Druga domaća zadaća iz Podatkovnog inženjerstva (English instructions)

Your task is to connect to the database and create several charts based on the data contained in the database, displaying them on a dashboard.

Database assignment

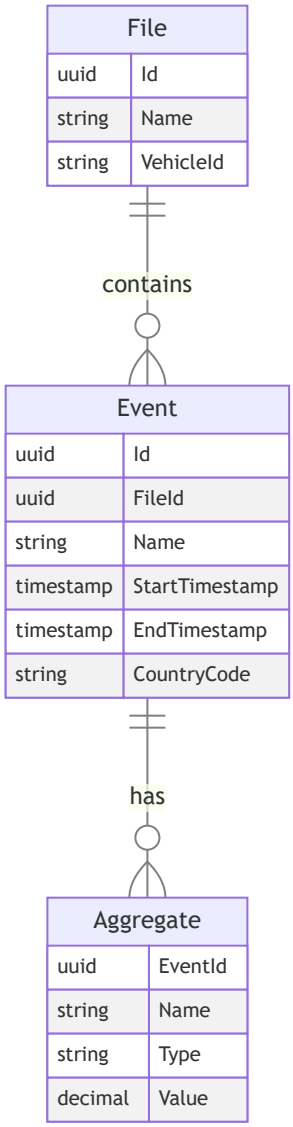
For the database and storage assignment we will work with PostgreSQL, more precisely using its extension TimescaleDB since this database is optimized for time series data storage.

Contents:

Data model

The data that you have already processed in the previous assignment can be represented with the ER diagram below.

- A single file can contain multiple events
  - A file is represented with its unique identifier and its name
- A single event instance is located in a single file
  - An event is represented with its unique identifier, its name and the start and end timestamps
    - The start and end timestamps represent when the event started and when it ended
  - A single event has multiple aggregations calculated on it
    - An event has to contain an unique combination of `Aggregate Name` and `Aggregate Type`, e.g. only a single aggregate `('Speed', 'max')` possible for a single event
- A single aggregate is connected to a single event
  - An aggregate is defined with its name, type and its value



Database

The database model with which we are going to work with is shown in the Data model section and the way we are going to create the tables.

Database connection credentials

- **host** -> `db-pmf-lab.postgres.database.azure.com`
- **port** -> `5432`
- **database** -> `pmf_lab_db`
- **schema** -> `public`
- **username** -> `pmf_lab_reader`
- **password** -> `S0oIdwSI7k`
- **display name** -> You can choose this on your own, this is the display name for Superset

Apache Superset

Official website for `Apache Superset` contains general documentation and guidelines on how to work with the tool.

Starting the Application

There are several ways how we can use install and use `Apache Superset` and from the available options we are going to use a docker image since it is easy to set it up locally and run it if `Docker` is installed on the computer.

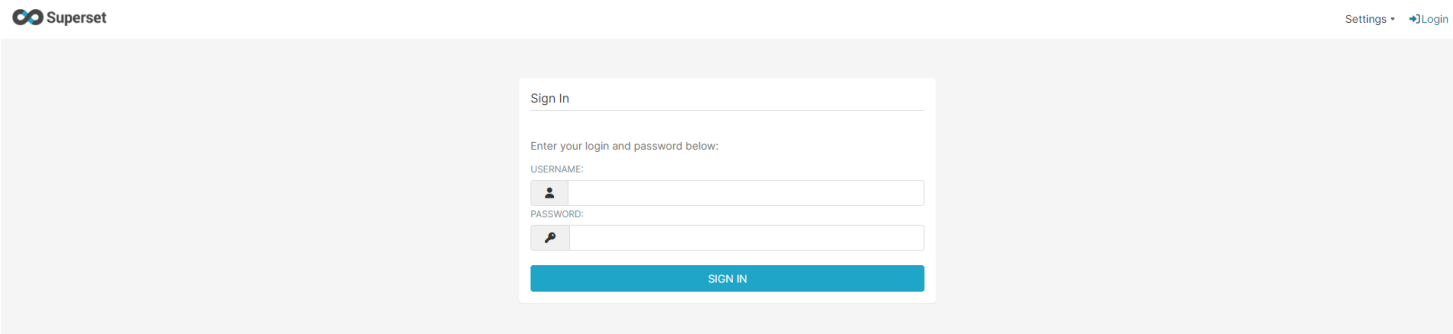
To make it easier we prepared an installation script that will pull the image and initialize it with required configuration as well as scripts for starting and running the Superset container. The script as well as additional information is available in the `superset/scripts/local-docker` folder, for more information about the installation process look at the `readme-docker.md` file.

To run the bash script in a unix environment by running in the containing folder type the following

```
./installation.sh
```

If you are using a Windows environment then you can instead use the other installation script `installation.ps1`

When the script finishes with executing you can open the application in your browser at http://localhost:8080 and the login screen should appear



If you didn't change anything in the installation script the user credentials should be the following ones:

- **username** -> admin
- **password** -> admin

In case you changed anything in the installation script regarding the following code snippet, the username and password will be the ones you have set here

```
# execute user creation in the 'superset' container to be able to login in the app
docker exec -it superset superset fab create-admin \
              --username admin \
              --firstname Superset \
              --lastname Admin \
              --email admin@superset.com \
              --password admin
```

You can also create an additional admin user and use that one instead, this is for you to choose, default should be ok since this is used on local computer

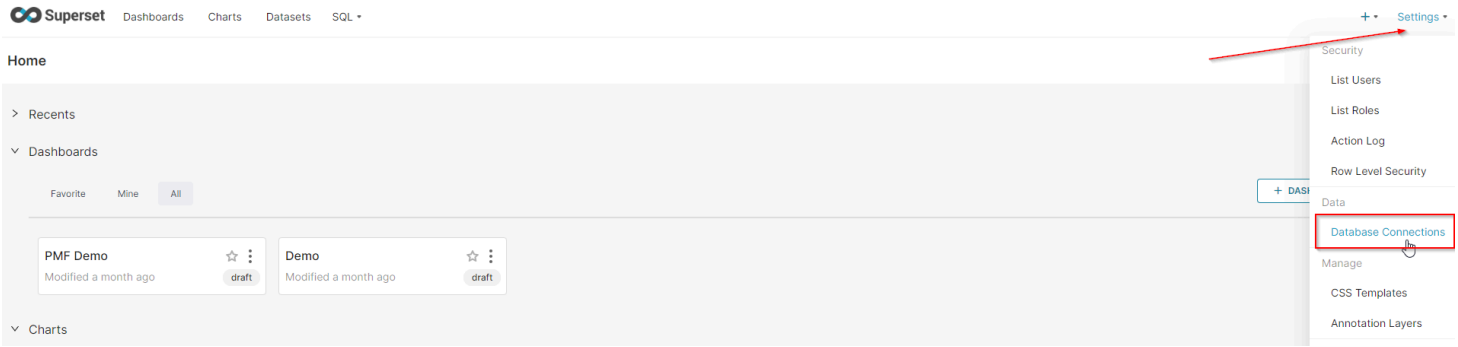
Working with dockerized Superset

> ⚠ **WARNING** All of the application configuration this deployment of Superset uses is stored in a database bundled in the docker image and when you remove or kill the container you work will be removed as well.
> To mitigate the problem of accidentally losing your work you can use the following approach after the initial installation of the docker image
> To start the application you can use the `start-superset.sh` script that is found under `superset/scripts/local-docker`
> To stop the application you can use the `stop-superset.sh` script that is found under `superset/scripts/local-dcoker`

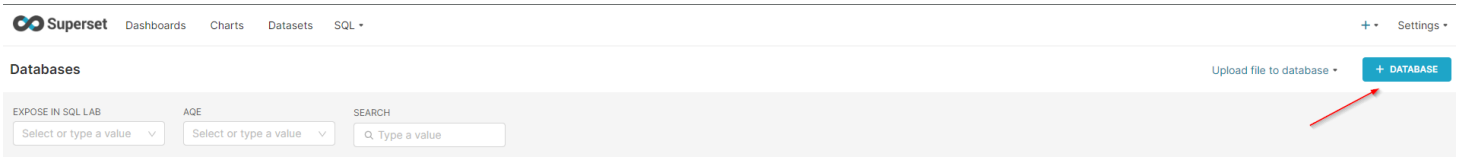
Add a new database

Database credentials are the same ones you received in the Database connection credentials

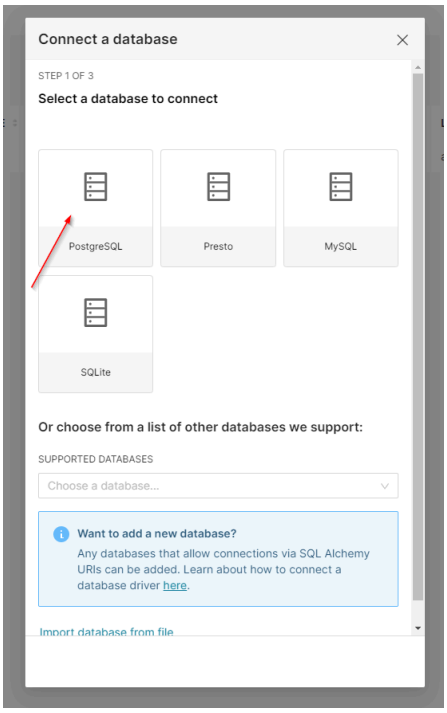
Before creating and running your first query we need to configure the database connection to the PostgreSQL database.



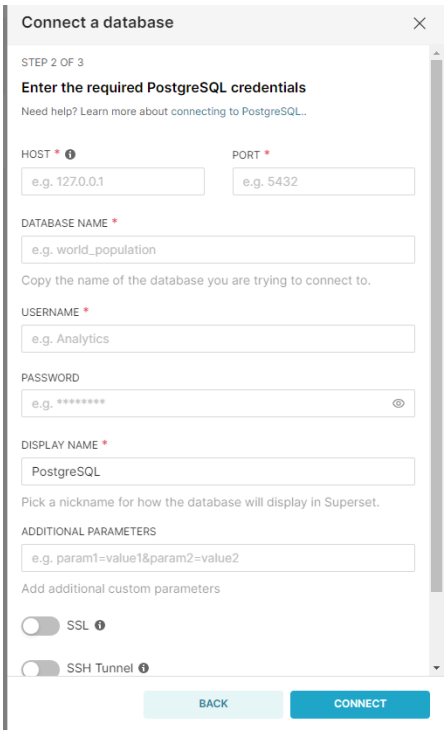
Click on the `Settings` menu and select the `Database Connections` item to open the page with the available connections.



Click on the `+Database` button to open the database connection setup dialog.



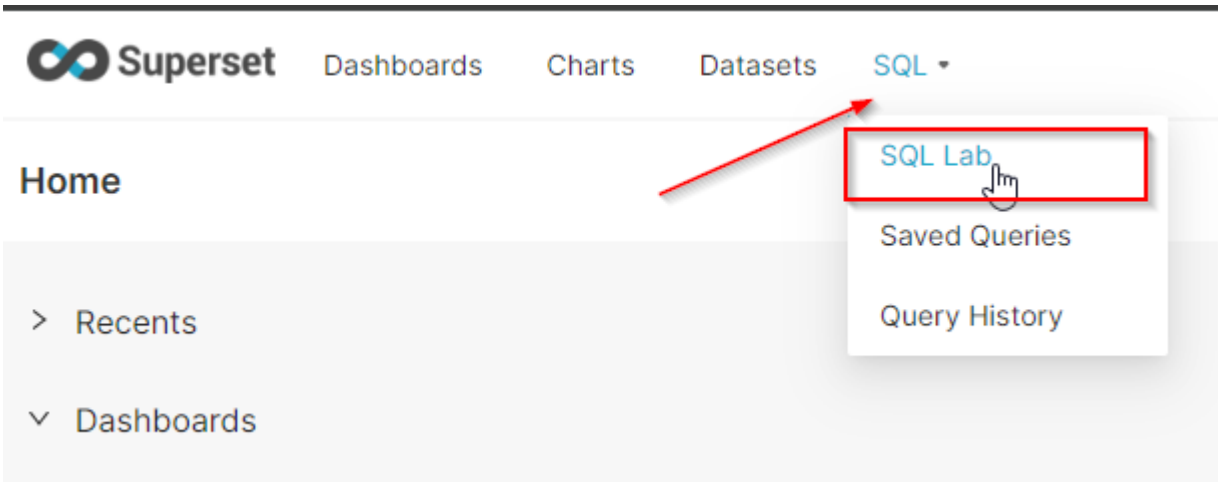
In the dialog window select the `PostgreSQL` database type.



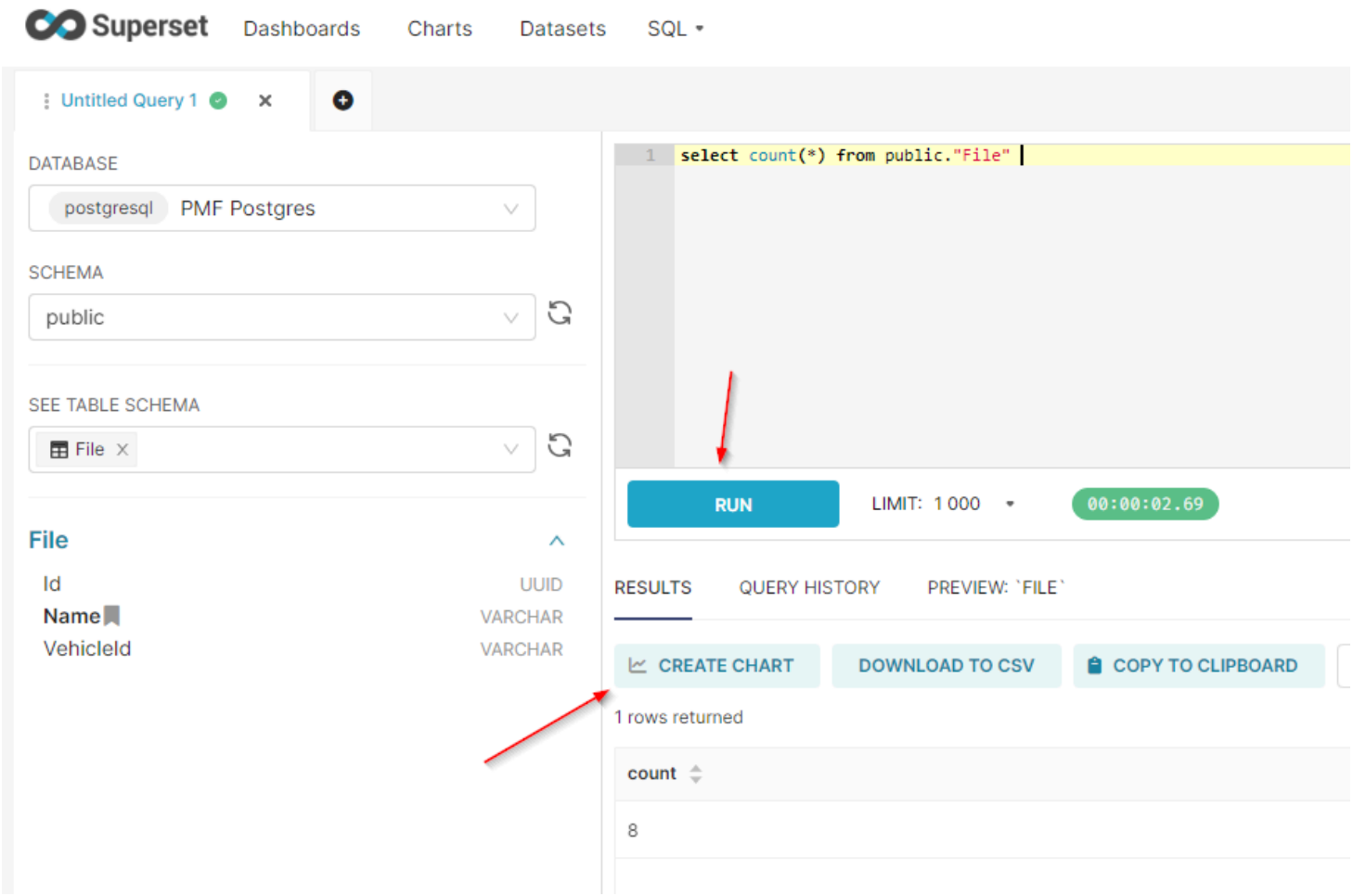
Clicking on `Connect` should test your connection to the database for your user, on the last step no modification is needed just confirm.

Create first query

You will be provided with a database schema to which you will connect with your user. You will only have access to this schema, in the examples we will use the `public` schema.



There are different views where you can write a query, but the suggested approach would be to go to the `SQL Lab` as in the example.



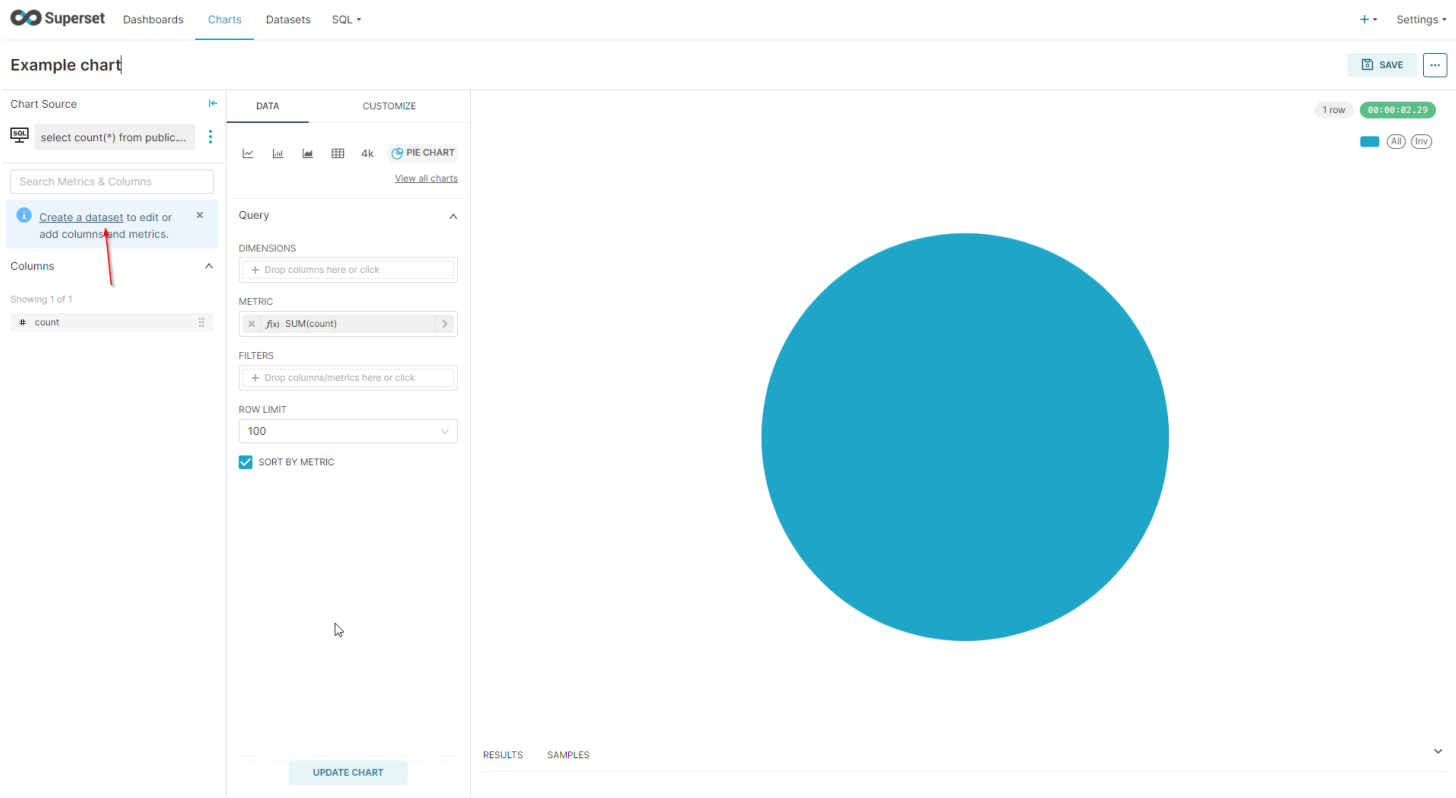
In `SQL Lab` on the left side is the `Database` and `schema` selector where you should select the database connection you have created earlier. The `See Table Schema` dropdown could prove useful if

you want to review the created tables additionally or just to use them as a reminder of what data they have.

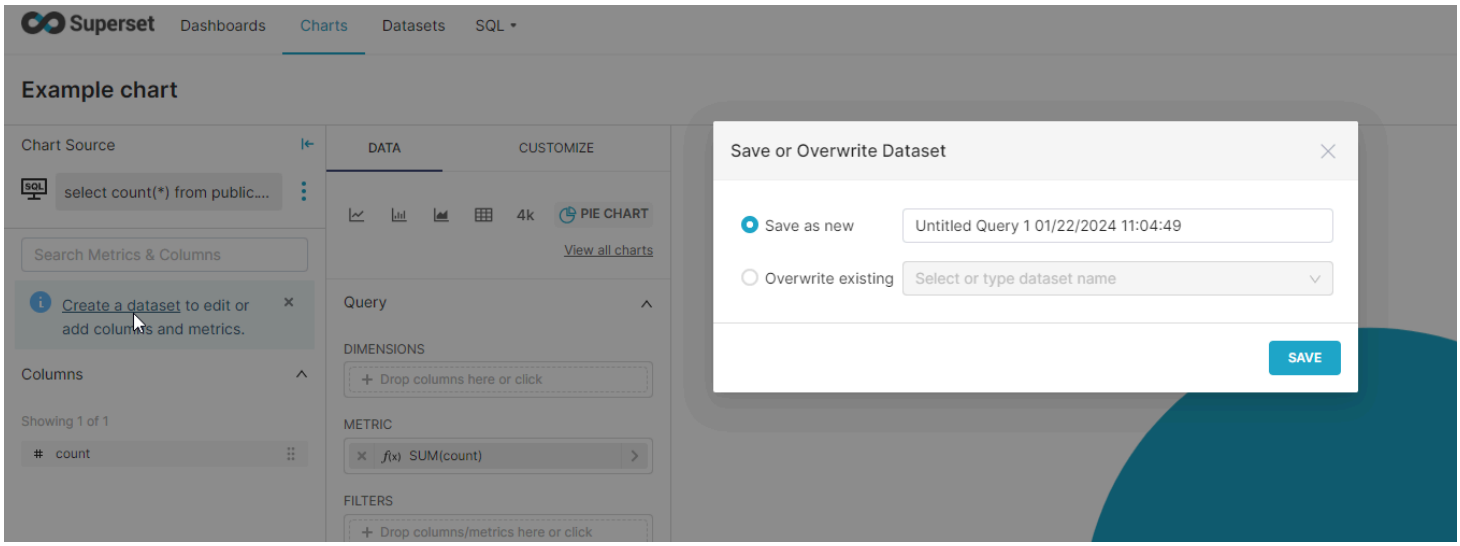
In the editor section you can write a test query using the `schema` you have been provided. Comparing the example your query would be

```
select count(*) from pmf_lab."File"
```

To run the query click the `Run` button and after the query passed you should your result in the table below. To create your first chart click on the `Create Chart` button that will open the following page.



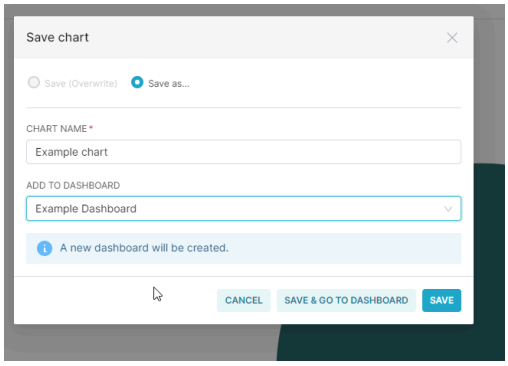
On this page one thing that we would recommend is to `Create a dataset`, so that you can edit the query afterwards if needed without creating the whole chart from stratch, clicking on that link the following dialog opens.



Name the dataset however you like as long as it makes sense, most likely the naming should follow the intention of the query.

There are a lot of customization options you can choose on the `Chart` page from chart type selection, to additional aggregations on the dataset.

To save a chart click on the top-right `Save` button which will open an additional dialog



Here you need to name the chart and select a `Dashboard` you want to save it. On the first creation the dashboard may not exist, so the application will create one for us.

Dashboards in Superset (Tasks)

Your task is to connect to the database using the instructions above and create several charts based on the data contained in the database, displaying them on a "dashboard".

Save and display the following charts:

1. Show as a pie plot event counts per event names
2. Show in a tabular way all events that have a duration longer than 40 seconds. Along the event information show the calculated event duration
    - Event duration is the time difference between when the event ended and started

- For time difference comparing use the INTERVAL data type e.g.

  `timestamp1 - timestamp2 > INTERVAL '3 hours'`

3. Show on a map visualization the number of events that happened per country by color coding that country
4. Show as a pie plot event counts per vehicle id
5. Show as a bar chart how many events are in which time bucket based on their duration. Group the events by event name. Time buckets to use:
   - `'< 30 s'`
   - `'30s - 120s`
   - `'> 120s'`
6. Show as a bar chart how many events have started in one hour buckets using the time_bucket function from TimescaleDB
   - to use the `time_bucket` function you need to prefix it with `public`, so it should be `public.time_bucket`
7. Show in a tabular way all events that have `aggregate ('SPEED', 'max')` in range of `[50,60]`
8. Show as a scatter plot values of `aggregate ('T_OIL', 'min')` only for events that have `aggregate2 ('SPEED', 'max')` in range of `[50, 60]`, additionally group the events by the hourly time bucket and by the event name
   - for the aggregation of values use the adequate aggregation
     - type = 'max' -> aggregate using 'max(a.Value)'
     - type = 'min' -> aggregate using 'min(a.Value)'
   - the x axis should be the time bucket and the y axis the aggregated aggregation
   - use event name to additionally categorize the data into multiple groups on the plot, e.g. each event name should be a different color
9. Show in a tabular way some statistics about how the event table data is stored as a hypertable under the hood. To do this use the chunks_detailed_size function from TimescaleDB and visualize only the `chunk_name` and `table_bytes` columns, the other are not important here. Since the `table_bytes` column isn't really human readable since it is in bytes use the pg_size_pretty function to show it in a more readable way. Order the chunks from the biggest to the smallest, use `table_bytes` for ordering.
   - to use the `chunks_detailed_size` function you need to prefix it with `public`, so it should be `public.chunks_detailed_size`