

# Lab 1 – Data Analytics

## Introduction

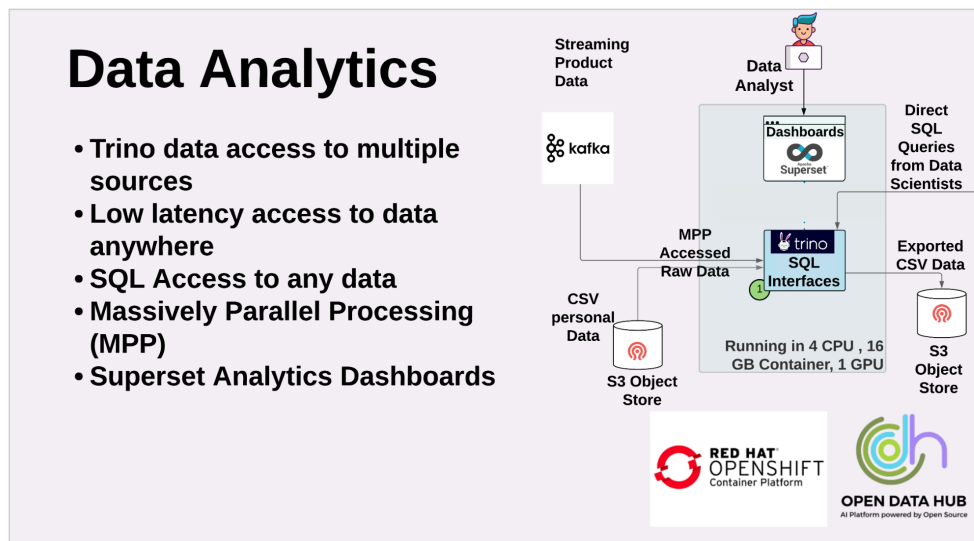
One of the first things your data analysts and data engineers will need to do is analyse the raw data, with a view to preparing and transforming it to a state that will be consumable by AI/ML model algorithms.

In this lab, we're going to use a powerful toolset combining

- an in-memory data analytics engine called Trino. Trino provides high speed access to many different on-premises and cloud based data sources. These include relational and no-SQL databases, object stores over S3 interfaces, Streaming data from Kafka and many more. Trino abstracts the actual underlying data store implementation and provides a uniform ANSI SQL interface, with which to access its many supported data stores.
- a visualization tool called Superset, which will use Trino as the backing data source.

The combination of these two tools will provide powerful data analytics capabilities, critical at this stage in the workflow.

This diagram illustrates what we're implementing:



You can see Trino is an SQL exposing abstraction in front of actual data located in Kafka and S3 Object storage. Superset using this Trino interface to display charts and dashboards.

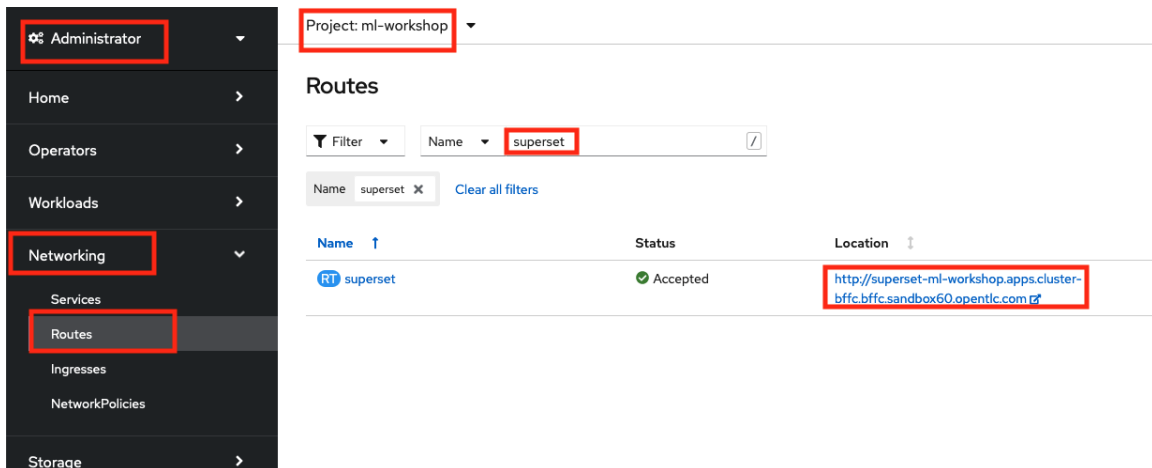
## Instructions for the Trino backed Superset workshop

To save time, the workshop administrators have already made simple connections using Trino to two datasets:

1. To a CSV file over an S3 interface. This CSV file is located in an underlying Object storage implementation called Minio. The file contains demographic type data on our customer data set, data such as gender, whether they have dependents and other demographic features.
2. To Streaming data located in an Apache Kafka store on the OpenShift cluster. This dataset contains product consumption for the same customers as are in the CSV file. Each record is labeled indicating whether that customer churned or not.

In Superset, using Trino, we've created two logical SQL tables corresponding underlying data sets as well as Query joining them on customer id.

Login to OpenShift using the credentials your administrator gave you. Choose **Administrator** from the dropdown (If it's your first time logging in, **Developer** will be selected there) Navigate to Network -> Routes. Ensure the desired project is selected (**ml-workshop** in my case). Filter on the word Superset and open that route, by clicking on the URL as shown.



The screenshot shows the OpenShift console interface. On the left sidebar, the 'Administrator' role is selected. The 'Project: ml-workshop' dropdown is set to 'ml-workshop'. The 'Routes' section is active, showing a filter for 'superset'. A table lists routes, with one route named 'superset' having a status of 'Accepted' and a location URL: 'http://superset-ml-workshop.apps.cluster-bffc.bffc.sandbox60.opentlc.com/g'.

Name	Status	Location
superset	Accepted	<a href="http://superset-ml-workshop.apps.cluster-bffc.bffc.sandbox60.opentlc.com/g">http://superset-ml-workshop.apps.cluster-bffc.bffc.sandbox60.opentlc.com/g</a>



Enter credentials **admin / admin**.

Superset Sign In form. The form is titled "Sign In" and asks for "Enter your login and password below:". It has fields for "USERNAME:" (containing "admin") and "PASSWORD:" (containing "\*\*\*\*\*"). A "SIGN IN" button is at the bottom.

As this is a shared service between all participants, and the setup has already been done by your instructor, we'll just show you the steps we took to connect Superset to Trino & from there to underlying data.

Choose menu item Data -> Databases. Create or Edit the **trino** Database

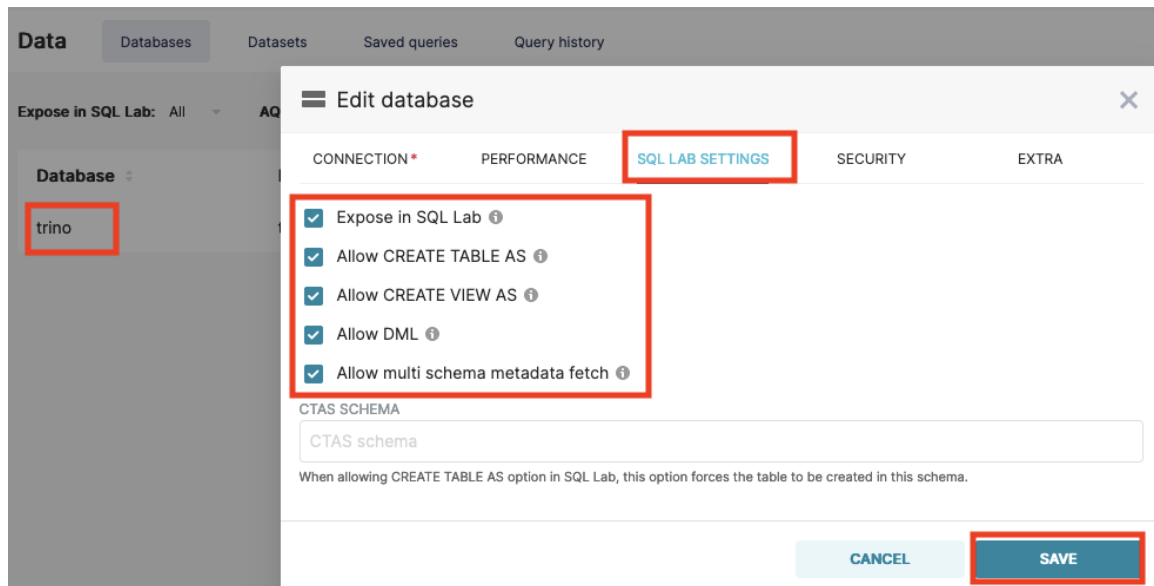
Superset Databases table. The table has columns: Database, Backend, AQE, DML, CSV upload, Expose in SQL Lab, Created by, Last modified, and Actions. The first row is for the "trino" database. The "trino" cell in the first column is highlighted with a red box. The "Actions" column for the "trino" row has a red box around the "Edit" icon.

Database	Backend	AQE	DML	CSV upload	Expose in SQL Lab	Created by	Last modified	Actions
trino	trino	x	✓	x	✓	admin admin	27 minutes ago	

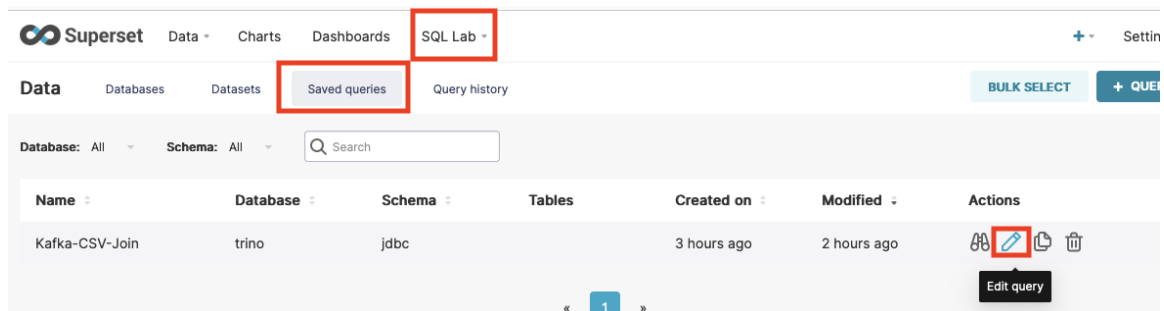
Notice it's simply a matter of adding the URI **trino://admin@trino-service:8080/** to connect to Trino as shown. Test the Connection.

Superset "Edit database" form. The form has tabs: CONNECTION, PERFORMANCE, SQL LAB SETTINGS, SECURITY, and EXTRA. The "CONNECTION" tab is active. It has fields for "DATABASE NAME\*" (containing "trino") and "SQL ALCHEMY URI\*" (containing "trino://admin@trino-service:8080/"). A "TEST CONNECTION" button is highlighted with a red box. There are also "CANCEL" and "SAVE" buttons at the bottom.

Move to the SQL LAB SETTINGS tab and notice we needed full access by selecting the checkboxes.



In Superset, choose SQL LAB -> Saved Queries. Edit the query **Kafka-CSV-Join** as shown (though your query may be named differently)



Notice:

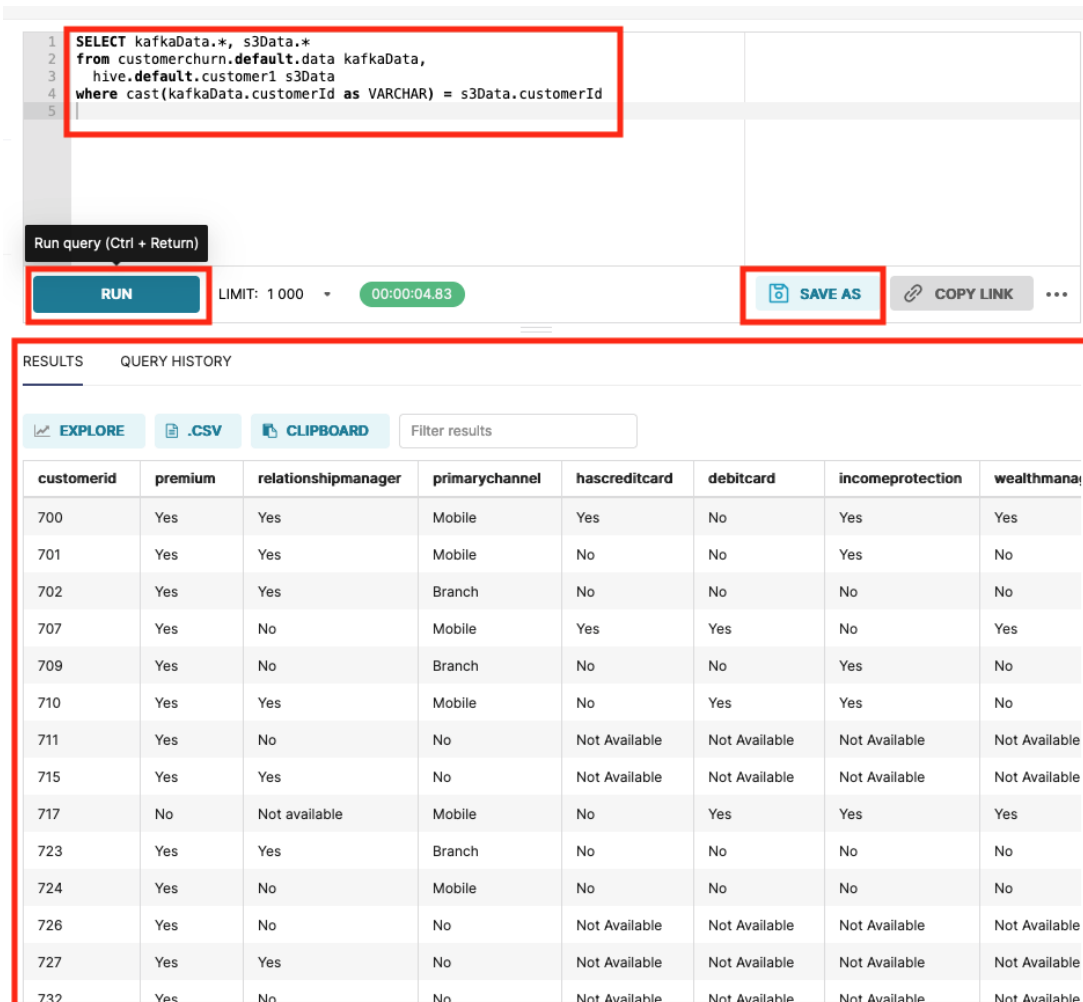
Earlier the workshop admin created a virtual '**table**' (hive.default.customer1) that uses the CSV data in our Minio S3 Object store as it's actual data - located in the bucket **rawdata**.

We also created second virtual '**table**' is backed by our Kafka streaming data. In our case this is the customer product consumption data. We also set this up earlier during the workshop provisioning.

```
SELECT kafkaData.*, s3Data.*  
from customerchurn.default.data kafkaData,  
hive.default.customer1 s3Data  
where cast(kafkaData.customerId as VARCHAR) = s3Data.customerId
```

Now Trino allows us to create a SQL Join across data that resides in S3 Object storage and Kafka!

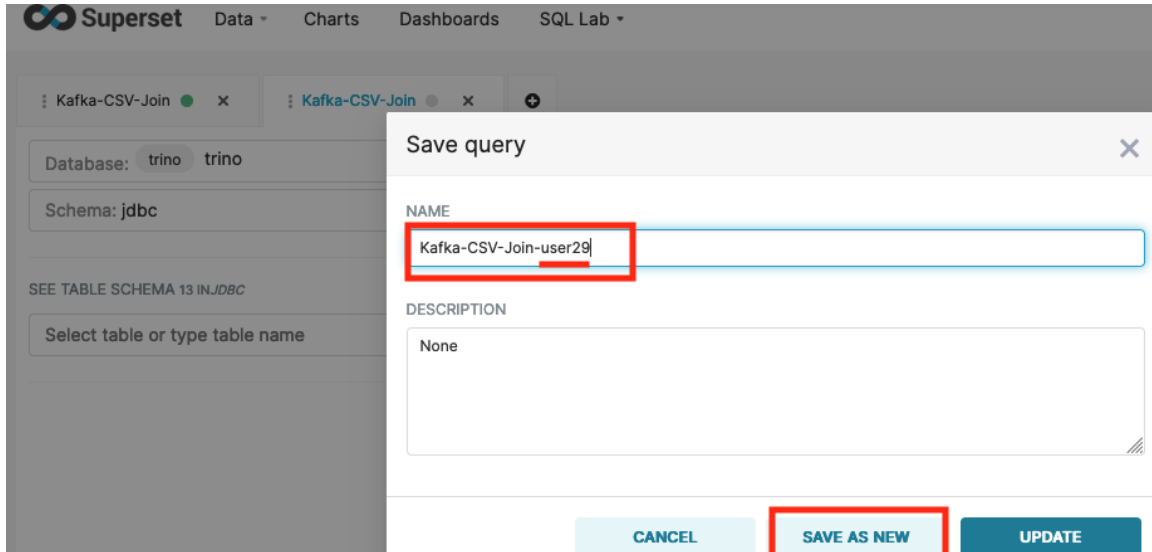
Very cool!



The screenshot shows the Trino SQL interface. At the top, a SQL query is entered in a text area, highlighted with a red box. Below the query area, there is a 'Run query (Ctrl + Return)' button, a 'RUN' button (also highlighted with a red box), a 'LIMIT' dropdown set to '1 000', a timer showing '00:00:04.83', a 'SAVE AS' button (highlighted with a red box), and a 'COPY LINK' button. Below the query execution controls, the 'RESULTS' tab is selected, showing a table of query results. The table has 8 columns: customerid, premium, relationshipmanager, primarychannel, hascreditcard, debitcard, incomeprotection, and wealthmanager. The results are displayed in a table with 15 rows, with the first row highlighted in blue. The table is also highlighted with a red box.

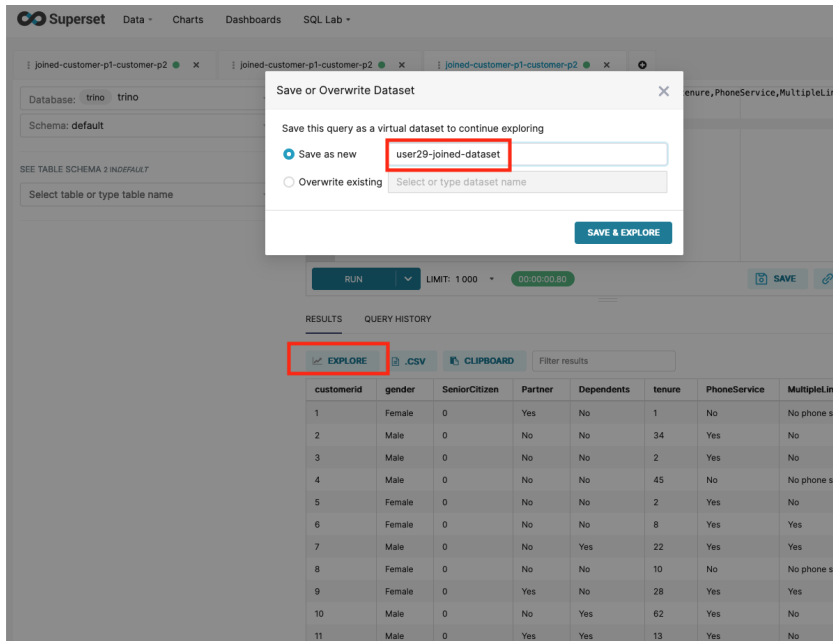
customerid	premium	relationshipmanager	primarychannel	hascreditcard	debitcard	incomeprotection	wealthmanager
700	Yes	Yes	Mobile	Yes	No	Yes	Yes
701	Yes	Yes	Mobile	No	No	Yes	No
702	Yes	Yes	Branch	No	No	No	No
707	Yes	No	Mobile	Yes	Yes	No	Yes
709	Yes	No	Branch	No	No	Yes	No
710	Yes	Yes	Mobile	No	Yes	Yes	No
711	Yes	No	No	Not Available	Not Available	Not Available	Not Available
715	Yes	Yes	No	Not Available	Not Available	Not Available	Not Available
717	No	Not available	Mobile	No	Yes	Yes	Yes
723	Yes	Yes	Branch	No	No	No	No
724	Yes	No	Mobile	No	No	No	No
726	Yes	No	No	Not Available	Not Available	Not Available	Not Available
727	Yes	Yes	No	Not Available	Not Available	Not Available	Not Available
732	Yes	No	No	Not Available	Not Available	Not Available	Not Available

Click **Save As** and **Save as New**. Append your username to the name, I was user29.



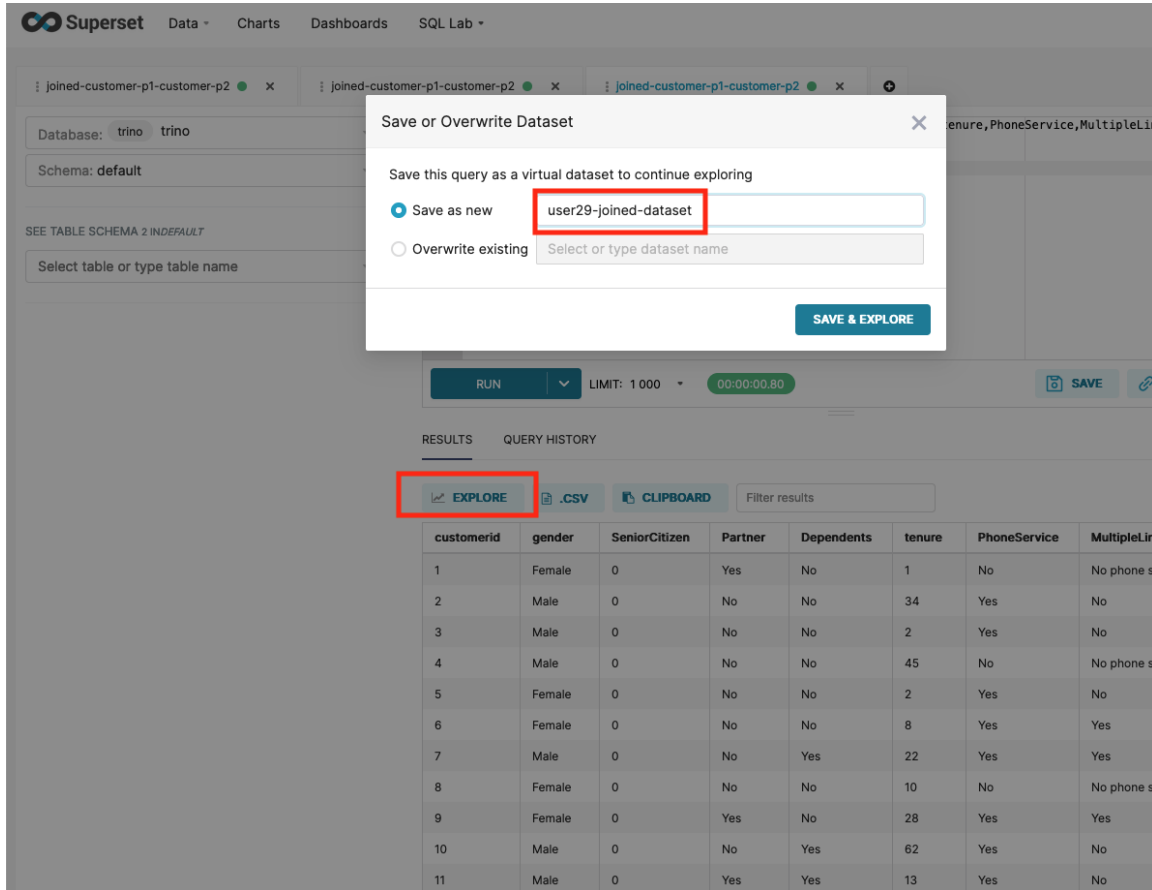
The image shows the 'Save query' dialog box in the Superset interface. The 'NAME' field is highlighted with a red box and contains the text 'Kafka-CSV-Join-user29'. The 'DESCRIPTION' field is empty and contains the text 'None'. At the bottom, the 'SAVE AS NEW' button is highlighted with a red box.

Notice it's a **dataset** comprising demographic customer data joined to product consumption customer data - joined on customer id. Click Explore then Save as a new query in the format **userXX-joined-dataset**. I was **user29** so I saved mine as **user29-joined-dataset**:



The image shows the 'Save or Overwrite Dataset' dialog box in the Superset interface. The 'Save as new' option is selected, and the 'user29-joined-dataset' text is entered in the input field, which is highlighted with a red box. Below the dialog box, the 'EXPLORE' button is highlighted with a red box. The data table below shows customer information.

customerid	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLine
1	Female	0	Yes	No	1	No	No phone service
2	Male	0	No	No	34	Yes	No
3	Male	0	No	No	2	Yes	No
4	Male	0	No	No	45	No	No phone service
5	Female	0	No	No	2	Yes	No
6	Female	0	No	No	8	Yes	Yes
7	Male	0	No	Yes	22	Yes	Yes
8	Female	0	No	No	10	No	No phone service
9	Female	0	Yes	No	28	Yes	Yes
10	Male	0	No	Yes	62	Yes	No
11	Male	0	Yes	Yes	13	Yes	No



Superset Data Charts Dashboards SQL Lab

joined-customer-p1-customer-p2 x joined-customer-p1-customer-p2 x joined-customer-p1-customer-p2 x

Database: trino trino

Schema: default

SEE TABLE SCHEMA 2 IN DEFAULT

Select table or type table name

Save or Overwrite Dataset

Save this query as a virtual dataset to continue exploring

☒ Save as new user29-joined-dataset

☐ Overwrite existing Select or type dataset name

SAVE & EXPLORE

RUN LIMIT: 1 000 00:00:00.80 SAVE

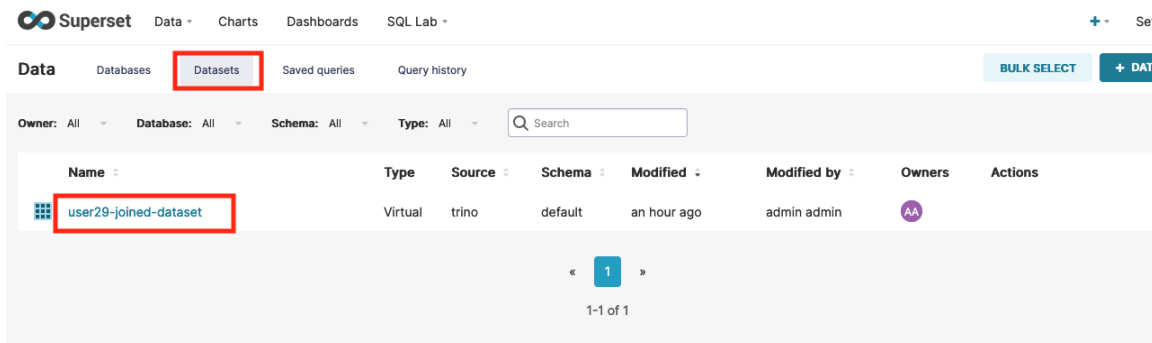
RESULTS QUERY HISTORY

EXPLORE .CSV CLIPBOARD Filter results

customerid	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLine
1	Female	0	Yes	No	1	No	No phone se
2	Male	0	No	No	34	Yes	No
3	Male	0	No	No	2	Yes	No
4	Male	0	No	No	45	No	No phone se
5	Female	0	No	No	2	Yes	No
6	Female	0	No	No	8	Yes	Yes
7	Male	0	No	Yes	22	Yes	Yes
8	Female	0	No	No	10	No	No phone se
9	Female	0	Yes	No	28	Yes	Yes
10	Male	0	No	Yes	62	Yes	No
11	Male	0	Yes	Yes	13	Yes	No

**You may see an error being reported. This is a small bug. The dataset has most likely been created. Proceed to the next step.**

Move to DataSets, find your new dataset - then click it to open it:




Superset Data Charts Dashboards SQL Lab

Data Databases **Datasets** Saved queries Query history

BULK SELECT + DATA

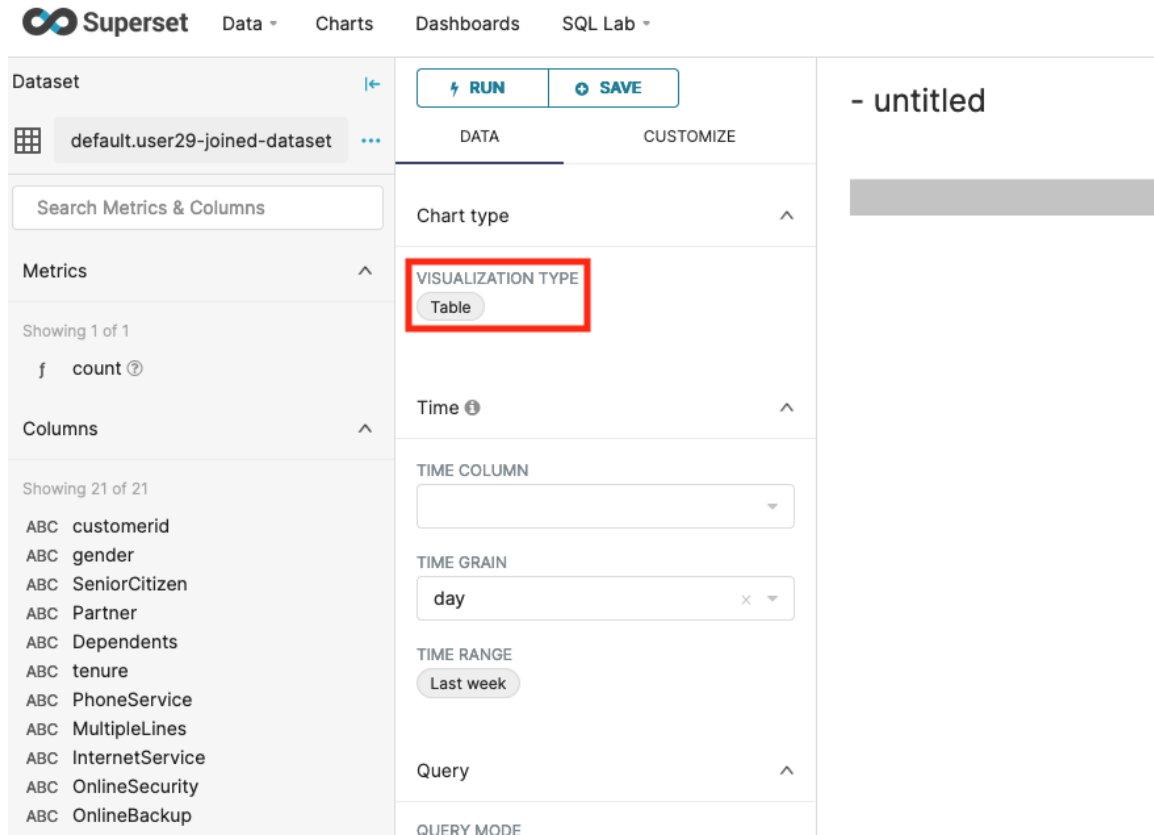
Owner: All Database: All Schema: All Type: All Search

Name	Type	Source	Schema	Modified	Modified by	Owners	Actions
 user29-joined-dataset	Virtual	trino	default	an hour ago	admin admin	AA	

« 1 »

1-1 of 1

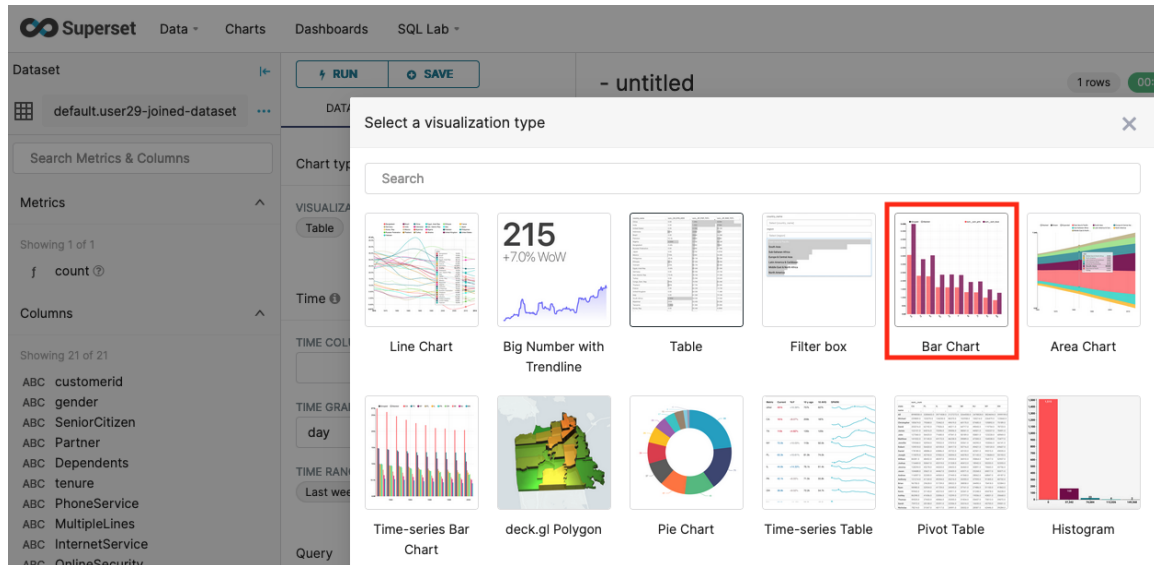
By default **Table** Visualisation Type is selected



The screenshot displays the Red Hat Superset web interface. At the top, the navigation bar includes the Superset logo and tabs for Data, Charts, Dashboards, and SQL Lab. The main interface is divided into three sections. On the left, the 'Dataset' panel shows 'default.user29-joined-dataset' and a search bar for metrics and columns. Below this, the 'Metrics' section shows 'count' and the 'Columns' section lists various attributes like 'customerid', 'gender', 'SeniorCitizen', etc. The central panel has tabs for 'DATA' and 'CUSTOMIZE'. The 'DATA' tab is active, showing the 'Chart type' section where 'Table' is selected under the 'VISUALIZATION TYPE' heading. Other settings like 'Time', 'Time Column', 'Time Grain' (set to 'day'), and 'Time Range' (set to 'Last week') are visible. The right panel shows a placeholder for the visualization, labeled '- untitled'.



Click Table. You have a large number of Visualization Types to choose from. Choose Bar Chart by clicking on it:



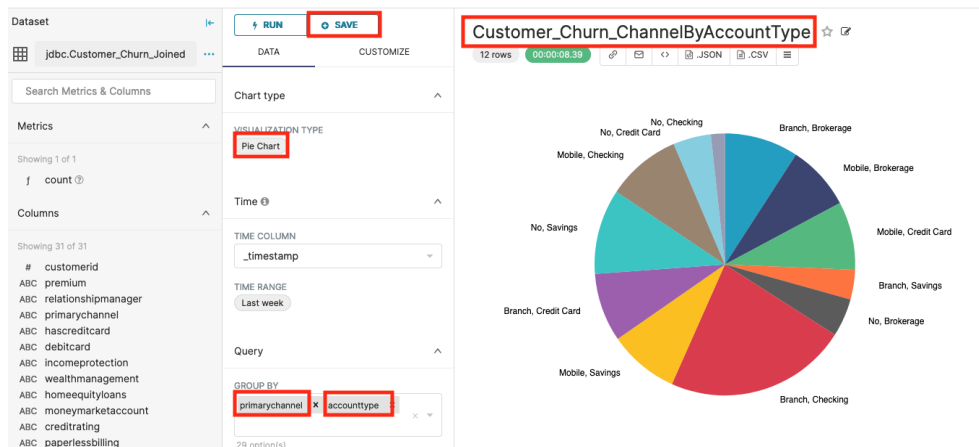
You can now start visualising and understanding your data. For our purposes, we'll create a nice bar chart representation of the entire dataset, of the count of the different categories of *primary channel* - *Branch*, *Mobile* or *No* primary channel. Select **Count** under **Metrics** and select *PrimaryChannel* under **Series**. Name the chart something (in my case **Customer\_Churn\_By\_PrimaryChannel**) and save it. (you may need to remove the Time Range - **last week** in the screenshot)

Then Run Query

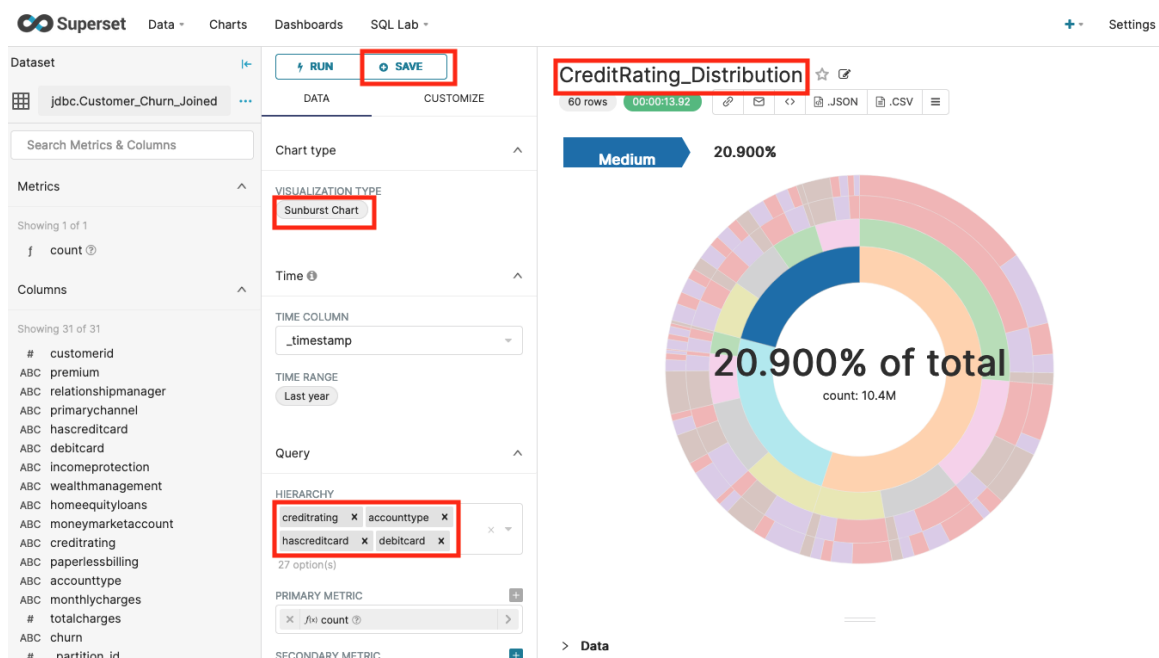
This is very useful to understand the data.



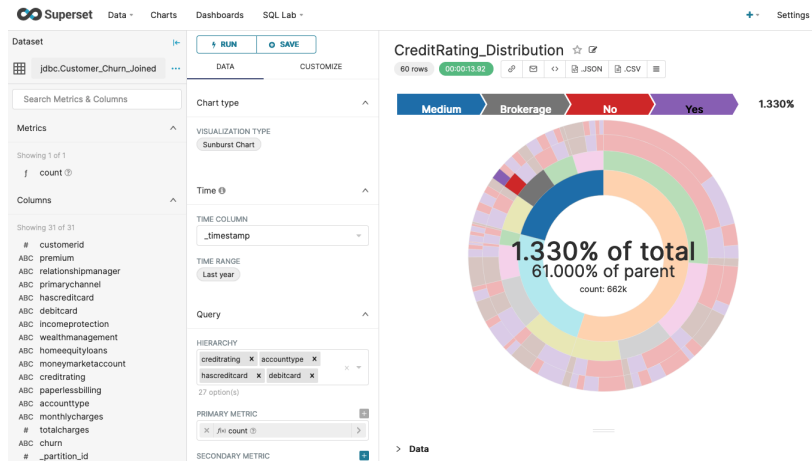
Now let's create a Pie Chart - and add another dimension - *account type*. If you now choose Pie Chart (where you previously chose Bar Chart) - and run the query again, it's that easy to get a new view type with this extra dimension. Again name (*Count\_ChannelByAccountType*) and save the file:



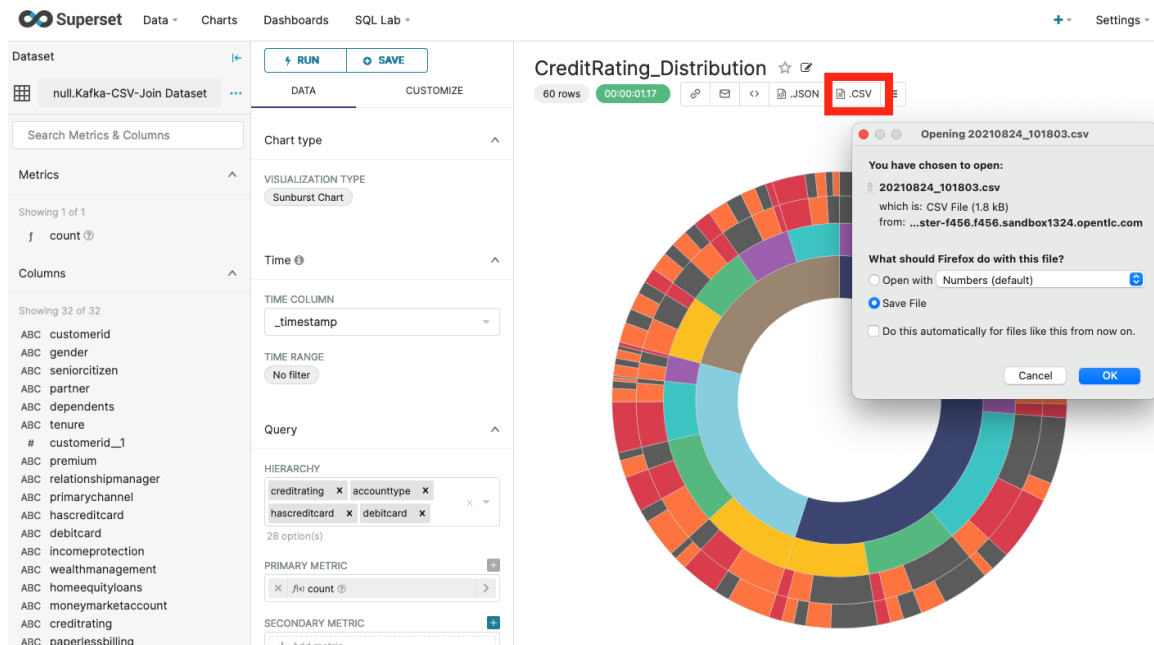
Next, we're going to showcase another unusual though informative visualization - the Sunburst Chart. It allows you to visualise splits of your data with varying degrees of granularity - on the same chart. Make the selection as follows and save the chart (in my case *CreditRating\_Distribution*). If you hover over the inner circle, the breakdown according to the first hierarchical element is shown, in my case *medium* credit rating.



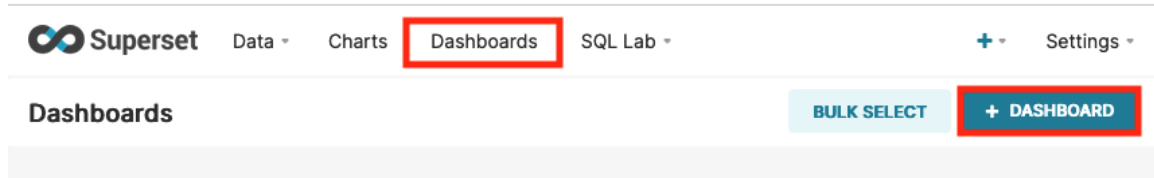
You can see, you can also further refine, by hovering out towards over the outer circle - giving a very fine grained breakdown - according to the 4 hierarchies:



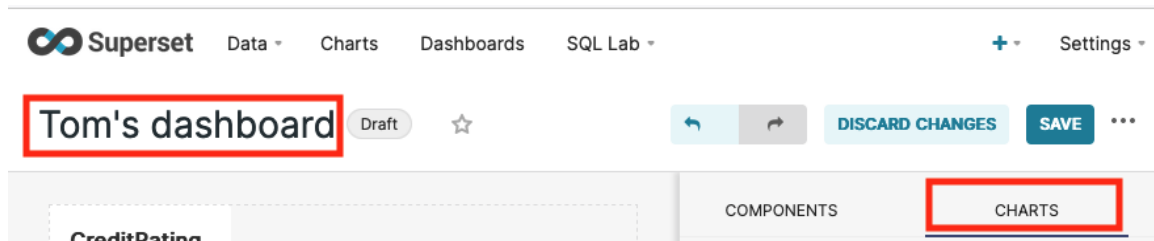
And any of these datasets can be easily exported to JSON or CSV - as shown below. Then fed for example to an AI model training use case.



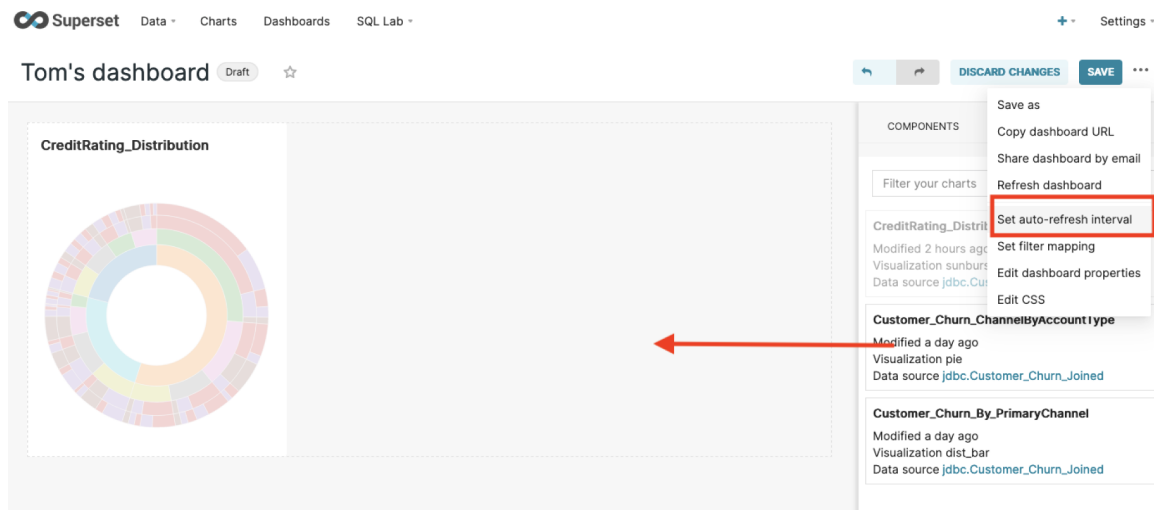
Finally we're going to show you how to create a dashboard - to which we can add previously saved charts. Choose Dashboards -> Add Dashboard as shown:



Name it and choose the Charts tab:



You can simply drag your charts from the right over to the display panel on the left as shown. You can also make them dynamic by choosing a refresh interval. Very cool!



Feel free to continue to experiment different ways of accessing and visualising the underlying data.

When you're finished, move to the next lab: [Lab 2 - Data Engineer prepares data](#).