Related tables are displayed in the following format in the table tree:

`to_table_name[to_column1→from_column1, to_column2→from_column2...]`



## Query Design Data Tab

For each column selected in the data select tree pane an associated column configuration panel will display in the Data tab.

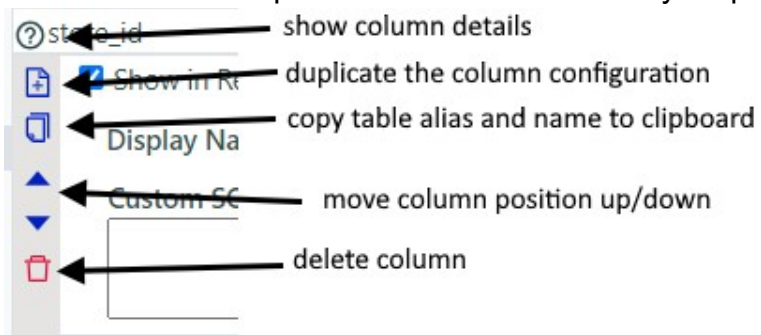


## Data Column Configuration Description

The user can apply various options to each selected data column. In addition, the column configuration panel support deleting, moving and duplication the associated column.

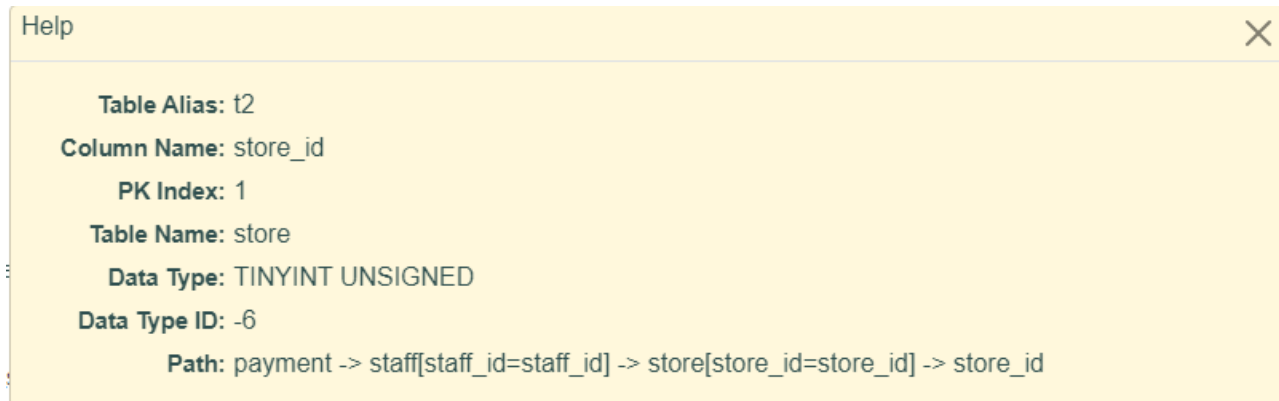
### Data Column Configuration Panel Icons

Below you will find a description of the icon functionality on panel:



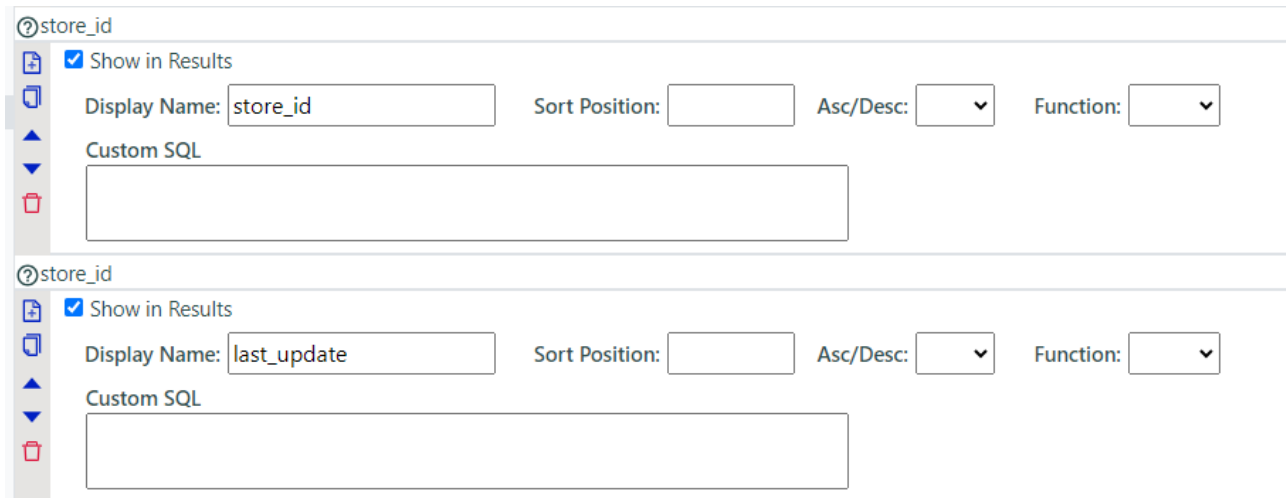
## Column Details Icon

Clicking the column details icon display more information about the column:



## Duplicate Column Icon

Clicking the duplicate column icon will create a duplicate column definition which the user can then configure in a different manner than the original. This allows for the same column to be selected in different ways in final select statement.



## Copy Table Alias and Name to Clipboard

This is a useful tool when making a custom SQL entry. A custom SQL entry must contain database-specific valid SQL with the associated raw base table aliases and column names. Name. Clicking this icon will copy this name to the clipboard – for example: *t2.store\_id*

### ***Move Column Position***

The generated SQL select column order will be in the order of the columns listed in the **Data** tab. To change this order the user can click the associated arrow icons to move a column up and down. The column can also be dragged by clicking on the name bar and dragging the column configuration to the desired position.

### ***Data Column Entry Fields***

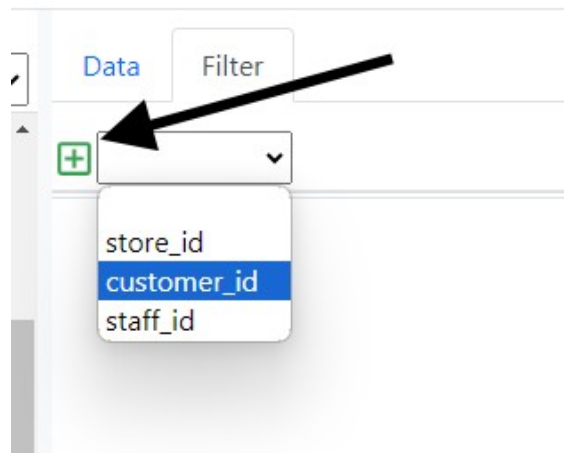
The entry fields are described in the table below:

Display in Results	This is checked by default. If unchecked, column is still available for filtering but will not appear in the result set data
Display Name	This will be the column name return from the SQL query – will become the “as” clause in the select.
Sort Position	Enter an integer sort position greater than 0 here if you want to sort by this column
Asc/Desc	Select sort direction – defaults to Asc
Function	Aggregate functions can be applied to the column based on type. Select the function as desired. Qvu will handle building the appropriate Group By and Having clauses
Custom SQL	The user can enter database-specific SQL here to perform any query operation desired. The entered SQL must be valid for the database type. This is where the <b>Copy Table Alias and Name</b> clipboard functionality discussed above comes into play. The user can sum multiple columns, apply database-specific functions etc. When this field is populated the actual column associated with the panel is ignored and the select SQL is replaced by the custom SQL entered.

### **Query Design Filter Tab**

The filter tab allows the user to build the SQL where clause. In order to execute a query it must have a filter configuration – no open-ended queries are allowed.

To add a filter column select the desired column from the drop down and click the Add icon:



Data Filter SQL

+ customer\_id

customer\_id

and/or	(	Column	Operator	Value	)
		customer_id	=	100	

Custom SQL

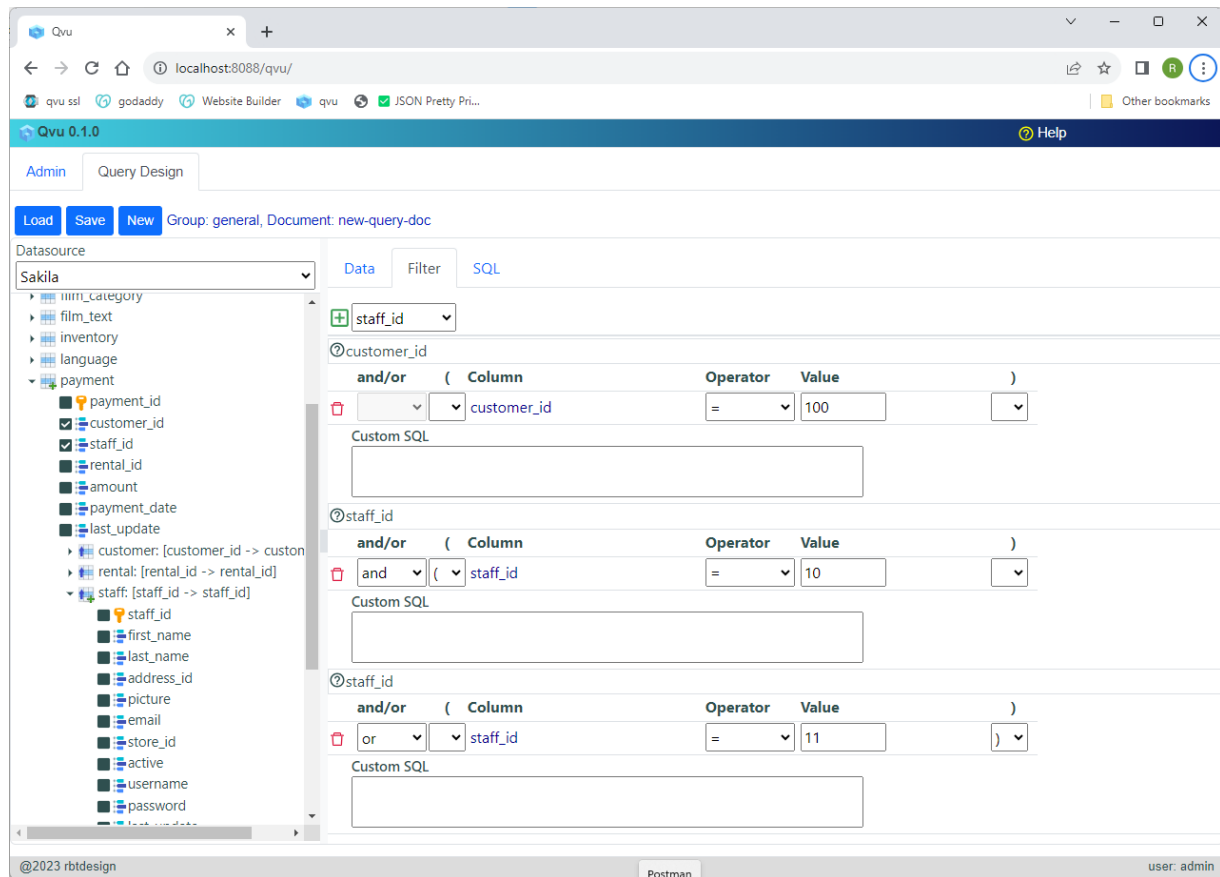
The user can select parenthesis and the and/or operators as required. If the user hovers over the filter select drop down additional column details will display:

+

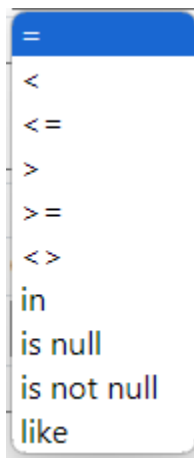
first\_name  
last\_name  
email  
rental\_id  
amount

Column	Operator	Value
amount	>	0

Table Alias: t1  
Path: customer->payment[customer\_id=customer\_id,staff\_id= 2]->rental\_id



The available comparison operators are shown below:



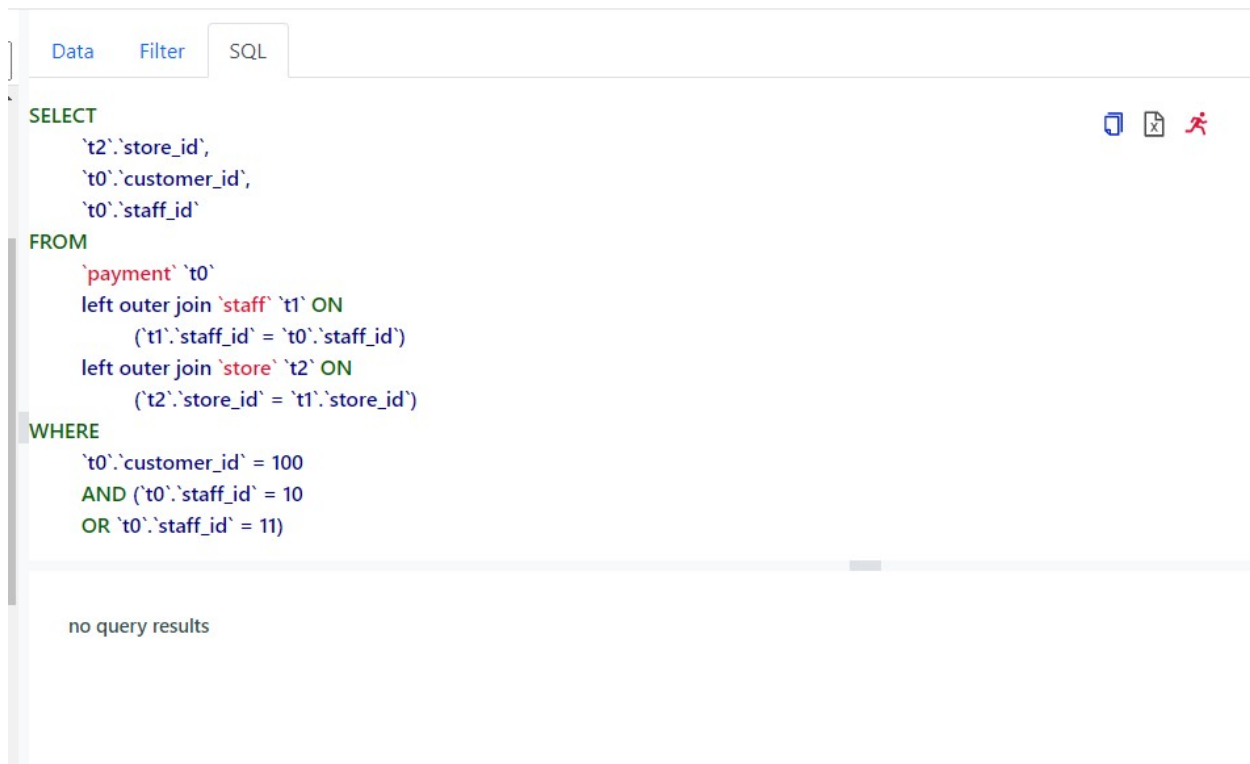
When entering comparison operator enter the value only – no quotes are required. When using the “in” operator, enter a comma-delimited list of values. When using the “like” operator enter the appropriate wildcard characters (% and \_).



Similar to the data select entry, custom SQL can be entered in the Custom SQL field – see the discussion of the [custom SQL entry](#) in the Data Column Entry Fields table for more information.

## Query Design SQL Tab

Once filter entries have been created the SQL Tab becomes visible



The SQL tab is a split pane with 2 sections – the upper section displays the generated SQL and action icons and the lower pane displays the query results after the query is run.

## SQL Tab SQL Pane

The **SQL** Pane displays the database-specific generated SQL statement as well as action icons. The icons functions are described below:



Clicking this icon will copy the SQL statement to the clipboard



Clicking this icon will export query results to excel – icon is disabled if no results available.



Clicking this icon will run the query.

When the run icon is clicked, the query will execute and results will populate the results table in the bottom SQL Tab pane:

Data
Filter
SQL

```

SELECT
    `t0`.`title`,
    `t0`.`description`
FROM
    `film` `t0`
WHERE
    `t0`.`title` is not null

```

	title	description
1	ACADEMY DINOSAUR	A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in The Canadian Rockies
2	ACE GOLDFINGER	A Astounding Epistle of a Database Administrator And a Explorer who must Find a Car in Ancient China
3	ADAPTATION HOLES	A Astounding Reflection of a Lumberjack And a Car who must Sink a Lumberjack in A Baloon Factory
4	AFFAIR PREJUDICE	A Fanciful Documentary of a Frisbee And a Lumberjack who must Chase a Monkey in A Shark Tank
5	AFRICAN EGG	A Fast-Paced Documentary of a Pastry Chef And a Dentist who must Pursue a Forensic Psychologist in The G
6	AGENT TRUMAN	A Intrepid Panorama of a Robot And a Boy who must Escape a Sumo Wrestler in Ancient China
7	AIRPLANE SIERRA	A Touching Saga of a Hunter And a Butler who must Discover a Butler in A Jet Boat
8	AIRPORT POLLOCK	A Epic Tale of a Moose And a Girl who must Confront a Monkey in Ancient India
9	ALABAMA DEVIL	A Thoughtful Panorama of a Database Administrator And a Mad Scientist who must Outgun a Mad Scientist
10	ALADDIN CALENDAR	A Action-Packed Tale of a Man And a Lumberjack who must Reach a Feminist in Ancient China
11	ALAMO VIDEOTAPE	A Boring Epistle of a Butler And a Cat who must Fight a Pastry Chef in A MySQL Convention

Page Size: 200
Page: 1 of 5
Total Records: 1000

## Filter Values Prompts

When a defined filter value is left blank in the Filter Tab the user will be prompted for entry when running the query – for example if the user creates this filter:

Data
Filter
SQL

+ customer\_id

? rental\_date

and/or	(	Column	Operator	Value	)
<div> <div></div> <div></div> </div>	<div></div>	rental_date	>	mm/dd/yyyy	<div></div>

Custom SQL

? customer\_id

and/or	(	Column	Operator	Value	)
<div> <div></div> <div></div> </div>	<div></div>	customer_id	>		<div></div>

Custom SQL

Then when the **run** icon is clicked the Run query value entry dialog will display to prompt for user input:

Run query

rental\_date >
mm/dd/yyyy

customer\_id >

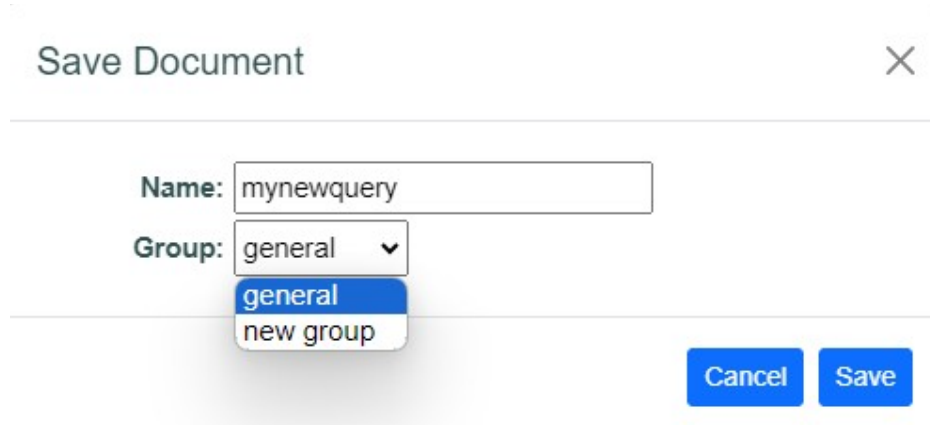
Cancel
Run

## Saving a Query Document

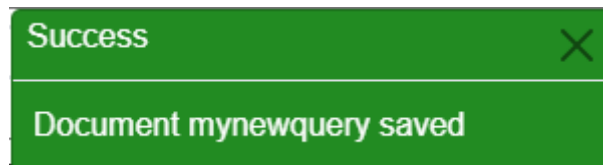
Once a user has created a query the query can be saved as a re-usable Query Document. To save a query click the Save button



to display the Save Document dialog:



Enter a unique name for the new Query Document and select an appropriate Document Group then click Save. If document saves successfully you should see a success message:



## Loading a Saved Query Document

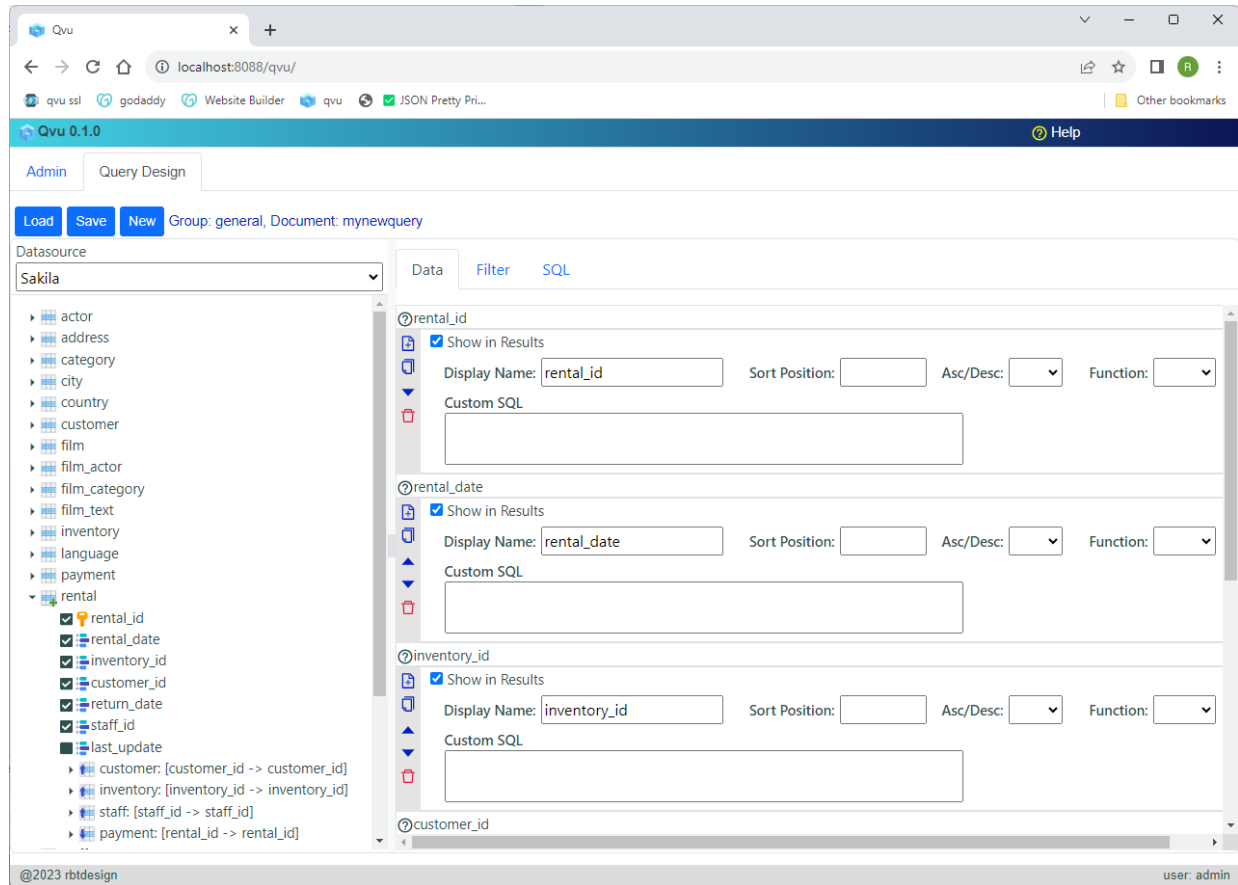
To load a saved query document click the Load button to display the Select query document dialog, open the appropriate document group and select the desired document.

## Select query document



- ▼  general
  -  mynewquery.json
- ▶  sales
- ▶  shipping

The Query Design tab should populate with the selected query document setting.



## REST API

The Qvu Data Service provides REST API endpoint to execute created query documents. The requesting application must successfully authenticate when using the API. For Basic Auth this consists of sending an appropriate username and password in the header. For OIDC this is dependent open the identity provider.

The API calls to retrieve data are shown below:

- `<qvu-hostname>/qvuvapi/v1/query/run/json`
- `<qvu-hostname>/qvuvapi/v1/query/run/json/objectgraph`
- `<qvu-hostname>/qvuvapi/v1/query/run/tabular`

The `/run/json` endpoint return results as an array of flat JSON records. The `/run/json/objectgraph` endpoint returns a json object graph with parent child relationships. and the `/run/tabular` returns a JSON object with tabular results in the following format:

```
{
  "rowCount": 100,
  "header": ["header1", "header2"...],
  "columnTypes": [jdbcType1, jdbcTy2...],
  "initialColumnWidths": [column1Width, column2Width...],
  "data": [[row1col1, row1col2...], [row2col1, row2col2...]]
}
```

The first data column in the tabular results will hold the row number.

All endpoints expect a POST request with a JSON body in the following format:

```
{
  "documentName": "query-document-name",
  "groupName": "query-document-group",
  "user": "current-user-name",
  "parameters": [
    {"dataTypeName": "integer", "value": someIntValue},
    {"dataTypeName": "string", "value": "someStringValue"},
    {"dataTypeName": "date", "value": "1990-01-01"}]
}
```

Where parameters are any required runtime parameters. The runtime parameter types supported are listed below:

- string
- float
- integer
- date
- timestamp
- time
- boolean

## Custom Security

To implement a custom security scheme in Qvu Data Service follow the following steps:

1. Clone the qvu-client-utils git project found at:  
<https://rbtucker@github.com/rbtucker/qvu-client-utils.git>
2. Build the project with “mvn clean package”

3. Install the qvu-client-utils-1.0.0.jar in your local git repository using the following command:  
`mvn install:install-file -Dfile=qvu-client-utils-1.0.0.jar -DgroupId=org.rbtddesign -DartifactId=qvu-client-utils -Dversion=1.0.0 -Dpackaging=jar`
4. Build a custom service class that implements the `org.rbtddesign.qvu.client.utils.SecurityService` interface from the library built in step 3. Make sure that the maven dependency for the qvu-client-utils is set to scope "provided"
5. An example project can be found at <https://rbttucker@github.com/rbttucker/qvu-custom-security-example.git>
6. Copy the SecurityService implementation jar to the <qvu-repository-folder>/extlibs
7. Now start the qvu data service with the system parameter:  
`-Dloader.path=<qvu-repository-folder>/extlibs`

You can now enter your custom java class in the System Settings Custom Security Service field as shown below:

The screenshot shows a 'System Settings' window. At the top, there's a title bar with 'System Settings' and a close button. Below the title bar, there's a section for 'Default Security Type' with a dropdown menu currently showing 'basic'. Underneath this, there are two tabs: 'Basic' (which is selected) and 'Oidc'. In the 'Basic' tab, there's a field labeled 'Custom Security Service' with a question mark icon to its left. This field contains the text 'org.mypath.CustomSecurityService' and has a red squiggly underline. At the bottom right of the window, there are two buttons: 'Cancel' and 'Save'.

Restart the Qvu Data Service and the custom class should be used for user authentication.

## Qvu Repository

The Qvu repository is where all the Qvu artifact and configuration is stored – the layout is shown below:

```
<repository-root>
|
|_____ config
|_____ documents
```



```
|_____ extlibs
|_____ help
|_____ logs
```

Below is a description of each folder:

## Qvu Repository config Folder

The config folder holds the following configuration files used by the Qvu Data Service:

- application.properties – contains basic server configuration information such as root context, SSL properties and server port
- qvu-datasources.json – contains the configured Qvu datasources
- qvu-document-groups.json – contains the configured document groups
- qvu-language.json - contains the Qvu language text setup
- qvu-security.json – contains the authentication setup and the configured users and roles

## Qvu Repository documents Folder

The documents folder contains query documents stored by group name in the following layout:

```
documents
|_____ group1
|           |_____ query
|                   doc1.json
|                   doc2.json
|_____ group2
|           |_____ query
|                   doc1.json
|                   doc2.json
```

## Qvu Repository help Folder

Contains Qvu help and getting started documents. Document names are in the format qvu-gettingstarted-<language-property>.pdf and qvu-help-<language-property>.pdf. The default files are qvu-gettingstarted-EN-us.pdf and qvu-help-EN-us.pdf. Other language help files can be added as desired. If no file for a particular language exists then the EN-us version will be used.

## Qvu Repository logs Folder

Holds the Qvu server logs and archived logs

## Language Support

The Qvu Data Service is designed to support multiple languages. The base language file is found in the Qvu repository at config/qvu-language.json. A snippet of the file is shown below:

```
{
  "en-US": {
    "?": "?",
    "Add datasource": "Add datasource",
    "Add group": "Add group",
    "Add role": "Add role",
    "Add role": "Add role",
    "Add user": "Add user",
    "Add user": "Add user",
    "Admin": "Admin",
    .
    .
    .
  }
}
```

To add another language add the associated language property and replace the JSON value fields with the appropriate text. You can also modify the existing text to customize the UI display as desired.

The default help files are located in the Qvu repository under the help folder:

- qvu-gettingstarted-en-US.pdf
- qvu-help-en-US.pdf

Custom help files can be created with the appropriate language property included in the name and those will display based on the language property provided by the browser.