

Lab 4 - Data Analytics

Introduction

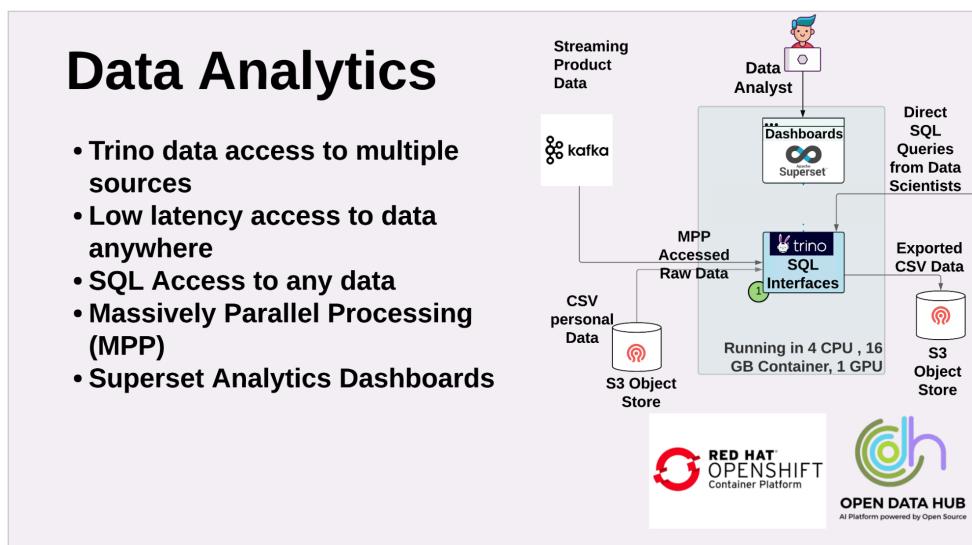
One of the first things your data analysts and data engineers will need to do is analyse the raw data, with a view to preparing and transforming it to a state that will be consumable by AI/ML model algorithms.

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, 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. No data is moved – rather Trino provides a high speed parallel access mechanism for Kafka and S3 allowing Superset to easily display charts and dashboards.



To save time, the workshop administrators have already wired up the SQL exposing engine Trino to two backing 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.

Instructions for the Trino backed Superset workshop

Login to OpenShift using the credentials your administrator gave you. Choose **Administrator** from the 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).

Login to OpenShift using the credentials your administrator gave you. Ensure your workshop project ml-workshop is selected.

The first thing you will do is login to Superset.

Choose the **Administration perspective**

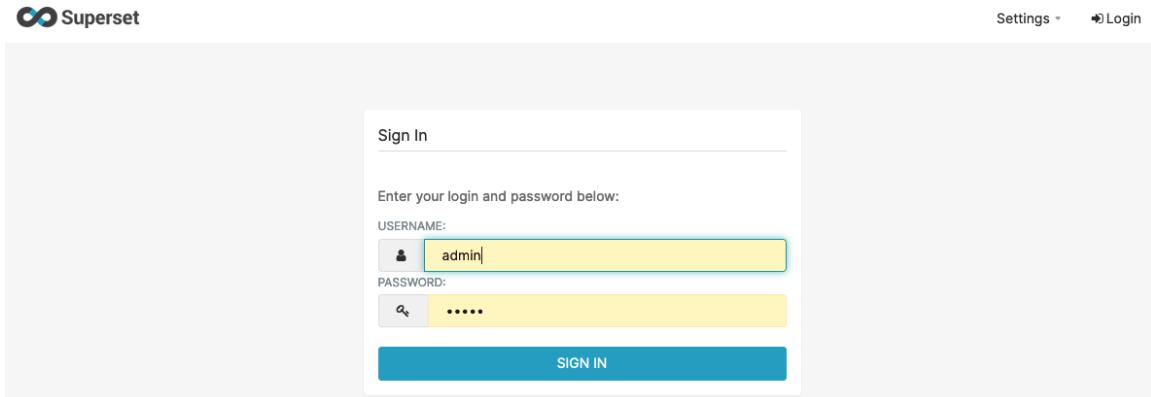
1. Navigate to **Networking > Routes**.
2. Filter on the word Superset and open that route, by clicking on the URL as shown.

The screenshot shows the OpenShift UI for Networking > Routes. The left sidebar has 'Administrator' selected in the dropdown. The 'Routes' section is highlighted. A filter bar at the top has 'Project: ml-workshop' and a 'Name' filter set to 'superset'. The main table lists one route: 'superset' with status 'Accepted' and location 'http://superset-ml-workshop.apps.cluster-bffc.bffc.sandbox60.opentlc.com'. The URL in the location column is highlighted with a red box.

Name	Status	Location
superset	Accepted	http://superset-ml-workshop.apps.cluster-bffc.bffc.sandbox60.opentlc.com



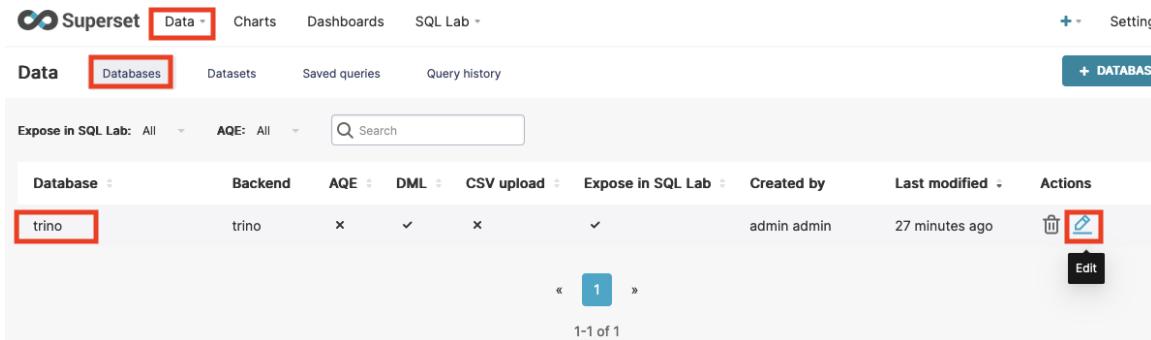
3. Enter credentials **admin / admin**.



The screenshot shows the Superset sign-in interface. At the top right are "Settings" and "Login" buttons. Below them is a "Sign In" form with the heading "Sign In" and the sub-instruction "Enter your login and password below:". It has two input fields: "USERNAME:" containing "admin" and "PASSWORD:" containing ".....". At the bottom is a blue "SIGN IN" button.

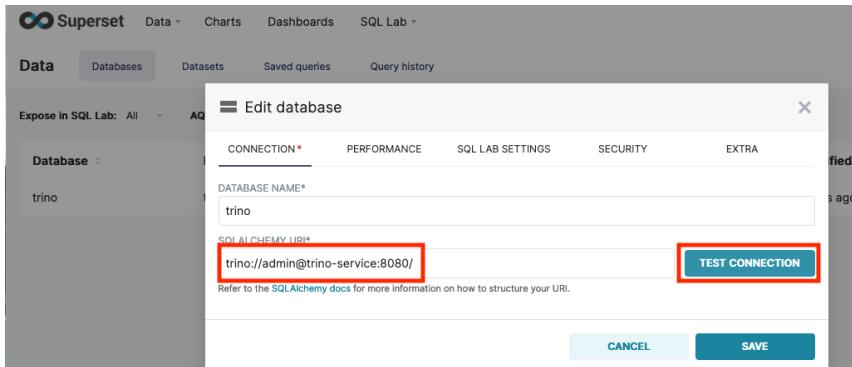
4. Choose menu **Data > Databases**. Edit the **trino** Database

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.

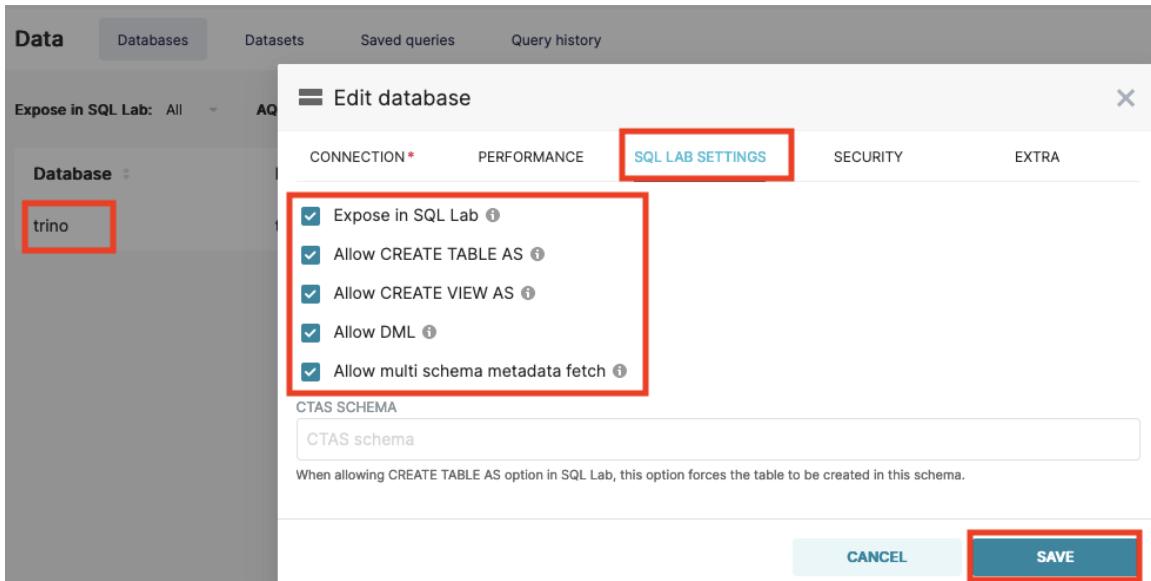


The screenshot shows the Superset "Databases" page. The top navigation bar includes "Superset" (with a gear icon), "Data" (which is highlighted with a red box), "Charts", "Dashboards", "SQL Lab", and "Setting". Below the navigation is a secondary header with tabs: "Data" (highlighted with a red box), "Databases" (highlighted with a red box), "Datasets", "Saved queries", and "Query history". A "DATABASE" button is located at the top right of this header. The main content area displays a table of databases. The first row, "trino", is also highlighted with a red box. The table columns are: Database, Backend, AQE, DML, CSV upload, Expose in SQL Lab, Created by, Last modified, and Actions. The "Actions" column for "trino" contains a trash bin icon and an edit icon (highlighted with a red box). At the bottom of the table, there are navigation arrows, a page number "1", and the text "1-1 of 1".

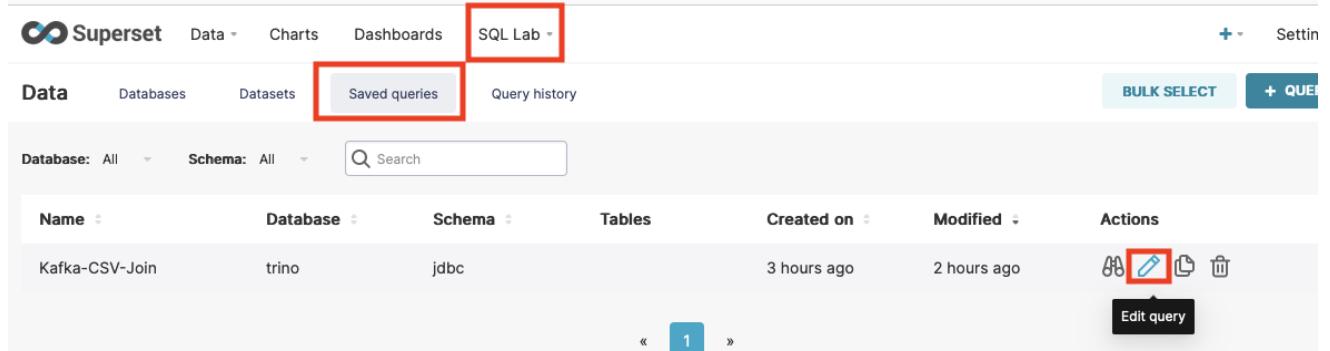
Notice we simply added the URI **trino://admin@trino-service:8080/** to connect to Trino as shown. Test the Connection.



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



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



Name	Database	Schema	Tables	Created on	Modified	Actions
Kafka-CSV-Join	trino	jdbc		3 hours ago	2 hours ago	Edit query

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**.

Earlier we also created a second virtual '**table**' backed by our Kafka streaming data. In our case this is the customer product consumption data.

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

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

Very cool!



7. **Run** the Query. Notice the result set spanning data covering S3 and Kafka, joined on *customerId*. Click **Explore**

The screenshot shows the Apache Zeppelin SQL Lab interface. At the top, there is a code editor with the following SQL query:

```
1 SELECT kafkaData.*, s3Data.*  
2 FROM customerchurn.default.kafkaData,  
3      hive.default.customer1 s3Data  
4 WHERE cast(kafkaData.customerId AS VARCHAR) = s3Data.customerId
```

Below the code editor is a toolbar with a 'RUN' button (highlighted with a red box), a 'LIMIT' dropdown set to 1000, and a timer showing 00:00:07:17. To the right of the toolbar are 'SAVE', 'COPY LINK', and a three-dot menu button.

The main area displays the query results as a table. The table has 12 columns: customerId, premium, relationshipmanager, primarychannel, hascreditcard, debitcard, incomeprotection, wealthmanagement, homeequityloans, moneymarketaccount, and creditrating. The data consists of 12 rows, each containing values for these columns. The 'EXPLORER' tab is highlighted with a red box at the top of the results table.

customerId	premium	relationshipmanager	primarychannel	hascreditcard	debitcard	incomeprotection	wealthmanagement	homeequityloans	moneymarketaccount	creditrating
3	Yes	No	Mobile	Yes	Yes	No	No	No	No	High
4	No	Not available	Mobile	Yes	No	Yes	Yes	No	No	Medium
12	Yes	No	No	Not Available	Not Available	Not Available	Not Available	Not Available	Not Available	Low
13	Yes	Yes	Branch	No	No	Yes	No	Yes	Yes	Medium
17	Yes	No	No	Not Available	Not Available	Not Available	Not Available	Not Available	Not Available	Medium
25	Yes	No	Mobile	Yes	Yes	No	Yes	No	No	High
28	No	Not available	Mobile	No	Yes	No	No	No	No	High
31	Yes	Yes	Branch	Yes	Yes	Yes	Yes	No	No	Low
36	Yes	Yes	Branch	Yes	Yes	No	Yes	Yes	No	Low
38	Yes	No	Branch	No	No	Yes	No	No	No	High
39	Yes	Yes	Branch	No	Yes	Yes	No	Yes	Yes	High
42	Yes	Yes	Mobile	Yes	Yes	No	No	Yes	No	Low

8. Select **Save As New**

Give it a name appending your username to **Kafka-CSV-Join**. In my case, with user29: **Kafka-CSV-Join-user29**. Then **Save & Explore**:

The screenshot shows a dialog box titled 'Save or Overwrite Dataset'. It contains the following text: 'Save this query as a virtual dataset to continue exploring'. Below this are two radio buttons: 'Save as new' (highlighted with a red box) and 'Overwrite existing'. The 'Save as new' field contains the value 'Kafka-CSV-Join-user29'. Below the radio buttons is a text input field labeled 'Select or type dataset name' with the placeholder 'Select or type dataset name'. At the bottom right of the dialog is a large blue 'SAVE & EXPLORE' button (highlighted with a red box).



You may see an error saving the dataset. Ignore it (this is a small bug) - it probably saved successfully.

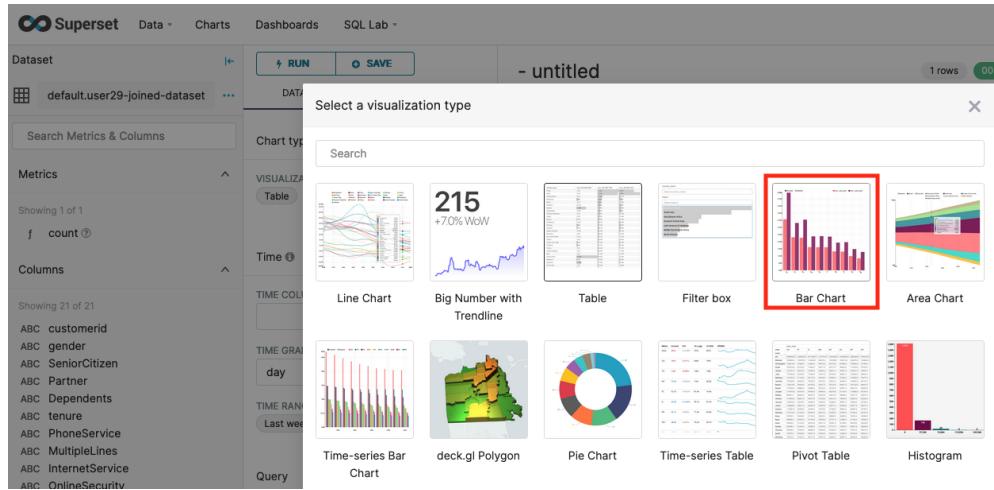
9. Move to **Data > DataSets**, find your new dataset - then click it to open it:

The screenshot shows the Superset interface with the 'Data' tab selected. Under the 'Datasets' tab, there is a list of datasets. The first dataset, 'user29-joined-dataset', is highlighted with a red box. The list includes columns for Name, Type, Source, Schema, Modified, Modified by, Owners, and Actions. At the bottom, there is a page number '1' and a note '1-1 of 1'.

10. By default **Table** Visualisation Type is selected

The screenshot shows the Superset interface for a specific dataset named 'default.user29-joined-dataset'. On the left, there are sections for Metrics and Columns. On the right, there are tabs for RUN and SAVE, and sections for Chart type, Visualization Type (with 'Table' selected), Time, and Query. The 'Table' visualization type is highlighted with a red box.

Click **Table**. You have a large number of Visualization Types to choose from. Click **Bar Chart**:



You can now start visualising and understanding your data. First we'll create a bar chart representation of the entire dataset, representing the count of the different categories of *Primary Channel*:

- *Branch*
- *Mobile and*
- *No primary channel.*



Click **Last Week** under **TIME RANGE**. On the popup, click the **Range Type** dropdown, select **No Filter** and then **Apply**.

Chart type

VISUALIZATION Bar Chart

Time

TIME COLUMN _timestamp

TIME RANGE Last week

Query

METRICS f(x) count

FILTERS + Add filter

SERIES 1

Edit time range

RANGE TYPE

- Last
- Previous
- Custom
- Advanced
- No filter
- last month
- last quarter
- last year

Actual time range

2021-10-07 ≤ col < 2021-10-14

CANCEL APPLY

Select **Count** under **Metrics** and select **PrimaryChannel** under **Series**.

Name the chart **Count-PrimaryChannel-userXX** (in my case

Count-PrimaryChannel-user29) and click **Save**.

DATA CUSTOMIZE

RUN SAVE

Count-PrimaryChannel-user29

1 rows 00:00:03.35

DATA CUSTOMIZE

Chart type

VISUALIZATION TYPE Bar Chart

Time

TIME COLUMN _timestamp

TIME RANGE No filter

Query

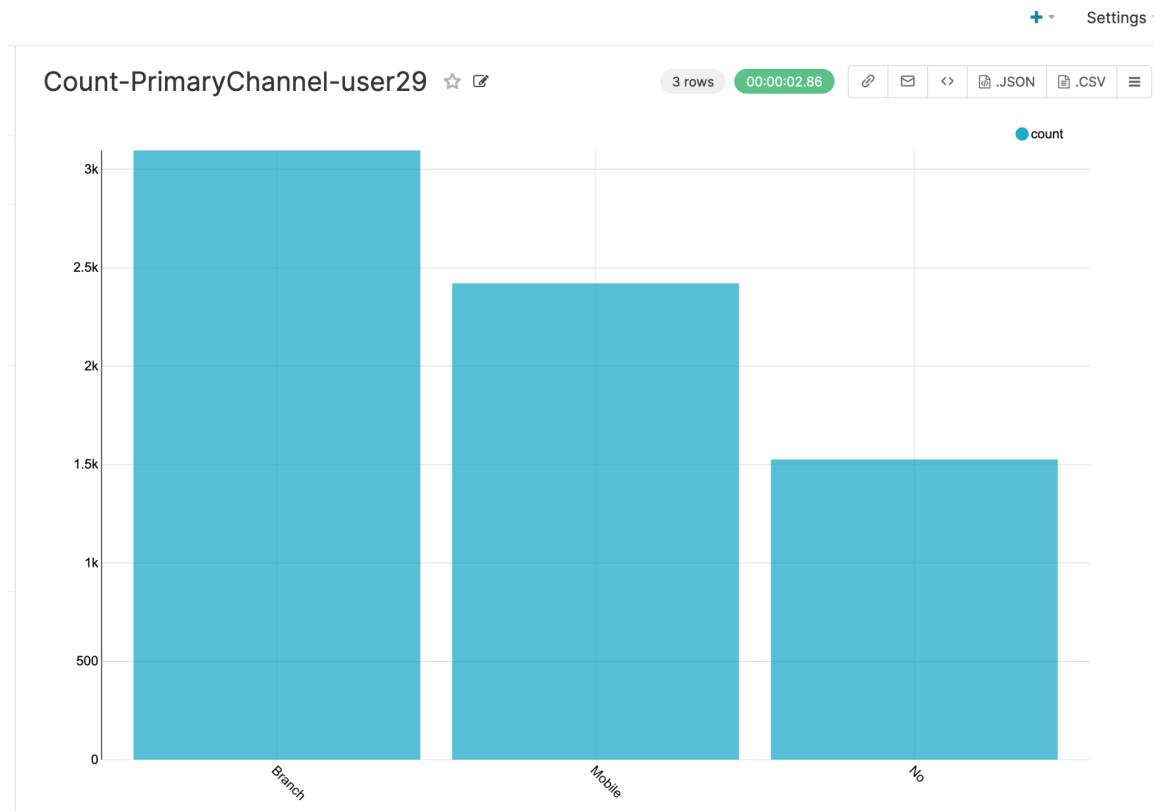
METRICS f(x) count

FILTERS + Add filter

SERIES primarychannel

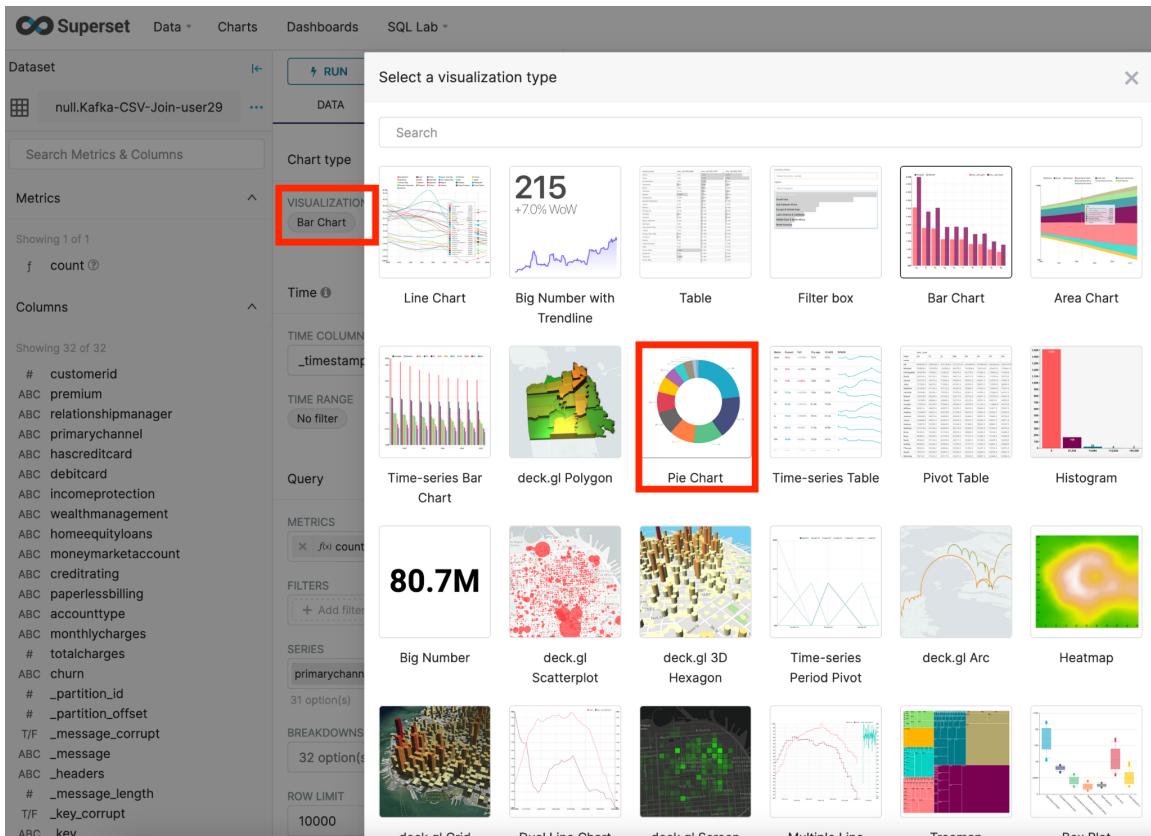
RUN QUERY

This will cause the query to run - and you will be presented with this bar chart - showing the breakdown of the resultset by Primary Channel, Branch, Mobile or None.



11. Now we'll create a Pie Chart - and add another dimension - *account type*.

On your open Bar Chart screen, click **Bar Chart**. On the popup click **Pie Chart**.

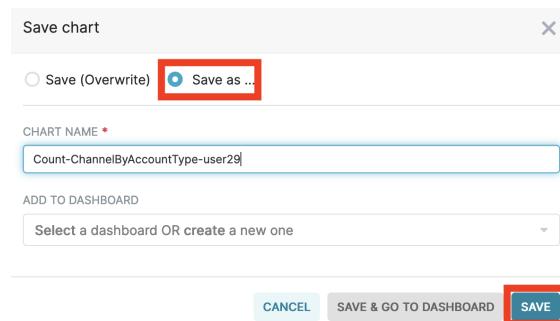


The screenshot shows the Superset interface with the following details:

- Left Sidebar (Dataset):** Shows a dataset named "null.Kafka-CSV-Join-user29".
- Left Sidebar (Metrics):** Shows a single metric "count".
- Left Sidebar (Columns):** Shows numerous columns, including "customerid", "premium", "relationshipmanager", "primarychannel", "hascreditcard", "debitcard", "incomeprotection", "wealthmanagement", "homeequityloans", "moneymarketaccount", "creditrating", "paperlessbilling", "accounttype", "monthlycharges", "totalcharges", "churn", "partition_id", "partition_offset", "message_corrupt", "message", "headers", "message_length", "key_corrupt", and "key".
- Top Bar:** Includes links for "Data", "Charts", "Dashboards", and "SQL Lab".
- Central Area:** A modal titled "Select a visualization type" displays a grid of visualization options. The "Bar Chart" option is highlighted with a red box in the top row.
- Bottom Row:** A row of visualization thumbnails includes "Big Number", "deck.gl Scatterplot", "deck.gl 3D Hexagon", "Time-series Period Pivot", "Treemap", and "Row Plot".

Now make the following changes:

- On the **Group By** dropdown, add **accounttype**
- Name the chart **Count-ChannelByAccountType-userXX** (in my case **Count-ChannelByAccountType-user29**) and click **Save**.
- Go with the defaults on the popup. Click **Save As** then **Save**

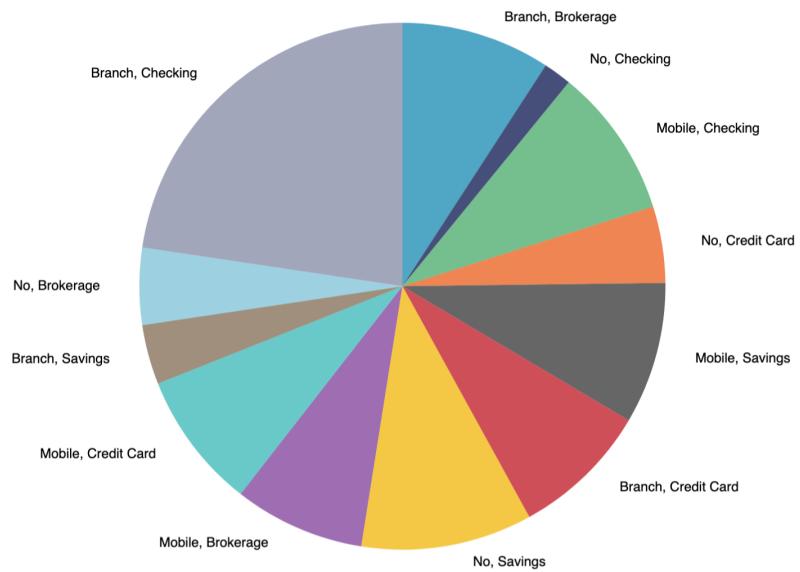


The dialog box has the following fields and buttons:

- Save chart** (title)
- Save (Overwrite)** (radio button)
- Save as...** (radio button, highlighted with a red box)
- CHART NAME *** (text input field) containing "Count-ChannelByAccountType-user29"
- ADD TO DASHBOARD** (button)
- Select a dashboard OR create a new one** (dropdown menu)
- CANCEL** (button)
- SAVE & GO TO DASHBOARD** (button)
- SAVE** (button, highlighted with a red box)

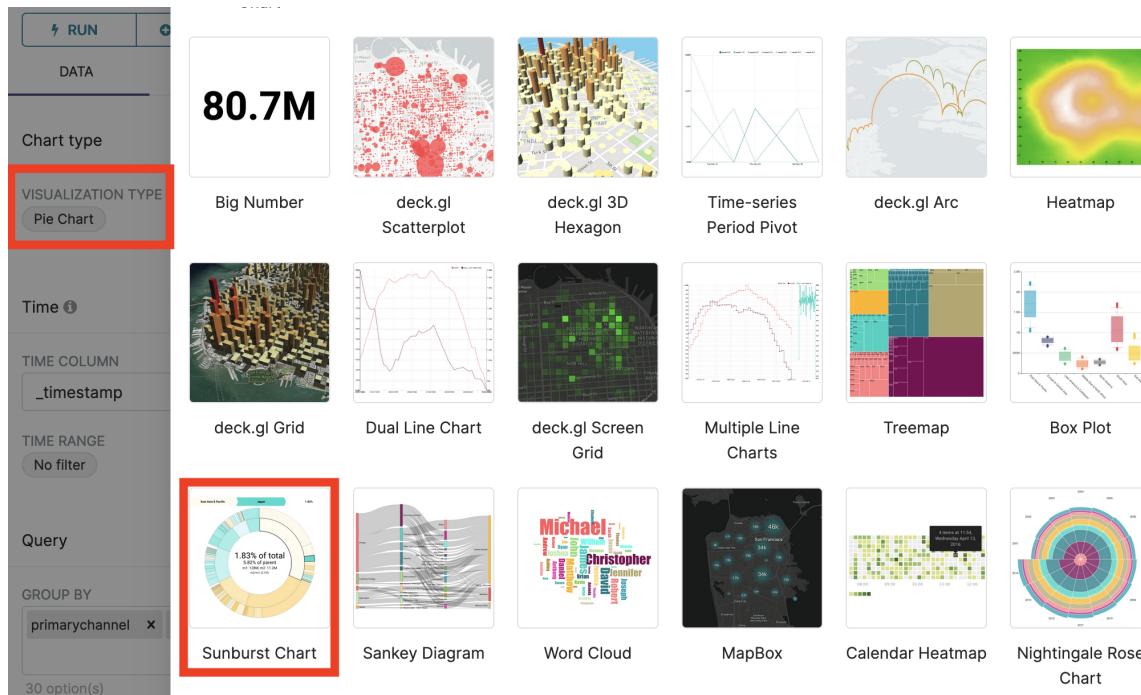
This will cause the following pie chart to display - grouping *Primary Channel* by Account Type

Count-ChannelByAccountType-user29 ⭐ ⓘ 12 rows 00:00:01.72 ⌂ ⌃ .JSON ⌃ .CSV ⌃



12. 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 – within the same chart.

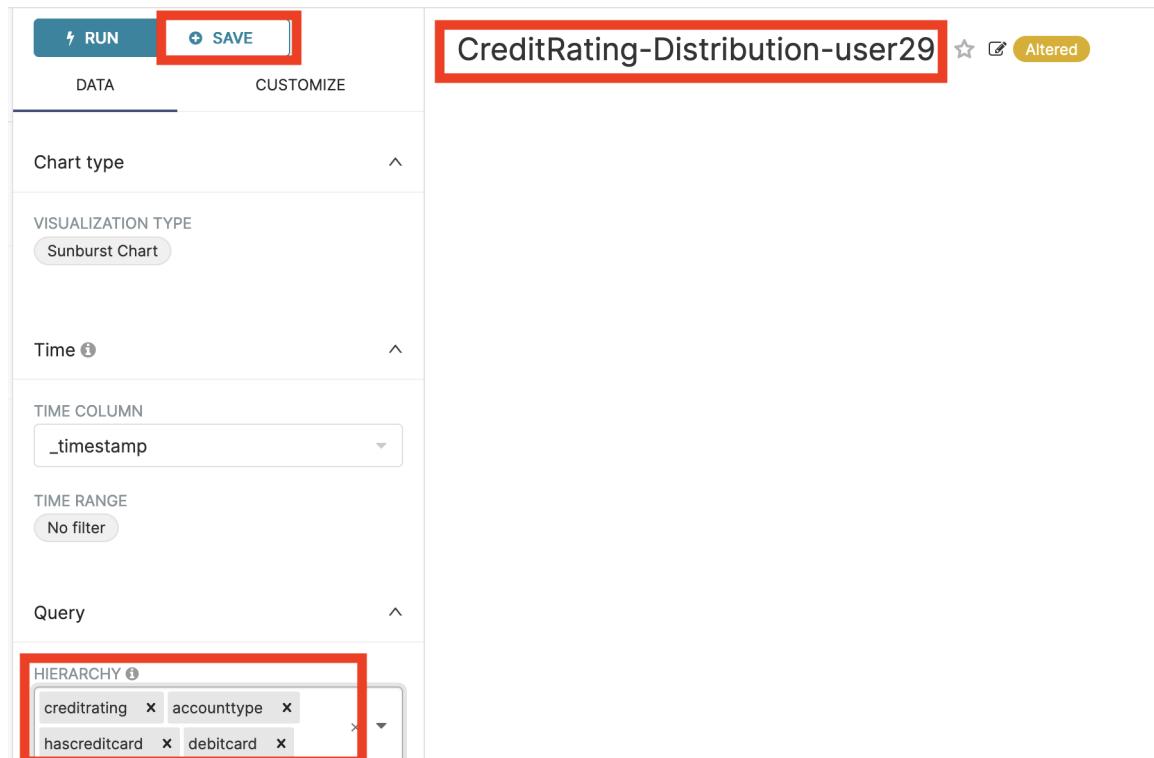
On your open Pie Chart screen, click **Pie Chart** then on the popup, scroll down and click **Sunburst Chart** as shown:



Make the following changes

- Name the chart **CreditRating-Distribution-userXX** (in my case **CreditRating-Distribution-user29**)
- On the **Hierarchy** dropdown, select
 - creditrating**
 - accounttype**
 - hascreditcard**
 - debitcard**
- click **Save**

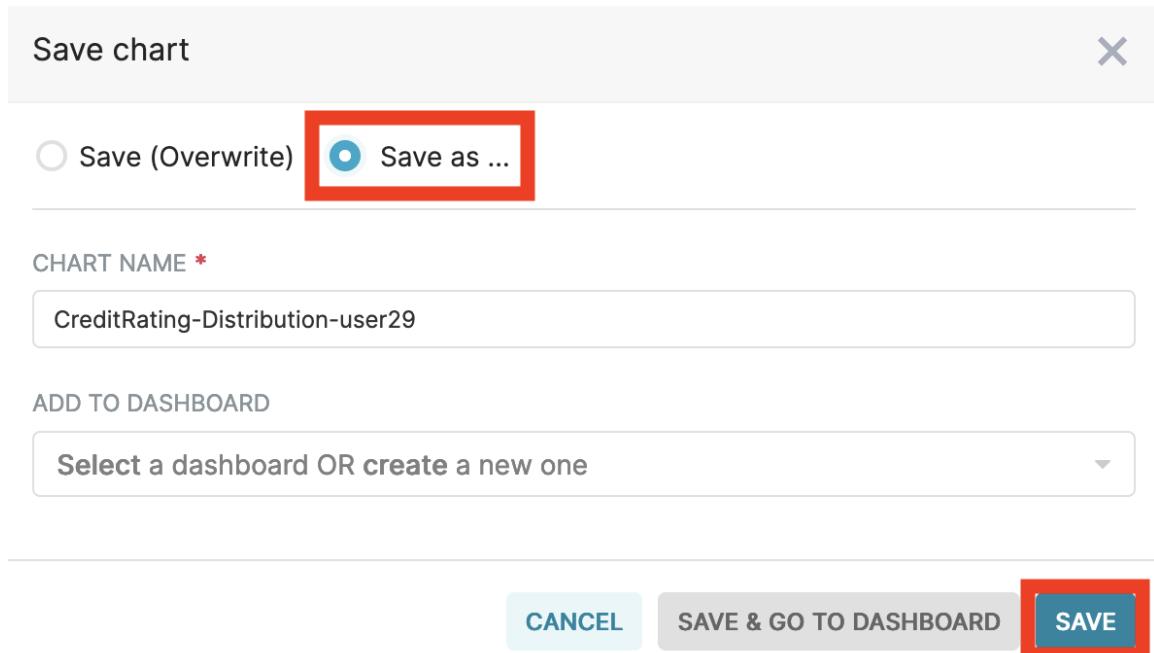
as shown:



The screenshot shows a data visualization interface with the following configuration:

- Top Bar:** Includes 'RUN' and 'SAVE' buttons, and 'DATA' and 'CUSTOMIZE' tabs.
- Title:** CreditRating-Distribution-user29 (highlighted with a red box).
- Chart Type:** Sunburst Chart.
- Time:** Set to '_timestamp'.
- Query Hierarchy:** Contains 'creditrating', 'accounttype', 'hascreditcard', and 'debitcard' (highlighted with a red box).

Confirm by clicking **Save As** then **Save**



Save chart

Save (Overwrite) Save as ... (highlighted with a red box)

CHART NAME *

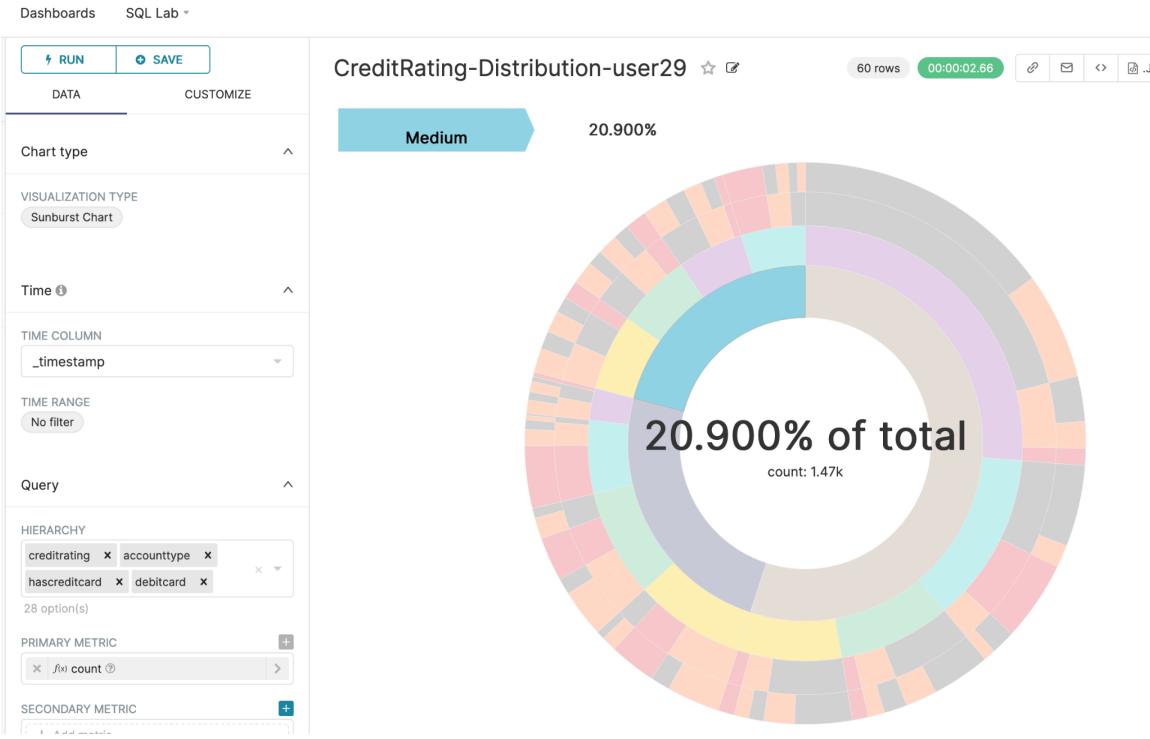
CreditRating-Distribution-user29

ADD TO DASHBOARD

Select a dashboard OR create a new one

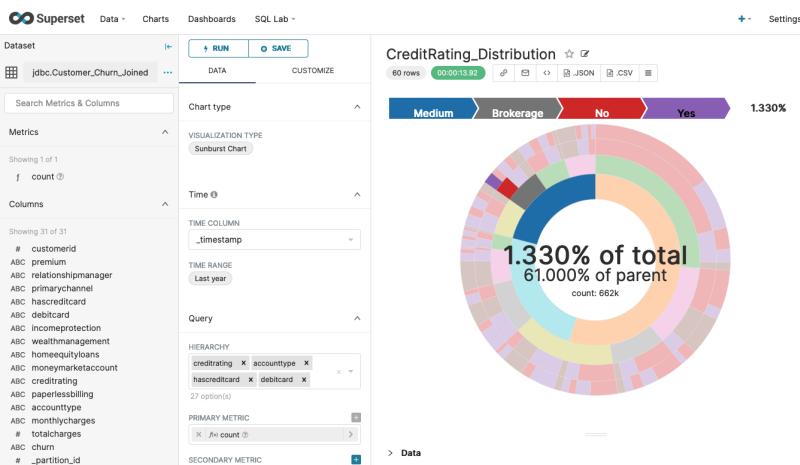
CANCEL **SAVE & GO TO DASHBOARD** **SAVE** (highlighted with a red box)

A chart similar to the following will appear.

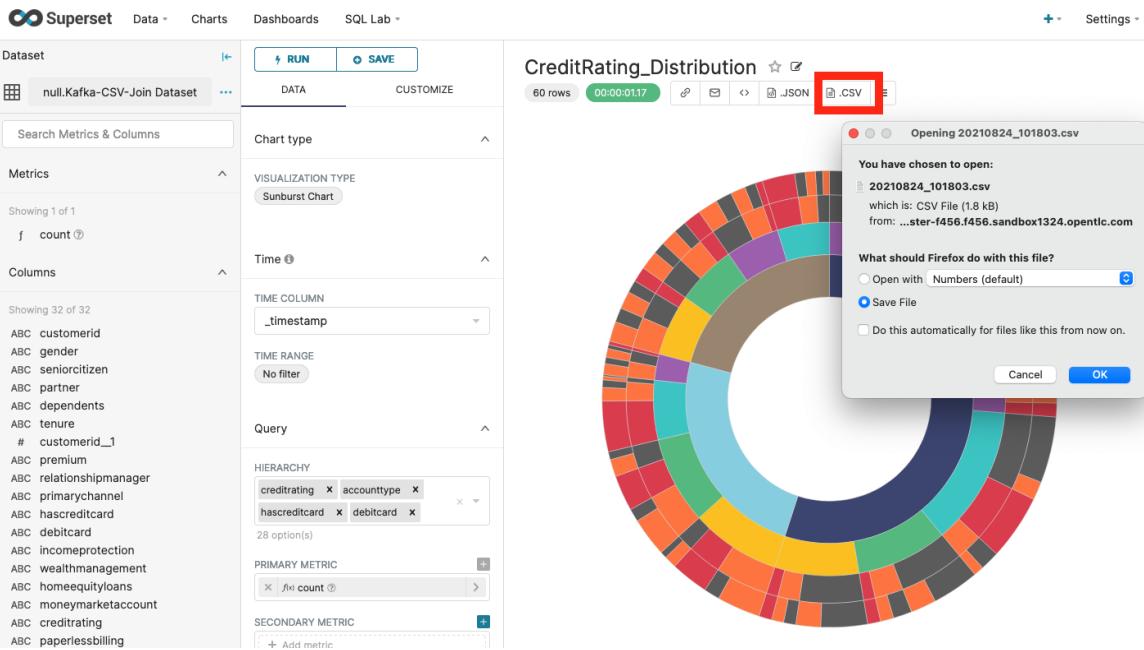


If you hover over the inner circle, the breakdown according to the first hierarchical element is shown, in my case *medium* credit rating.

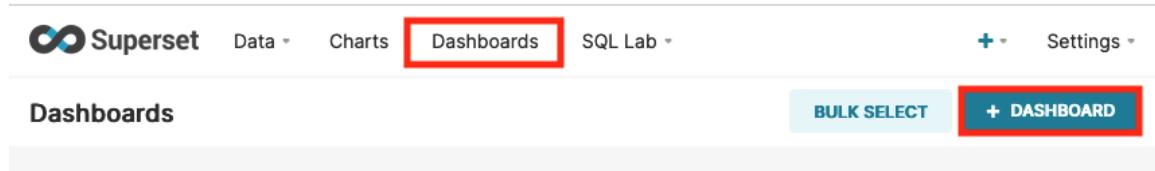
You can see, you can also further refine, by hovering out towards over the outer circle - giving a very fine grained breakdown - according to the 4 hierarchies:



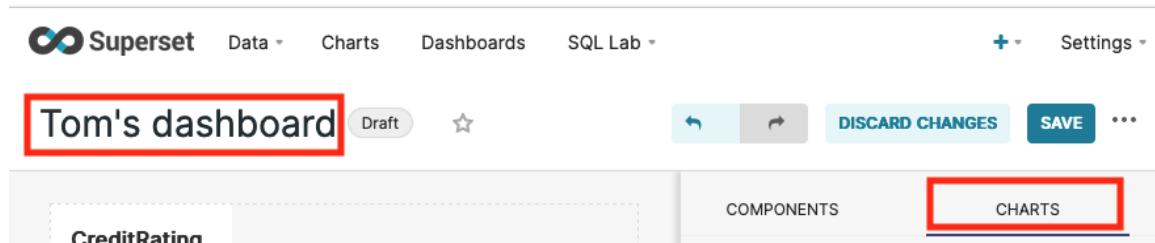
Any of these datasets can be easily exported to JSON or CSV – as shown below. Then fed for example to an AI model training use case.



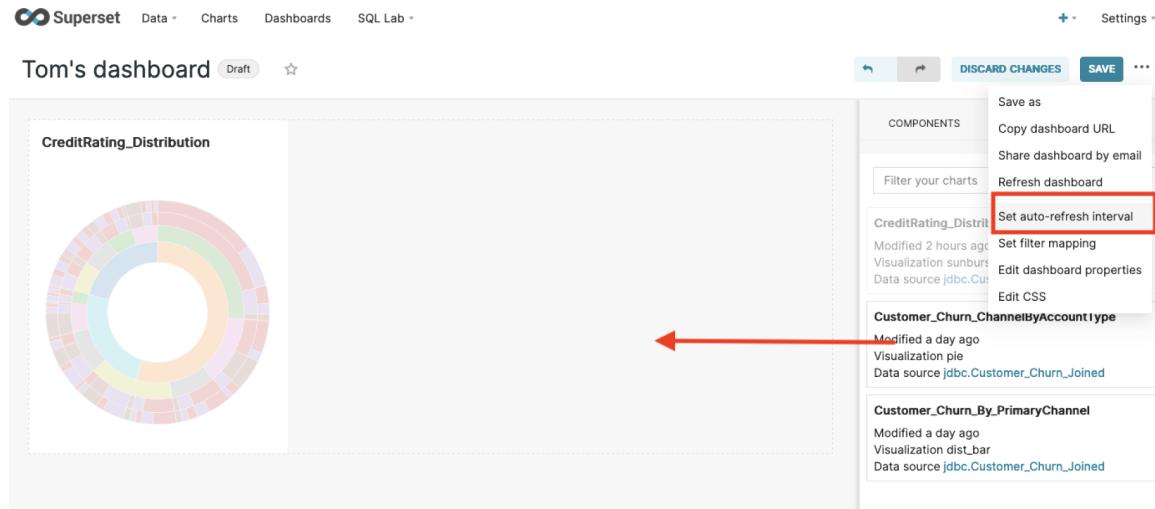
13. Finally we're going to show you how to create a dashboard - to which we can add previously saved charts. Choose Dashboards -> Add Dashboard as shown:



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.