Data Network Dashboards

This document is currently under construction

2021-08-03

Contents

Chapter 1

Preface

Automated Characterization of Health Information at Large-scale Longitudinal Evidence Systems (ACHILLES) is a profiling tool developed by the OHDSI community to provide descriptive statistics of databases standardized to the OMOP Common Data Model. These characteristics are presented graphically in the ATLAS tool. However, this solution does not allow for database comparison across the data network. The Data Network Dashboards aggregates ACHILLES results files from databases in the network and displays the descriptive statistics through graphical dashboards. This tool is helpful to gain insight in the growth of the data network and is useful for the selection of databases for specific research questions. In the software demonstration we show a first version of this tool that will be further developed in EHDEN in close collaboration with all our stakeholders, including OHDSI.

Contributors

To develop this tool, EHDEN organized a hack-a-thon (Aveiro, December 2-3, 2019), where we defined and implemented a series of charts and dashboards containing the most relevant information about the OMOP CDM databases. The team involved in this task were composed by the following members:

- João Rafael Almeida[1]
- André Pedrosa[1]
- Peter R. Rijnbeek[2]
- Marcel de Wilde[2]
- Michel Van Speybroeck[3]
- Maxim Moinat[4]
- Pedro Freire[1]
- Alina Trifan[1]
- Sérgio Matos[1]
- José Luís Oliveira[1]

1 - Institute of Electronics and Informatics Engineering of Aveiro, Department of Electronics and Telecommunication, University of Aveiro, Aveiro, Portugal

2 - Erasmus MC, Rotterdam, Netherlands

3 - Janssen Pharmaceutica NV, Beerse, Belgium

4 - The Hyve, Utrecht, Netherlands

Considerations

This manual was written to be a guide for a clean installation of this system with all the dashboards that we defined during the project. The first chapter describes the goal of the system and the second how to install the system. The remaining chapters are dedicated to the dashboards, in which chapters describes one dashboard and all its charts. To simplify the representation of the dashboard's layout, we used similar schemas as it is presented in Figure 1.1. The white box is the dashboard and the inside boxes are charts. The colour changes in relation to the type of chart.

Figure 1.1: Example of a dashboards tool presenting the databases available in the network (simulated data)

License

Acknowledges

This work has been conducted in the context of EHDEN, a project that receives funding from the European Union's Horizon 2020 and EFPIA through IMI2 Joint Undertaking initiative, under grant agreement No 806968.

Chapter 2

Introduction

The OHDSI research network has been growing steadily which results in an increasing number of healthcare databases standardized to the OMOP CDM format. The OHDSI community created the ACHILLES tool (Automated Characterization of Health Information at Large-scale Longitudinal Exploration System) to characterize those databases. The results are available to the data custodian in their local ATLAS tool and helps them to gain insights in their data and helps in assessing the feasibility of a particular research questions.

ACHILLES was designed to extract the metadata from a single database, which by itself does not allow the comparison with the remaining databases in the network. However, we believe there is even more value in sharing this information with others to enable network research in a Data Network Dashboard.

Data Network Dashboard

The European Health Data and Evidence Network (EHDEN) project therefore designed a Data Network Dashboard tool, a web application to aggregate information from distributed OMOP CDM databases. It uses the ACHILLES results files to construct graphical dashboards and enables database comparison (Figure 2.1). The tool is built on Apache Superset, which is an open-source enterprise-ready business intelligence web application that can provide powerful and fully customizable graphical representations of data. Achilles results can be uploaded through the EHDEN Database Catalogue using the dashboards plugin but can also be directly uploaded in the tool. Figure 1. Example of a dashboards tool presenting age and gender distributions (simulated data).

In this tools, we defined and implemented a series of charts and dashboards containing the most relevant information about the databases, such as:

- **General**: dashboards that shows the databases types per country, the distribution of data source types, the growth of the Network including

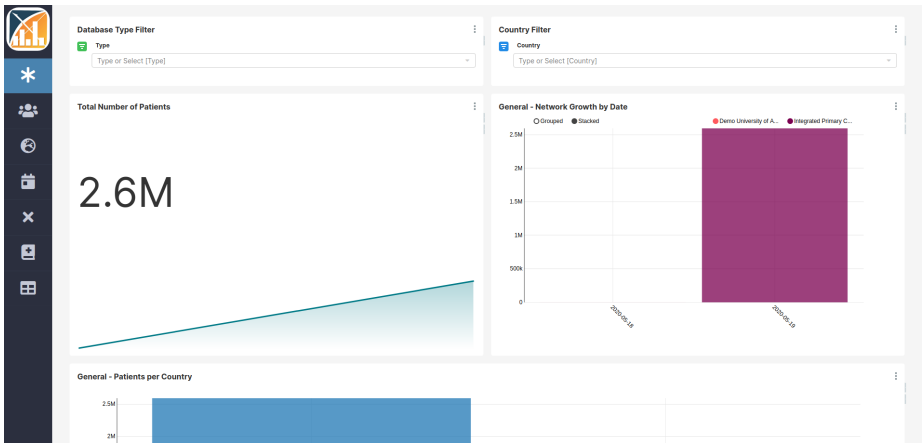


Figure 2.1: Example of a dashboards tool presenting the databases available in the network (simulated data)

the number of database and the number of patients in the databases over time;

- **Person**: representing the number of patients per country, age distribution at first observation, year of birth distribution and normalized gender distribution;
- **Population characteristics**: dashboard with the cumulative patient time, persons with continuous observation per month, and the start and end dates of those periods;
- **Visit**: chart to compare the number and type of visit occurrence records;
- **Death**: information about the number of death records by month, and the patient age at time of death;
- **Concepts**: bubble chart which shows the number of patients and records per concept over the databases;
- **Data domains**: heat map visualization of the major data domains in each database.

Chapter 3

Installation

Currently, we use docker to deploy our environment

First Steps

1. Clone the repository with the command `git clone --recurse-submodules https://github.com/EHDEN/NetworkDashboards`. If you already cloned the repository without the `--recurse-submodules` option, run `git submodule update --init` to fetch the superset submodule.
2. Create a `.env` file on the `docker` directory, using `.env-example` as a reference, setting all necessary environment variables (`SUPERSET\_MAPBOX\_API\_KEY` and `DASHBOARD\_VIEWER\_SECRET\_KEY`).

Dashboard Viewer setup

1. If you wish to expose the dashboard viewer app through a specific domain(s) you must add it/them to the `ALLOWED_HOSTS` list on file `dashboard_viewer/dashboard_viewer/settings.py` and remove the `'*'` entry.
2. Build containers' images: `docker-compose build`. This might take several minutes.
3. Set up the database and create an admin account for the dashboard viewer app: `docker-compose run --rm dashboard ./docker-init.sh`.

Insert Concepts

The concepts table is not in the repository due to its dimension, therefore we use directly the Postgres console to insert this table in the installation.

1. Get your concept csv file from Athena

2. Copy the file into postgres container

```
docker cp concept.csv dashboard_viewer_postgres_1:/tmp/
```

3. Enter in the postgres container:

```
docker exec -it dashboard_viewer_postgres_1 bash
```

4. Enter in the `achilles` database (value of the variable `POSTGRES_ACHILLES_DB` on the .env file) with the `root` user (value of the variable `POSTGRES_ROOT_USER` on the .env file):

```
psql achilles root
```

5. Create the `concept` table

```
CREATE TABLE concept (
  concept_id         INTEGER        NOT NULL,
  concept_name       VARCHAR(255)   NOT NULL,
  domain_id          VARCHAR(20)    NOT NULL,
  vocabulary_id      VARCHAR(20)    NOT NULL,
  concept_class_id   VARCHAR(20)    NOT NULL,
  standard_concept   VARCHAR(1)     NULL,
  concept_code       VARCHAR(50)    NOT NULL,
  valid_start_date   DATE           NOT NULL,
  valid_end_date     DATE           NOT NULL,
  invalid_reason     VARCHAR(1)     NULL
);
```

6. Copy the CSV file content to the table (this could take a while)

   To get both `'` (single quotes) and `"` (double quotes) on the `concept_name` column we use a workaround by setting the quote character to one that should never be in the text. Here we used `\b` (backslash).

```
COPY public.concept FROM '/tmp/concept.csv' WITH CSV HEADER
  DELIMITER E'\t' QUOTE E'\b';
```

7. Create index in table (this could take a while):

```
CREATE INDEX concept_concept_id_index ON concept (concept_id);
CREATE INDEX concept_concept_name_index ON concept (concept_name);
```

8. Set the owner of the `concept` table to the `achilles` user (value of the variable `POSTGRES_ACHILLES_USER` on the .env file):

```
ALTER TABLE concept OWNER TO achiller
```

9. Bring up the containers: `docker-compose up -d`.

10. Run the command `docker-compose run --rm dashboard python manage.py generate_materialized_views` to create the materialized

views on Postgres.

Superset setup

1. Make sure that the container `superset-init` has finished before continuing. It is creating the necessary tables on the database and creating permissions and roles.
2. Execute the script `./superset/one_time_run_scripts/superset-init.sh`. This will create an admin account and associate the `achilles` database to Superset. **Attention:** You must be in the docker directory to execute this script.
3. We have already built some dashboards so if you want to import them run the script `./superset/one_time_run_scripts/load_dashboards.sh`. **Attention:** You must be in the docker directory to execute this script.
4. If you used the default ports:
   - Go to `http://localhost` to access the dashboard viewer app.
   - Go to `http://localhost:8088` to access superset.
5. To any anonymous user view dashboards, add the following:
   - all datasource access on all_datasource_access
   - can csrf token on Superset
   - can dashboard on Superset
   - can explore json on Superset
   - can read on Chart
   - can read on CssTemplate

Dummy data

On a fresh installation, there are no achilles_results data so Superset's dashboards will display "No results". On the root of this repository, you can find the `demo` directory where we have an ACHILLES results file with synthetic data that you can upload to a data source on the uploader app of the dashboard viewer (http://localhost/uploader). If you wish to compare multiple data sources, on the `demo` directory there is also a python script that allows you to generate new ACHILLES results files, where it generates random count values based on the ranges of values for each set of analysis_id and stratums present on a base ACHILLES results file. So, from the one ACHILLES results fill we provided, you can have multiple data sources with different data.

Chapter 4

Processes

Data Sources

Target: platform user

Before uploading any data to this platform, a data source owner has to create a data source instance to then associated the upload data with.

The creation of data source is done through the `[BASE_URL]/uploader/` URL, where 7 fields are expected:

1. name: an extensive name
2. acronym: a short name
3. country: where is the data source localized
4. link (Optional): web page of the data source
5. database type: type of OMOP database
6. coordinates: a more accurate representation of the data source's localization
7. hash (Optional): the internal unique identifier of a data source

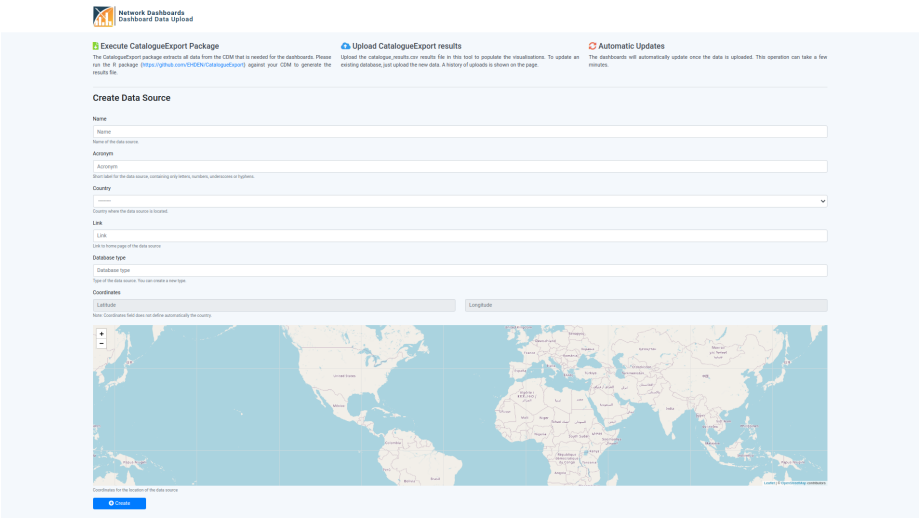
If you access `[BASE_URL]/uploader/` the 7th field (hash) is set automatically for something random, however, if you want to set it use the `[BASE_URL]/uploader/[HASH]/` URL.

To avoid duplication on the database type field, this field is transformed (use title case and trimmed) and then is checked there is already a record (Database Type) with the same value.

There are several ways to create a data source:

1. Create through a web form

By accessing the `[BASE_URL]/uploader/` URL, you will get a form where you can field the fields, where the country field is a dropdown and the coordinates field is set through a map widget.



2. Automatically create when performing a GET to the `[BASE_URL]/uploader/` URL

If the Network Dashboards platform is being used as a third-party application and the main application has all the data for the required fields, the data source can be automatically created and the user is redirected directly to the upload files page.

To perform this, each field should be provided as a URL parameter when accessing the `[BASE_URL]/uploader/` URL. If all required fields are provided and are valid the data source is created and the user is redirected to the upload files page. If a required field is missing or is not valid the webform is presented to the user so it can manually fill those fields.

3. Automatically create by performing a POST to the `[BASE_URL]/uploader/` URL

Since the creation URL does not have csrf cookie protection, you can perform a POST request as you were submitting a form.

Notes For the automatic options: -. Since the coordinates field is composed of two fields (latitude, longitude), it should be submitted as `coordinates_0=[latitude]` and `coordinates_1=[longitude]` -. The country field should match one of the available on the dropdown of the webform.

Catalogue Results Files

Target: platform user

Once a data source is created you can access its upload page by accessing the `[BASE_URL]/uploader/[HASH]/`. If no data source has the provided hash you

will be redirected back to the data source creation form.

On the upload page you can:

1. Go to the edit page of your data source
2. Upload a catalogue results file
3. Check the upload history

A catalogue results file is a CSV file, the result obtained after running the EHDEN/CatalogueExport R package on an OMOP database. It is a variant of the OHDSI/Achilles where it only extracts a subset of analyses of the ACHILLES' original set.

The upload form expects a CSV file with the following columns:

Name	Type	Required/Non-Nullable/Non-Empty
analysis_id	int	Yes
stratum_1	string	No
stratum_2	string	No
stratum_3	string	No
stratum_4	string	No
stratum_5	string	No
count_value	int	Yes
min_value	double	No
max_value	double	No
avg_value	double	No
stdev_value	double	No
median_value	double	No
p10_value	double	No
p25_value	double	No
p75_value	double	No
p90_value	double	No

The uploaded file must:

- either contain the first 7 columns OR all 16 columns
- contain the columns in the same order as presented in the table above

While parsing the uploaded file, some data is extracted to then present on the Upload history and to update data source information. This data is extracted from the record with analysis id 0, **which is required to be present on the file**, and 5000, which is optional. Next is presented the data extracted and their description:

- R Package Version: the version of CatalogueExport R package used
- Generation Date: date at which the CatalogueExport was executed on the OMOP database

- Source Release Date: date at which the OMOP database was released
- CDM Release Date: date at which the used CDM version was released
- CDM Version: version of the CDM used
- Vocabulary Version: version of the vocabulary used

The next table is presented where the previous data is stored on the rows with analysis id 0 and 5000:

Analysis Id	Stratum 1	Stratum 2	Stratum 3	Stratum 4	Stratum 5
0		R Package Version	Generation Date		
5000		Source Release Date	CDM Release Date	CDM Version	Vocabulary Ver

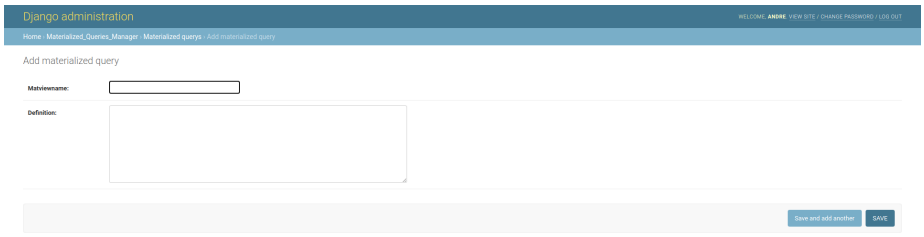
Materialized Views

Target: admin user

For each chart, Superset has an underlying SQL query which in our case is run every time a chart is rendered. If one of these queries takes too long to execute the charts will also take too long until they are rendered and eventually users might get timeout messages given a bad user experience.

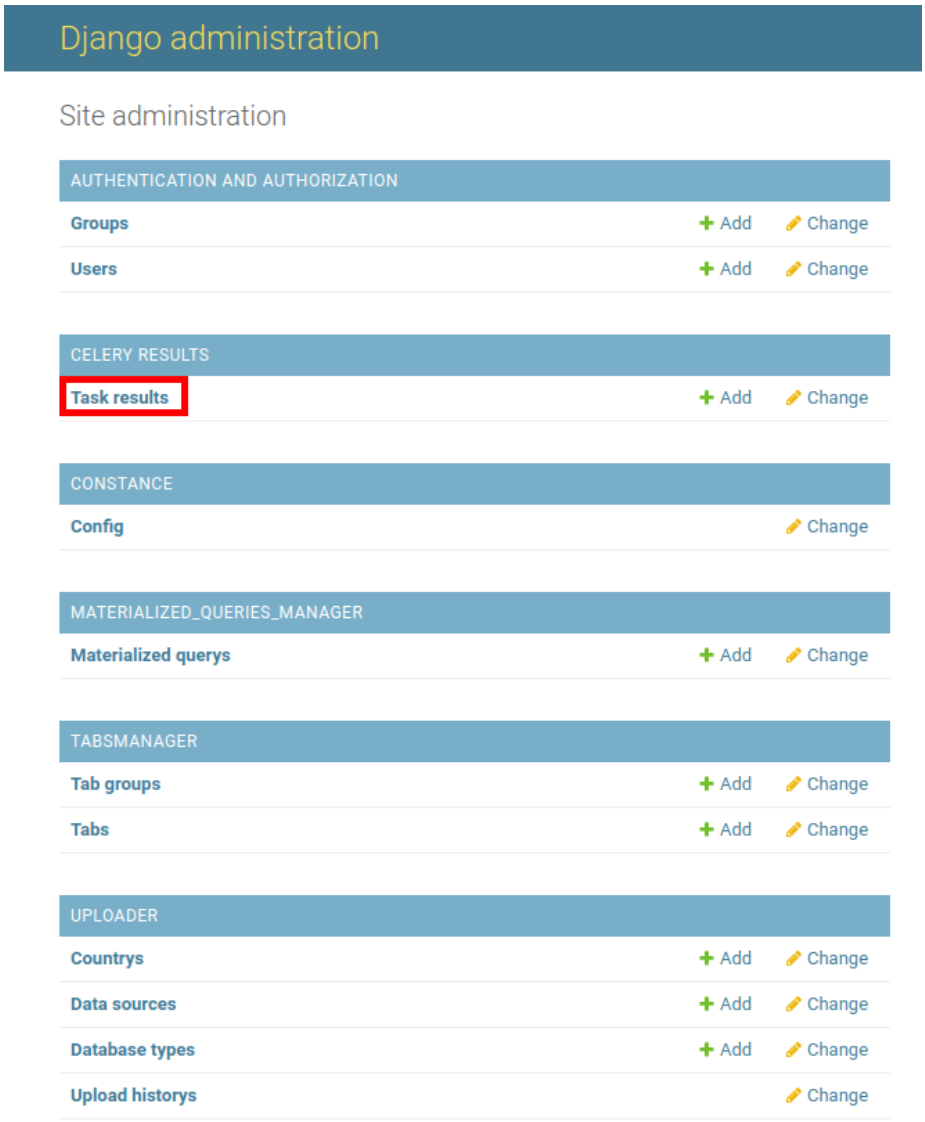
To avoid this problem, instead of executing the raw SQL query we create a postgres materialized view of the query, which is then used to feed the data to the chart. So only a simple `SELECT x FROM x` query is executed when a chart is rendered.

So whenever I create a chart I have to access the Postgres console? No, we created an unmanaged Materialized Queries model that maps to the materialized views on Postgres. With it you can create new materialized views through the Django admin app, by accessing the `[BASE_URL]/admin/` URL.

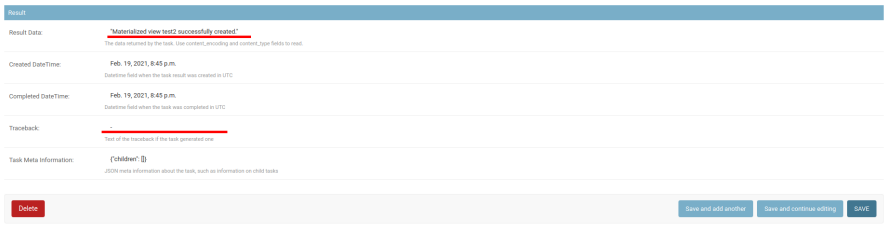


You have to provide the materialized view name and its query, which will then be used to execute the query `CREATE MATERIALIZED VIEW [name] AS [query]`, which will be executed on a background task so the browser doesn't hang and times out, in case of complicated queries. Taking this into account, the record associated will not appear on the Django admin app until the `CREATE MATERIALIZED VIEW` query finishes.

To give feedback on the background task we use celery/django-celery-results, so you can check the status of a task on the Task Results model of the Celery Results app



After the creation of a Materialized Query, the will be a message telling the id of the task which is executing the `CREATE MATERIALIZED VIEW` query. You can then check for the record associated with the task, click on the id to get more details. If something went wrong check the error message either on Result Data or Traceback fields under the Result section



After all this, the final step is to add the materialized view as a Dataset. Login into Superset, then go to Data -> Datasets and create a new one. Select the `Achilles` database, the `public` schema, then the created materialized view and click "ADD". After this, the materialized view can be used as a data source for a new chart.

Tabs View [Deprecated]

Target: admin user

Chapter 5

Backups

1. Create a credentials file (the structure of the file depends on the target cloud server)
2. Create a `.dashboards_backups.conf` file under your home directory (variable `$HOME`) using `dashboards_backups.conf.example` as base, setting the appropriate value for the several variables.

   For variables associated with files and directories always use *absolute* paths.

   Variables:

   - `RUN`: Set it to `0` if you don't want the next scheduled backup to run.

     This variable allows you to cancel any backup runs while you are doing some maintenance on the application.

   - `CONSTANCE_REDIS_DB`: Number of the Redis database where the django constance config is stored. The default value is 2. This value should be the same as the environment variable `REDIS_CONSTANCE_DB` of the dashboard container.
   - The following variables are associated with the arguemtns of the `backup_uploader` python package. Check its usage for more details:
     - `APP_NAME`: The backup process will generate some directories with this name in places that are shared with other applications.
     - `SERVER`: The name of the target cloud server to where backups should be uploaded (dropbox or mega).
     - `BACKUP_CHAIN_CONFIG`: Allows having different directories with backups of different ages.

   - `CREDENTIALS_FILE_PATH`: File containing the credentials to access the server to upload the backup file.

3. Install the `backup_uploader` python package by following its install instructions.

4. Schedule your backups

```
*    *    *   *    *  Command_to_execute
|    |    |    |   |
|    |    |    |    Day of the Week ( 0 - 6 ) ( Sunday = 0 )
|    |    |    |
|    |    |    Month ( 1 - 12 )
|    |    |
|    |    Day of Month ( 1 - 31 )
|    |
|    Hour ( 0 - 23 )
|
Min ( 0 - 59 )
```

   (Retrived from: Tutorialspoint)

   Ex: To run every day at 3:00 am

   1. `crontab -e`
   2. Add entry `0 3 * * * $HOME/NetworkDashboards/backups/backup.sh` (The path to the backup script might be different)

5.1 Restore

1. Select the compressed backup you want to restore and decompress it:

   `tar -xJf BACKUP_FILE.tar.xz`.

2. 1. **Redis**

      1. Make sure the redis docker container is down.
      2. (Re)place the file `dump.rdb` on the redis volume by the file `redis.rdb`. By default the redis volume is located where this repository was cloned on the directory `docker/volumes/redis`.
      3. Change its permissions, owner and group:

         ```
         chmod 0644 docker/volumes/redis/dump.rdb
         sudo chown -R 999:999 docker/volumes/redis
         ```

   2. **Postgres**

      1. Make sure all containers that make changes on the database are stopped.

2. Copy the file `postgres_backup.sql` into the postgres container

   `docker cp postgres.sql [CONTAINER_ID]:/tmp`.

3. Execute the backup script:

   `docker exec -u root dashboard_viewer_postgres_1 psql -f /tmp/postgres_backup.sql -U \$POSTGRES_USER -d \$POSTGRES_DB`.

3. **Media Files** If you have a volume pointing to where the media files are stored, replace all files with the ones present on the downloaded backup file. Else:

   1. Bring the dashoard container up `docker-compose up -d dashboard`
   2. Enter in the container `docker exec -it [CONTAINER_ID] bash`
   3. If you don't know where the media files are stored you can check the value of the MEDIA_ROOT variable
      1. `python manage.py shell`
      2. `from django.conf import settings`
      3. `print(settings.MEDIA_ROOT)`
   4. Remove the entire MEDIA_ROOT directory and exit the container
   5. Copy the media directory present on the backup file to the catalogue container `docker cp -a collected-media [CONTAINER_ID]:[MEDIA_ROOT_PARENT_PATH]`

Chapter 6

Dashboards

6.1 General

CSS

To hide the dashboard header insert the following css code to the `CSS` field on the edit page:

```
.dashboard > div:not(.dashboard-content) {  /* dashboard header */
  display: none;
}
```

With this every time you want to edit the dashboard layout you have to either comment the CSS inserted or remove it so the "Edit Dashboard" button can show again.

Database Type and Country Filter

Theses filter were designed to be used in the dashboard aiming the filtering of the data based on the field ''database_type" and "country" from the table ''data_source".

For the filters to work the name of the fields to filter should match in all tables used on the charts of this dashboard.

SQL query

```
SELECT source.name,
       country.country,
       source.database_type,
       source.acronym
```

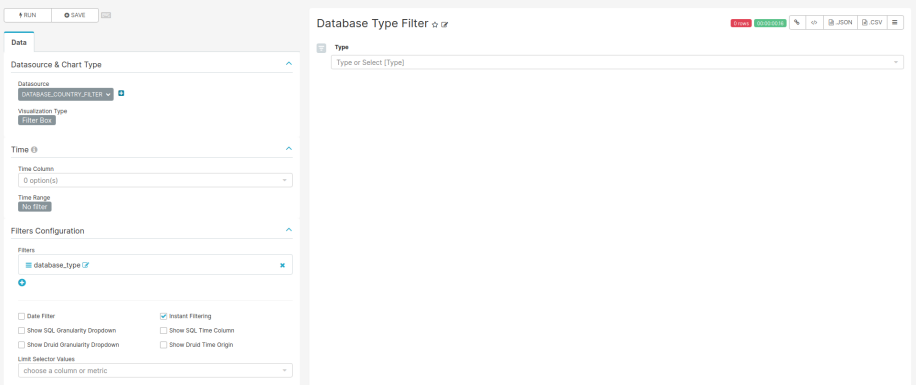


Figure 6.1: Settings for creating filters charts

```
FROM public.data_source AS source
INNER JOIN public.country AS country ON source.country_id=country.id
```

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Filter Box
  - Time
    - Time range: No filter
  - Filters Configuration
    - Filters:
      - database_type or country
    - Date Filter: off
    - Instant Filtering: on

Total Number of Patients

SQL query

```
SELECT
 country,
 database_type,
 release_date,
 SUM(count_value) OVER (ORDER BY release_date ASC)
FROM achilles_results
JOIN data_source ON data_source_id = data_source.id
JOIN country ON data_source.country_id = country.id
WHERE analysis_id = 1
```

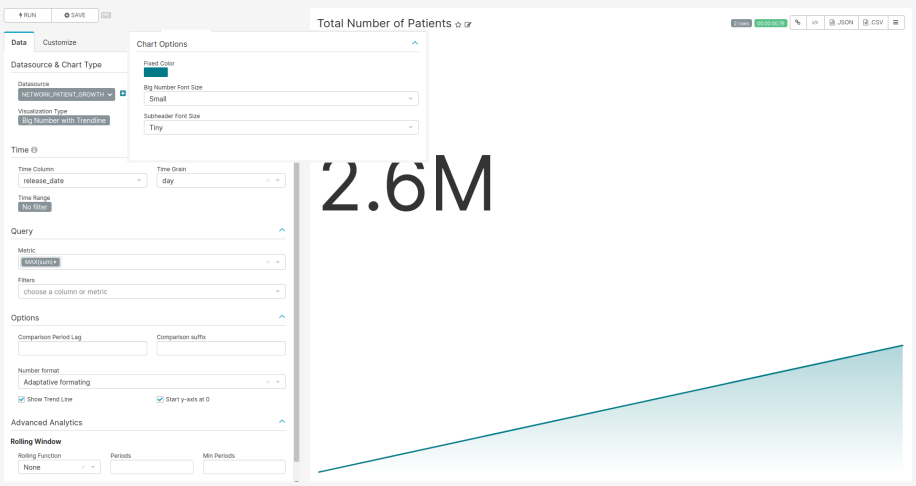


Figure 6.2: Settings for creating the Total Number of Patients chart

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Big Number with Trendline
  - Time
    - Time range: No filter
  - Query
    - Metrics: MAX(sum)
    - Series: release_date
    - Breakdowns: source
- Customize Tab
  - Chart Options
    - Big Number Font Size: Small
    - Subheader Font Size: Tiny

Network Growth by Date

SQL query

```
SELECT  source.name AS source,
        country.country,
        source.database_type,
        source.release_date,
        concepts.concept_name AS gender,
        achilles.count_value as count
FROM public.achilles_results AS achilles
```

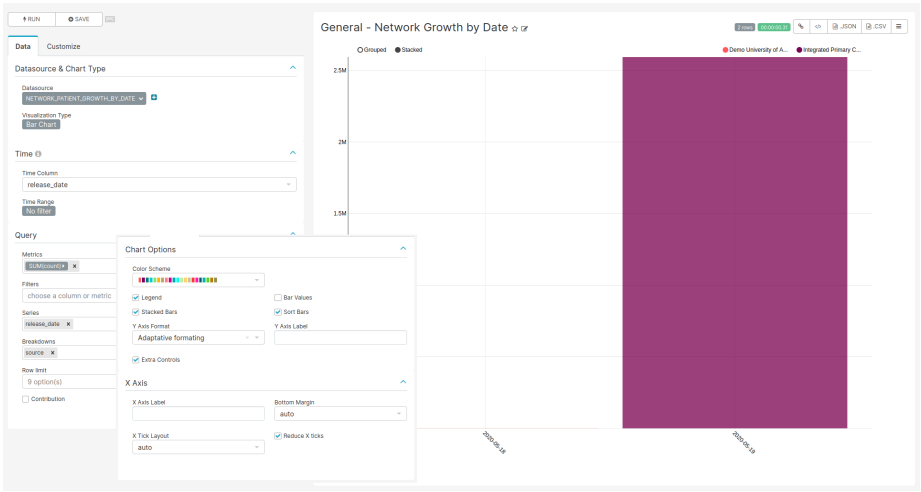


Figure 6.3: Settings for creating the Network Growth by Date chart

```
INNER JOIN public.data_source AS source ON achilles.data_source_id=source.id
INNER JOIN public.country AS country ON source.country_id=country.id
JOIN (
  SELECT '8507' AS concept_id, 'Male' AS concept_name
  UNION
  SELECT '8532', 'Female'
) AS concepts ON achilles.stratum_1 = concept_id
WHERE analysis_id = 2
```

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Bar Chart
  - Time
    - Time range: No filter
  - Query
    - Metrics: SUM(count_value)
    - Series: release_date
    - Breakdowns: source
- Customize Tab
  - Chart Options
    - Stacked Bars: on
    - Sort Bars: on
    - Extra Controls: on
  - X Axis

    * Reduce X ticks: on

Patients per Country

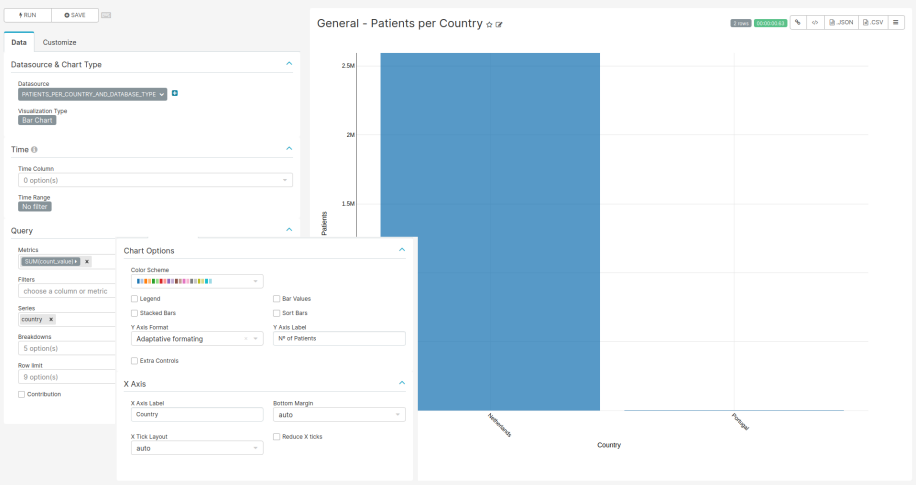


Figure 6.4: Settings for creating the Patients per Country chart

SQL query {#patientsPerCountryQuery}

```
SELECT country.country,
       source.database_type,
       count_value
FROM public.achilles_results AS achilles
INNER JOIN public.data_source AS source ON achilles.data_source_id=source.id
INNER JOIN public.country AS country ON source.country_id=country.id
WHERE analysis_id = 1
```

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Bar Chart
  - Time
    - Time range: No filter
  - Query
    - Metrics: SUM(count_value)
    - Series: country
- Customize Tab
  - Chart Options
    - Legend: off

    - Y Axis Label: Nº of Patients
  - X Axis
    - X Axis Label: Country

Database Types per Country

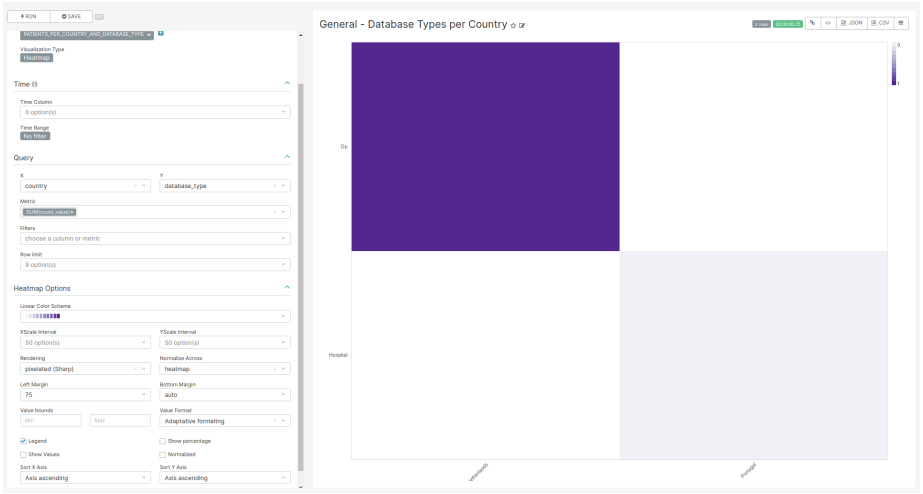


Figure 6.5: Settings for creating the Database Type per Country chart

SQL query

Same as Patients per Country query

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Heatmap
  - Time
    - Time range: No filter
  - Query
    - X: country
    - Y: database_type
    - Metric: SUM(countr_value)
  - Heatmap Options
    - Left Margin: 75
    - Show Percentage: off

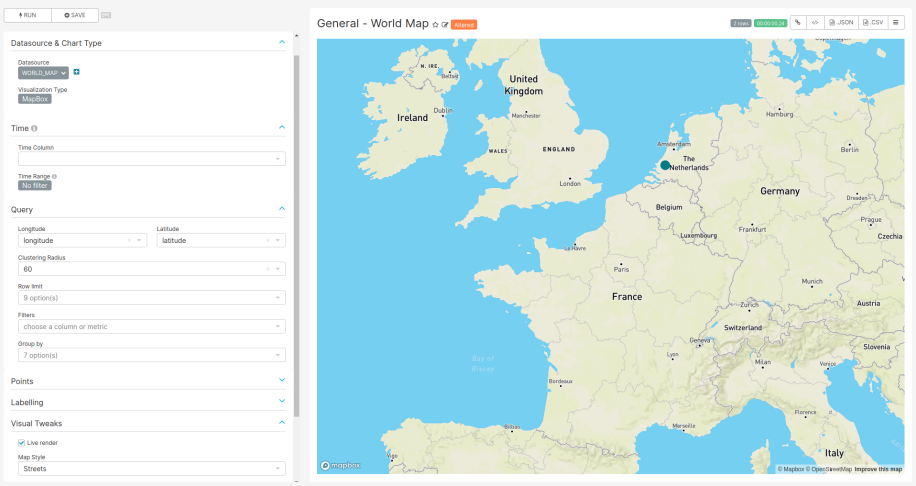


Figure 6.6: Settings for creating the World Map chart

World Map

SQL query

```
SELECT  name,
        acronym,
        database_type,
        latitude,
        longitude,
        country
FROM public.data_source AS source
INNER JOIN public.country AS country ON source.country_id=country.id
```

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: MapBox
  - Time
    - Time range: No filter
  - Query
    - Longitude: longitude
    - Latitude: latitude
  - Visual Tweaks
    - Map Style: Streets or Light or Outdoors

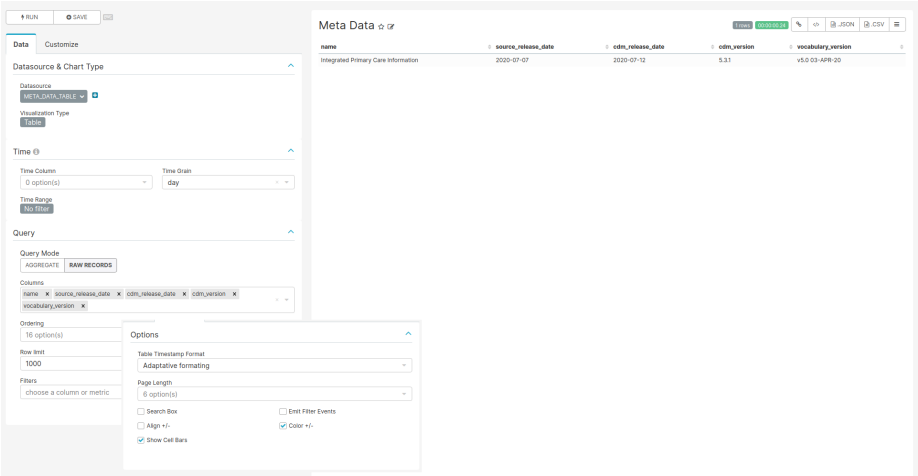


Figure 6.7: Settings for creating the Meta Data chart

Meta Data

SQL query

```
SELECT
  acronym,
  stratum_1 as "name",
  database_type,
  country,
  stratum_2 as "source_release_date",
  stratum_3 as "cdm_release_date",
  stratum_4 as "cdm_version",
  stratum_5 as "vocabulary_version"
FROM achilles_results
JOIN data_source ON achilles_results.data_source_id = data_source.id
JOIN country ON data_source.country_id = country.id
WHERE analysis_id=5000
```

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Table
  - Time
    - Time range: No filter
  - Query
    - Query Mode: Raw Records

* Columns: name, source_release_date, cdm_release_date, cdm_version, vocabulary_version

6.2 Person

Label Colors

In order to obtain the colors blue and rose in the chart representing the gender distribution, add the following JSON entry to the JSON object of the `JSON Metadata` field on the edit dashboard page:

```
"label_colors": {
    "Male": "#3366FF",
    "Female": "#FF3399"
}
```

CSS

To hide the dashboard header insert the following css code to the `CSS` field on the edit page:

```
.dashboard > div:not(.dashboard-content) {  /* dashboard header */
  display: none;
}
```

With this every time you want to edit the dashboard layout you have to either comment the CSS inserted or remove it so the "Edit Dashboard" button can show again.

Data Source Filter

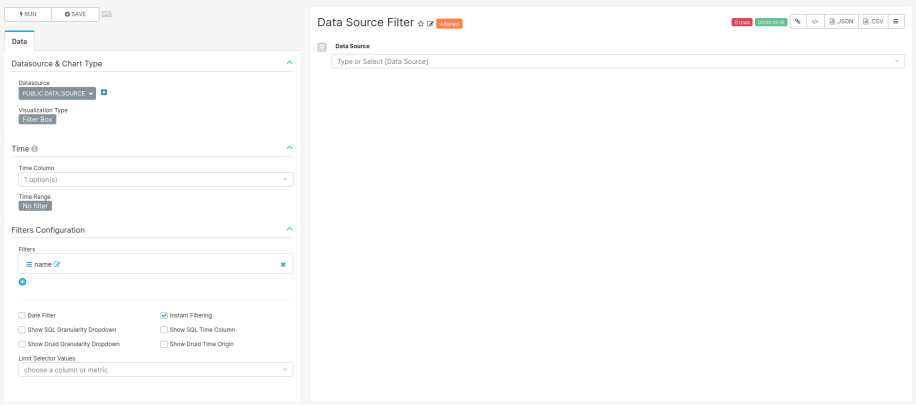


Figure 6.8: Settings for creating the Data Source filter chart

For the filter to work the name of the fields to filter should match in all tables used on the charts of this dashboard.

SQL query

No SQL query, use the sql table `data_source` of the `achilles` database.

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Filter Box
  - Time
    - Time range: No filter
  - Filters Configuration
    - Filters:
      - name
    - Date Filter: off
    - Instant Filtering: on

Age at first observation - Table {#age1ObservationTable}

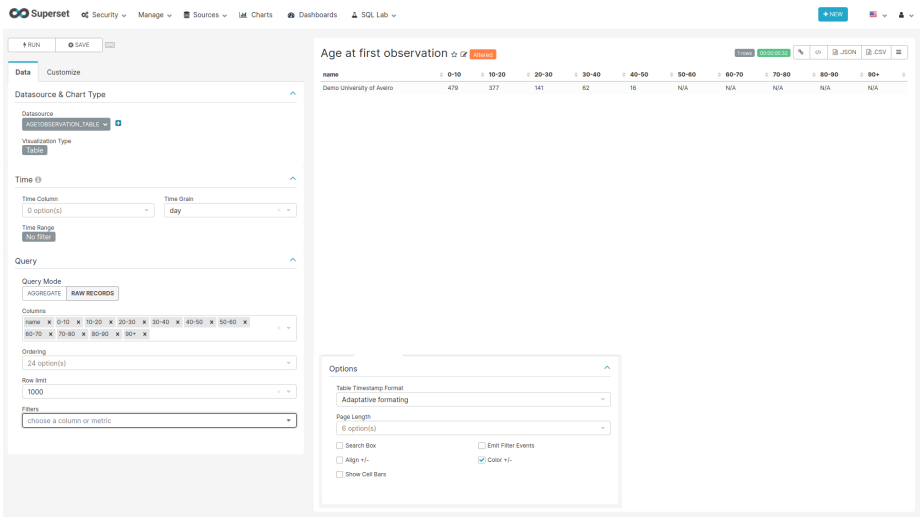


Figure 6.9: Settings for creating the Age at First Observation Table chart

SQL query

```
SELECT source.name,
       source.acronym,
```

```
       SUM(CASE WHEN CAST(stratum_2 AS INTEGER) < 10 THEN count_value END) AS "0-10",
       SUM(CASE WHEN CAST(stratum_2 AS INTEGER) >= 10 AND CAST(stratum_2 AS INTEGER) < 20 THEN co
       SUM(CASE WHEN CAST(stratum_2 AS INTEGER) >= 20 AND CAST(stratum_2 AS INTEGER) < 30 THEN co
       SUM(CASE WHEN CAST(stratum_2 AS INTEGER) >= 30 AND CAST(stratum_2 AS INTEGER) < 40 THEN co
       SUM(CASE WHEN CAST(stratum_2 AS INTEGER) >= 40 AND CAST(stratum_2 AS INTEGER) < 50 THEN co
       SUM(CASE WHEN CAST(stratum_2 AS INTEGER) >= 50 AND CAST(stratum_2 AS INTEGER) < 60 THEN co
       SUM(CASE WHEN CAST(stratum_2 AS INTEGER) >= 60 AND CAST(stratum_2 AS INTEGER) < 70 THEN co
       SUM(CASE WHEN CAST(stratum_2 AS INTEGER) >= 70 AND CAST(stratum_2 AS INTEGER) < 80 THEN co
       SUM(CASE WHEN CAST(stratum_2 AS INTEGER) >= 80 AND CAST(stratum_2 AS INTEGER) < 90 THEN co
       SUM(CASE WHEN CAST(stratum_2 AS INTEGER) >= 90 THEN count_value END) AS "90+"
FROM public.achilles_results AS achilles
INNER JOIN public.data_source AS source ON achilles.data_source_id=source.id
INNER JOIN public.concept ON CAST(stratum_1 AS BIGINT) = concept_id
WHERE analysis_id = 102
GROUP BY name, acronym
```

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Table
  - Time
    - Time range: No filter
  - Query
    - Query Mode: Raw Records
    - Columns: name, 0-10, 10-20, 20-30, 30-40, 40-50, 50-60, 60-70, 70-80, 80-90, 90+
- Customize Tab
  - Options
    - Show Cell Bars: off

Age at first observation - Bars {#age1ObservationBars}

SQL query

```
SELECT source.name,
    cast(stratum_1 AS int) AS Age,
    count_value AS count,
    source.acronym
FROM public.achilles_results AS achilles
INNER JOIN public.data_source AS source ON achilles.data_source_id=source.id
WHERE analysis_id = 101
```

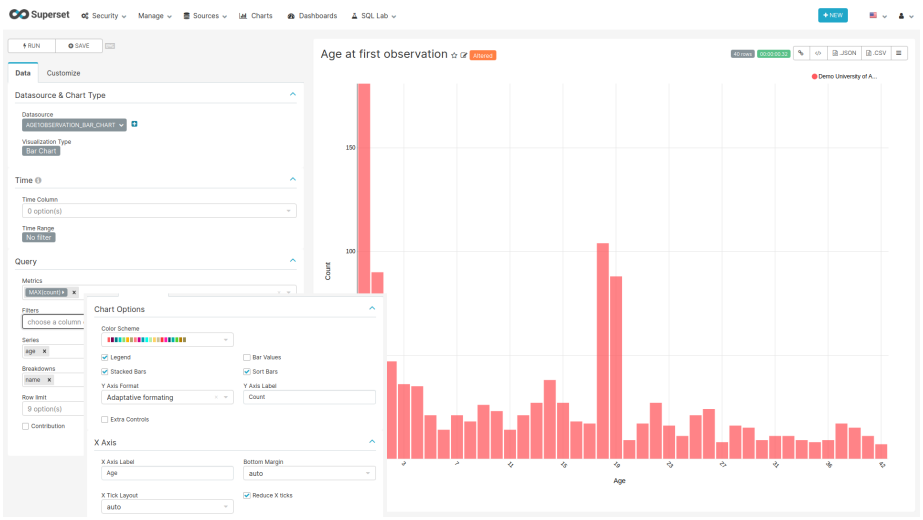


Figure 6.10: Settings for creating the Age at First Observation Bar chart

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Bar Chart
  - Time
    - Time range: No filter
  - Query
    - Metrics: MAX(count)
    - Series: age
    - Breakdowns: name
- Customize Tab
  - Chart Options
    - Stacked Bars: on
    - Sort Bars: on
    - Y Axis Label: Count
  - X Axis
    - X Axis Label: Age
    - Reduce X ticks: on

Year of Birth {#yearOfBirth}

SQL query

```
SELECT source.name,
       source.acronym,
```

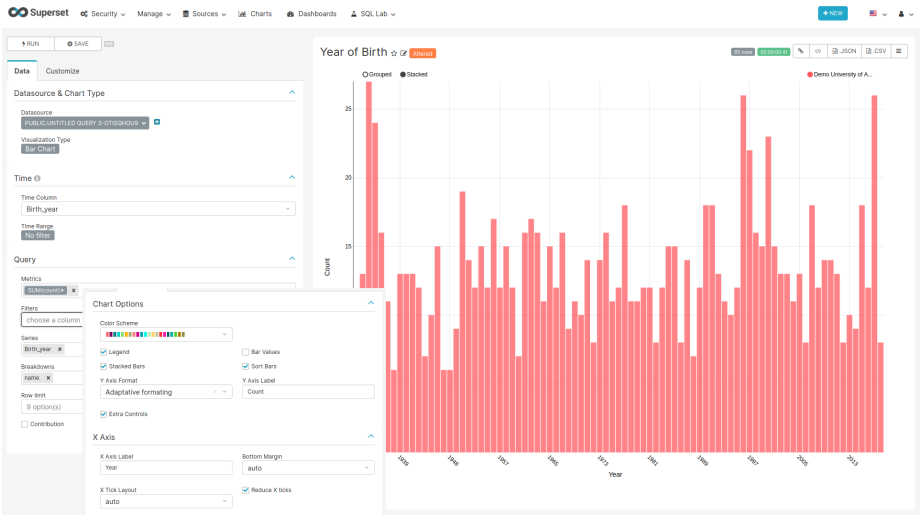


Figure 6.11: Settings for creating the Year of Birth chart

```
       stratum_1 AS "Birth_year",
       count_value AS count
FROM public.achilles_results AS achilles
INNER JOIN public.data_source AS source ON achilles.data_source_id=source.id
WHERE analysis_id = 3
```

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Bar Chart
  - Time
    - Time range: No filter
  - Query
    - Metrics: SUM(count)
    - Series: Birth_year
    - Breakdowns: name
- Customize Tab
  - Chart Options
    - Stacked Bars: on
    - Sort Bars: on
    - Y Axis Label: Count
    - Extra Controls: on
  - X Axis
    - X Axis Label: Year

* Reduce X ticks: on

Gender

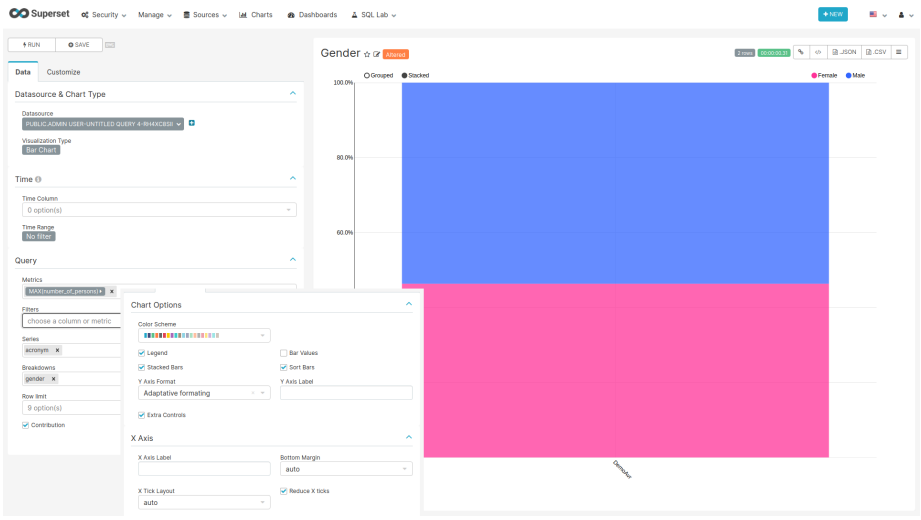


Figure 6.12: Settings for creating the Gender chart

SQL query

```
SELECT source.name,
       concept_name AS Gender,
       count_value AS Number_of_persons,
       source.acronym
FROM public.achilles_results AS achilles
INNER JOIN public.data_source AS source ON achilles.data_source_id=source.id
JOIN (
  SELECT '8507' AS concept_id, 'Male' AS concept_name
  UNION
  SELECT '8532' AS concept_id, 'Female' AS concept_name
) AS concepts ON achilles.stratum_1 = concept_id
WHERE analysis_id = 2
```

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Bar Chart
  - Time
    - Time range: No filter

    - Query
        - Metrics: MAX(Number_of_persons)
        - Series: acronym
        - Breakdowns: gender
        - Contribution: on
- Customize Tab
    - Chart Options
        - Stacked Bars: on
        - Sort Bars: on
        - Extra Controls: on
    - X Axis
        - Reduce X ticks: on

6.3 Observation Period

CSS

To hide the dashboard header insert the following css code to the `CSS` field on the edit page:

```
.dashboard > div:not(.dashboard-content) {  /* dashboard header */
  display: none;
}
```

With this every time you want to edit the dashboard layout you have to either comment the CSS inserted or remove it so the “Edit Dashboard” button can show again.

Data Source Filter

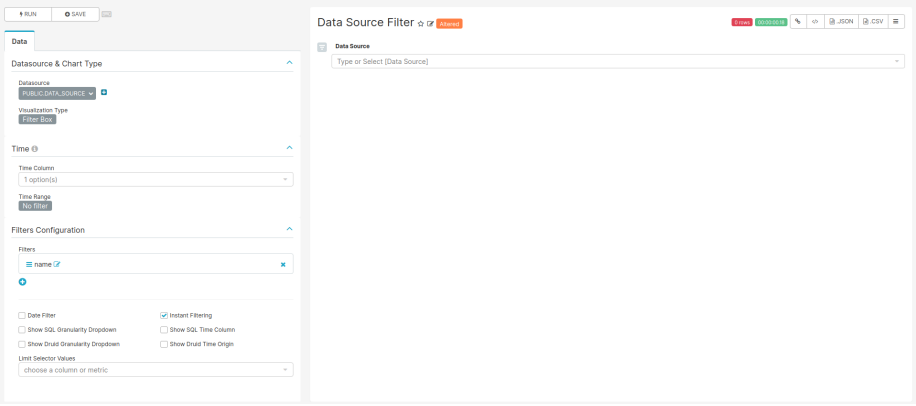


Figure 6.13: Settings for creating the Data Source filter chart

For the filter to work the name of the fields to filter should match in all tables used on the charts of this dashboard.

SQL query

No SQL query, use the sql table `data_source` of the `achilles` database.

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Filter Box
  - Time
    - Time range: No filter
  - Filters Configuration
    - Filters:
      - name
    - Date Filter: off
    - Instant Filtering: on

Number of Patients in Observation Period {#numInObservationPeriod}

The Number of Patients in Observation Period plot shows the number of patients that contribute at least one day in a specific month.

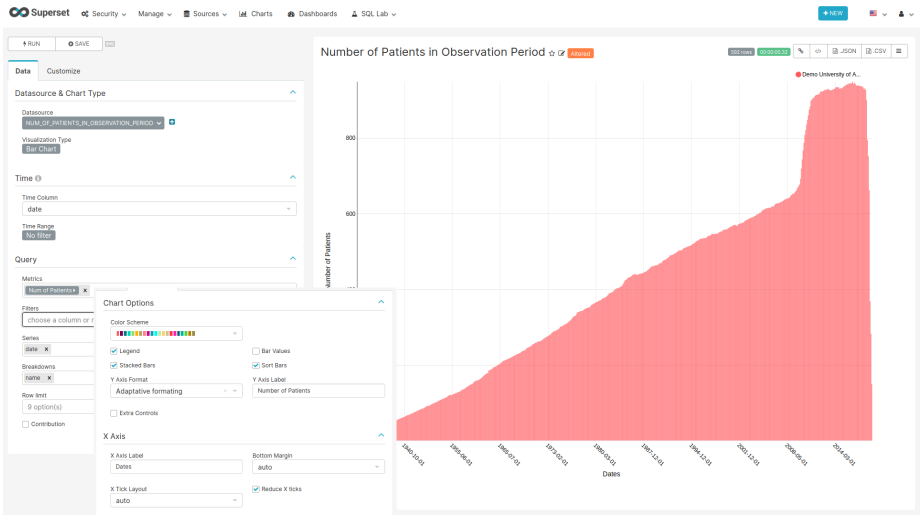


Figure 6.14: Settings for creating the Number of Patients in Observation Period chart

SQL query

```
SELECT source.name,
       source.acronym,
       to_date(stratum_1, 'YYYYMM') as Date,
       count_value
FROM public.achilles_results AS achilles
INNER JOIN public.data_source AS source ON achilles.data_source_id=source.id
WHERE analysis_id = 110
```

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Bar Chart
  - Time
    - Time range: No filter
  - Query
    - Metrics: MAX(count_value) with label “Num of Patients”
    - Series: date
    - Breakdowns: name
- Customize Tab
  - Chart Options
    - Stacked Bars: on
    - Sort Bars: on
    - Y Axis Label: Number of Patients
  - X Axis
    - X Axis Label: Dates
    - Reduce X ticks: on

Observation Period Start Dates

SQL query

```
SELECT source.name,
       source.acronym,
       to_date(stratum_1, 'YYYYMM') AS year_month,
       count_value
FROM public.achilles_results AS achilles
INNER JOIN public.data_source AS source ON achilles.data_source_id=source.id
WHERE analysis_id = 111
```

Chart settings

- Data Tab
  - Datasource & Chart Type

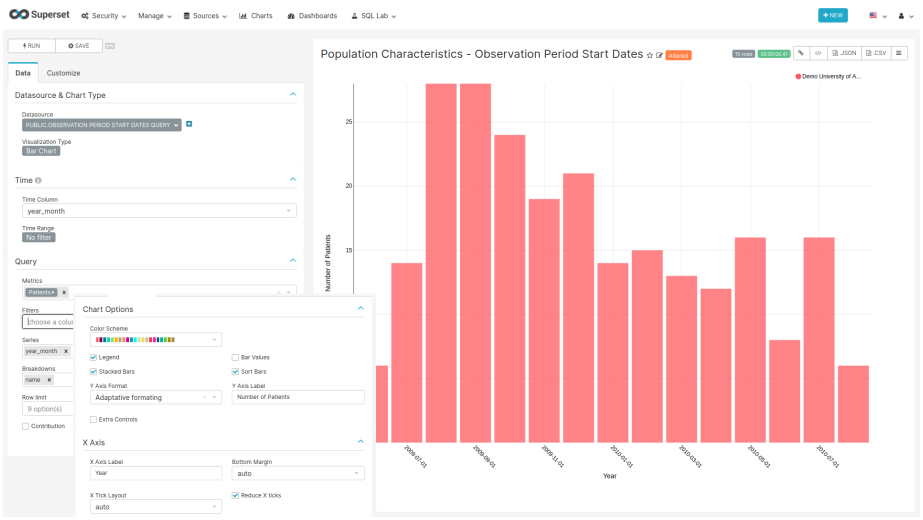


Figure 6.15: Settings for creating the Observation Period Start Dates chart

        * Visualization Type: Bar Chart
    - Time
        * Time range: No filter
    - Query
        * Metrics: SUM(count_value) with label "Patients"
        * Series: year_month
        * Breakdowns: name
- Customize Tab
    - Chart Options
        * Stacked Bars: on
        * Sort Bars: on
        * Y Axis Label: Number of Patients
    - X Axis
        * X Axis Label: Year
        * Reduce X ticks: on

Observation Period End Dates

SQL query

```
SELECT source.name,
       source.acronym,
       to_date(stratum_1, 'YYYYMM') AS year_month,
       count_value
FROM public.achilles_results AS achilles
INNER JOIN public.data_source AS source ON achilles.data_source_id=source.id
```

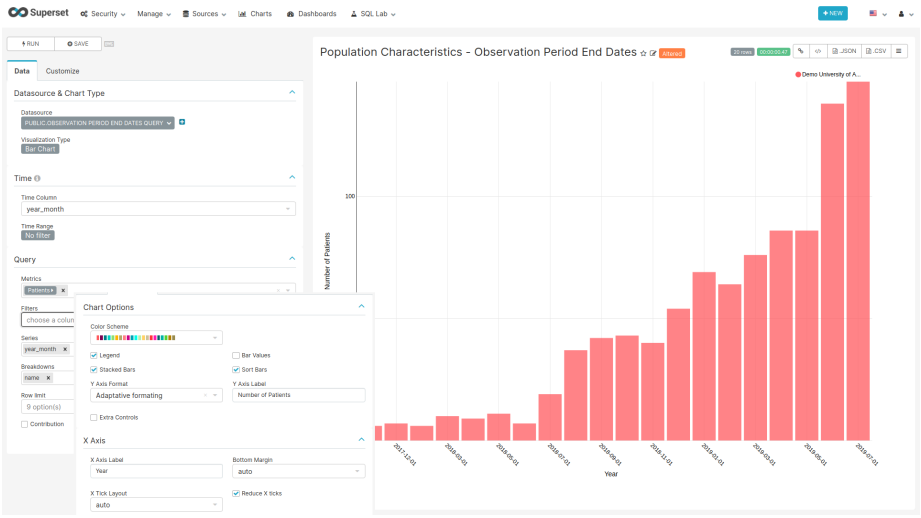


Figure 6.16: Settings for creating the Observation Period End Dates chart

```
WHERE analysis_id = 112
```

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Bar Chart
  - Time
    - Time range: No filter
  - Query
    - Metrics: SUM(count_value) with label “Patients”
    - Series: year_month
    - Breakdowns: name
- Customize Tab
  - Chart Options
    - Stacked Bars: on
    - Sort Bars: on
    - Y Axis Label: Number of Patients
  - X Axis
    - X Axis Label: Year
    - Reduce X ticks: on

6.4 Visit

This dashboard shows the different types of visits per data source (see Visit Occurence Table)

CSS

To hide the dashboard header insert the following css code to the `CSS` field on the edit page:

```
.dashboard > div:not(.dashboard-content) {  /* dashboard header */
  display: none;
}
```

With this every time you want to edit the dashboard layout you have to either comment the CSS inserted or remove it so the "Edit Dashboard" button can show again.

Data Source Filter

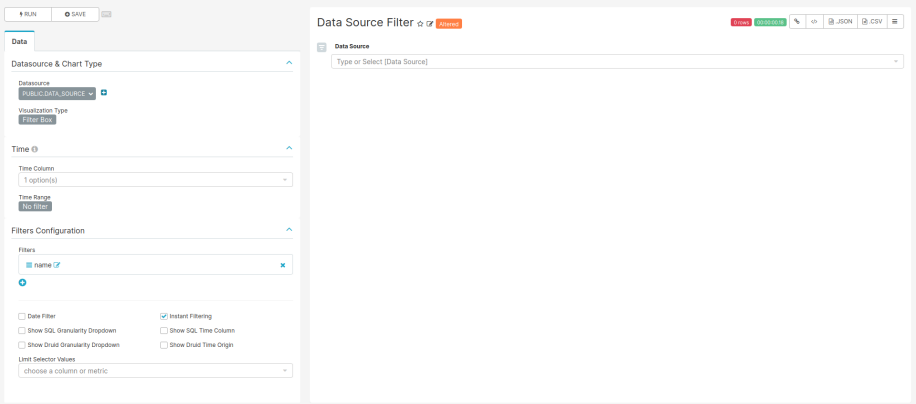


Figure 6.17: Settings for creating the Data Source filter chart

For the filter to work the name of the fields to filter should match in all tables used on the charts of this dashboard.

SQL query

No SQL query, use the sql table `data_source` of the `achilles` database.

Chart settings

- Data Tab
  - Datasource & Chart Type

    - Visualization Type: Filter Box
  - Time
    - Time range: No filter
  - Filters Configuration
    - Filters:
      - name
    - Date Filter: off
    - Instant Filtering: on

Visit Type Table {#visitTypeTable}

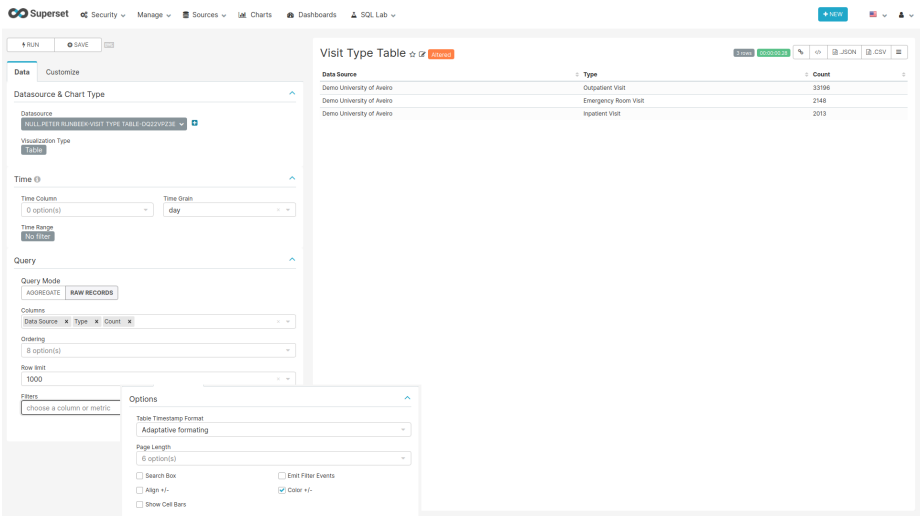


Figure 6.18: Settings for creating the Visit Type Table chart

SQL query

```
SELECT source.name,
       source.acronym,
       concept_name AS "Type",
       MAX(count_value) AS "Count"
FROM public.achilles_results AS achilles
INNER JOIN public.data_source AS source ON achilles.data_source_id=source.id
INNER JOIN public.concept ON CAST(stratum_1 AS BIGINT) = concept_id
WHERE analysis_id = 201
GROUP BY name, acronym, "Type"
ORDER BY "Count" DESC
```

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Table
  - Time
    - Time range: No filter
  - Query
    - Query Mode: Raw Records
    - Columns: name with label “Data Source”, Type, Count

Visit Types Bars

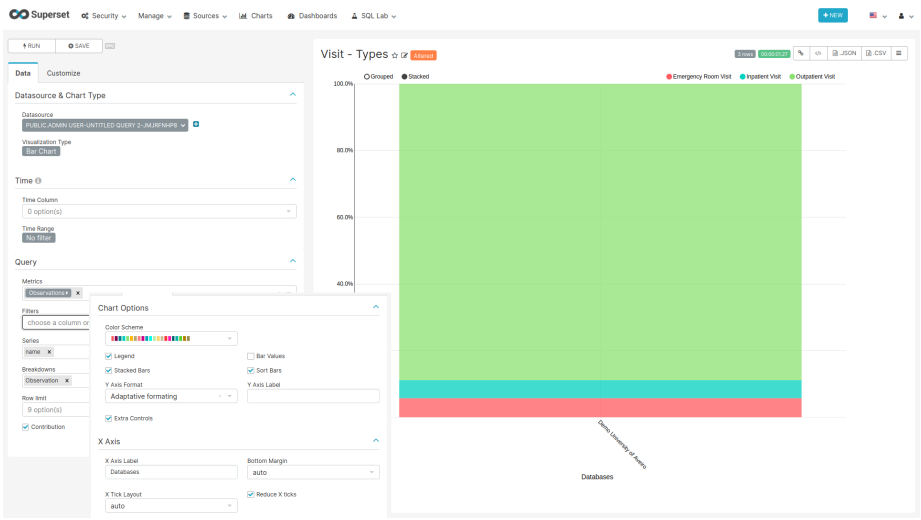


Figure 6.19: Settings for creating the Visit Types bar chart

SQL query

```
SELECT source.name,
       source.acronym,
       concept_name AS "Observation",
       count_value
FROM public.achilles_results AS achilles
INNER JOIN public.data_source AS source ON achilles.data_source_id=source.id
INNER JOIN public.concept ON CAST(stratum_1 AS BIGINT) = concept_id
WHERE analysis_id = 201
```

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Bar Chart
  - Time
    - Time range: No filter
  - Query
    - Metrics: MAX(count_value) with label Observations
    - Series: name
    - Breakdowns: Observation
- Customize Tab
  - Chart Options
    - Stacked Bars: on
    - Sort Bars: on
    - Extra Controls: on
  - X Axis
    - X Axis Label: Databases
    - Reduce X ticks: on

6.5 Death

CSS

To hide the dashboard header insert the following css code to the `CSS` field on the edit page:

```
.dashboard > div:not(.dashboard-content) {  /* dashboard header */
  display: none;
}
```

With this every time you want to edit the dashboard layout you have to either comment the CSS inserted or remove it so the "Edit Dashboard" button can show again.

Data Source Filter

For the filter to work the name of the fields to filter should match in all tables used on the charts of this dashboard.

SQL query

No SQL query, use the sql table `data_source` of the `achilles` database.

Chart settings

- Data Tab

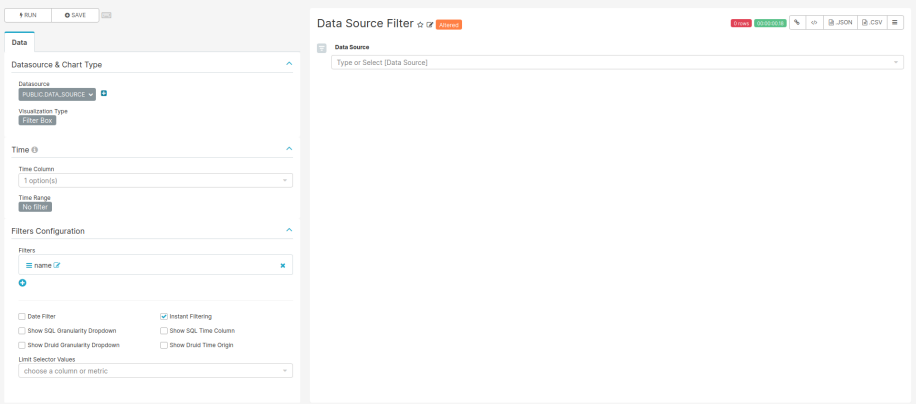


Figure 6.20: Settings for creating the Data Source filter chart

  - Datasource & Chart Type
    - Visualization Type: Filter Box
  - Time
    - Time range: No filter
  - Filters Configuration
    - Filters:
      - name
    - Date Filter: off
    - Instant Filtering: on

Number of Records

SQL query

```
SELECT source.name,
    count_value,
    source.acronym
FROM public.achilles_results AS achilles
INNER JOIN public.data_source AS source ON achilles.data_source_id=source.id
WHERE analysis_id = 501
```

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Bar Chart
  - Time
    - Time range: No filter
  - Query

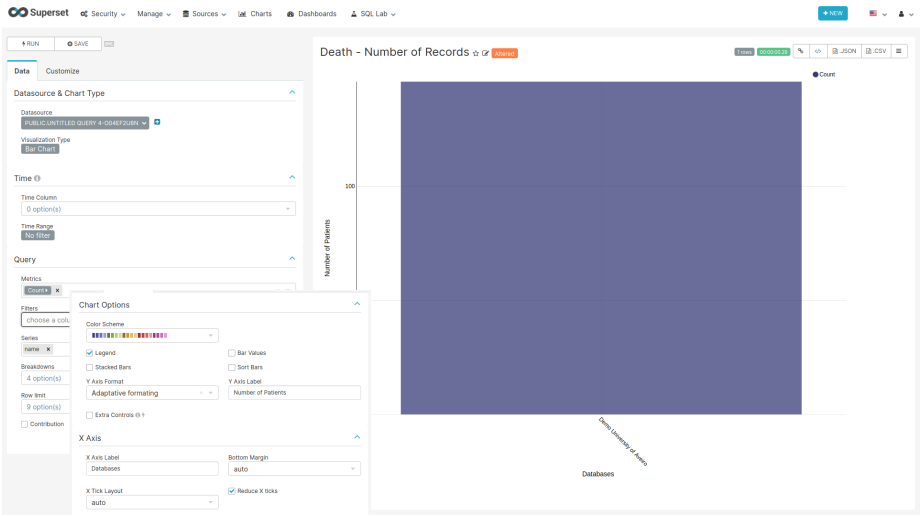


Figure 6.21: Settings for creating the Number of Records chart

    - Metrics: MAX(count_value) with label Count
    - Series: name
- Customize Tab
  - Chart Options
    - Y Axis Label: Number of Patients
  - X Axis
    - X Axis Label: Databases
    - Reduce X ticks: on

Death By Year per Thousand People

SQL query

```
SELECT source.name,
    source.acronym,
    EXTRACT(year FROM TO_DATE(stratum_1, 'YYYYMM')) AS Date,
    count_value
FROM public.achilles_results as achilles
INNER JOIN public.data_source as source ON achilles.data_source_id=source.id
WHERE analysis_id = 502
```

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Bar Chart

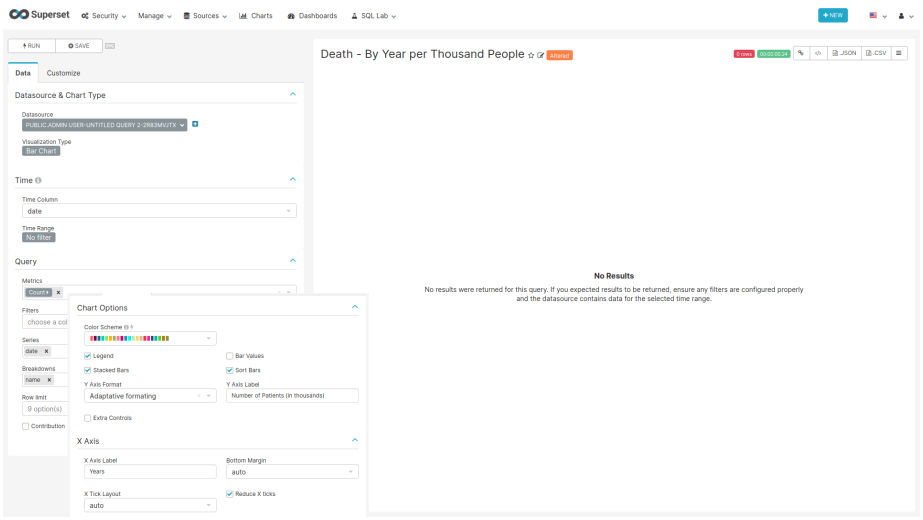


Figure 6.22: Settings for creating the Death by Year per Thousand People chart

  - Time
    - Time range: No filter
  - Query
    - Metrics: MAX(count_value) with label Count
    - Series: date
    - Breakdowns: name
- Customize Tab
  - Chart Options
    - Stacked Bars: on
    - Sort Bars: on
    - Y Axis Label:Number of Patients (in thousands)
  - X Axis
    - X Axis Label: Years
    - Reduce X ticks: on

6.6 Concepts Browser

The concepts browser allows you to search for concepts by name or concept_id in all the data sources you select. No exact number of patients or occurrences are provided but the magnitude of both.

CSS

To hide the dashboard header insert the following css code to the `CSS` field on the edit page:

```
.dashboard > div:not(.dashboard-content) {  /* dashboard header */
  display: none;
}
```

With this every time you want to edit the dashboard layout you have to either comment the CSS inserted or remove it so the "Edit Dashboard" button can show again.

Data Source and Domain Filters

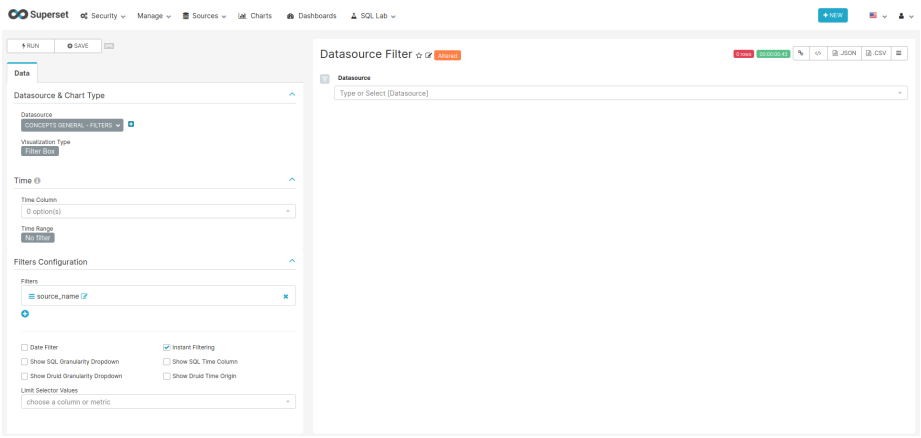


Figure 6.23: Settings for creating the Data Source and Domain filter charts

For the filters to work the name of the fields to filter should match in all tables used on the charts of this dashboard.

SQL query

```
SELECT concept_name,
     domain_id,
     source.name AS source_name,
     source.acronym
FROM achilles_results
JOIN concept ON cast(stratum_1 AS BIGINT) = concept_id
INNER JOIN public.data_source AS source ON data_source_id=source.id
WHERE analysis_id in (201, 401, 601, 701, 801, 901, 1001, 1801, 200, 400, 600, 700, 800, 1800)
```

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Filter Box

  - Time
    - Time range: No filter
  - Filters Configuration
    - Filters:
      - source_name or domain_id
    - Date Filter: off
    - Instant Filtering: on

Number of Concepts

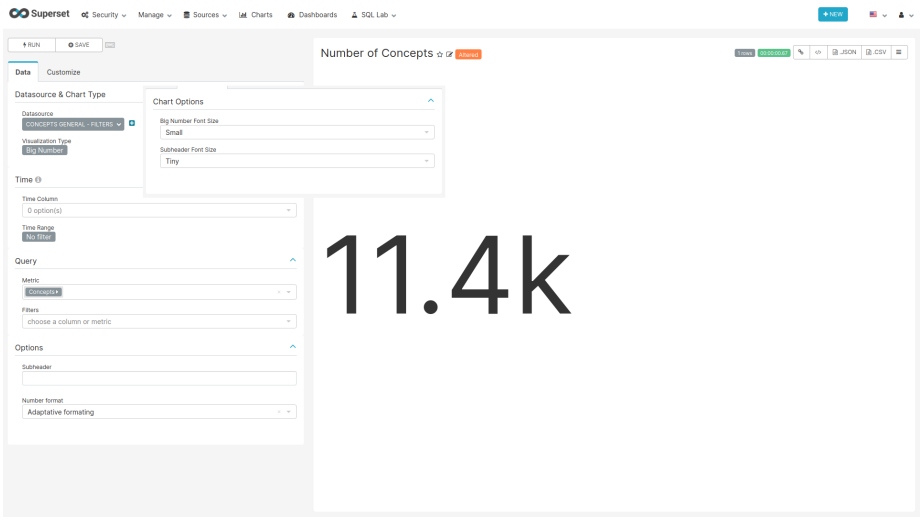


Figure 6.24: Settings for creating the Number of Concepts chart

SQL Query

Same as Data Source and Domain filters query

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Big Number
  - Time Time range: No filter
  - Query
    - Metric: COUNT_DISTINCT(concept_name) with label Concepts
- Customize Tab
  - Big Number Font Size: Small
  - Subheader Font Size: Tiny

Concept Browser Table {#conceptBrowserTable}

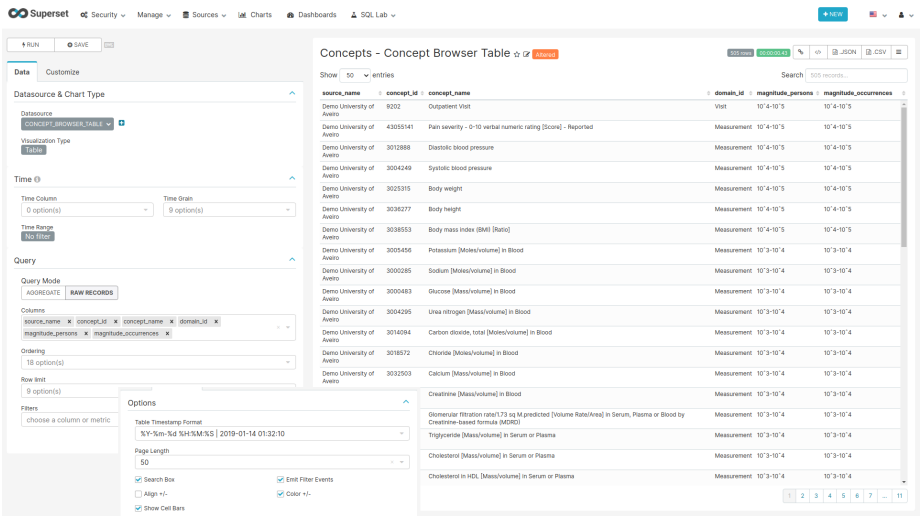


Figure 6.25: Settings for creating the Concepts Table chart

```
SELECT
    q1.concept_id AS concept_id,
    q1.concept_name AS concept_name,
    q1.domain_id,
    source.name AS source_name,
    source.acronym,
    sum(q1.count_value) as "Occurrence_count",
    sum(q1.count_person) as "Person_count",
    CASE
        WHEN sum(q1.count_value)<=10 THEN '<=10'
        WHEN sum(q1.count_value)<=100 THEN '11-10^2'
        WHEN sum(q1.count_value)<=1000 THEN '10^2-10^3'
        WHEN sum(q1.count_value)<=10000 THEN '10^3-10^4'
        WHEN sum(q1.count_value)<=100000 THEN '10^4-10^5'
        WHEN sum(q1.count_value)<=1000000 THEN '10^5-10^6'
        ELSE '>10^6'
    END as "magnitude_occurrences",
    CASE
        WHEN sum(q1.count_person)<=10 THEN '<=10'
        WHEN sum(q1.count_person)<=100 THEN '11-10^2'
        WHEN sum(q1.count_person)<=1000 THEN '10^2-10^3'
        WHEN sum(q1.count_person)<=10000 THEN '10^3-10^4'
        WHEN sum(q1.count_person)<=100000 THEN '10^4-10^5'
        WHEN sum(q1.count_person)<=1000000 THEN '10^5-10^6'
        ELSE '>10^6'
```

```
    END AS "magnitude_persons"
FROM (SELECT analysis_id,
             stratum_1 concept_id,
             data_source_id,
             concept_name,
             domain_id,
             count_value, 0 as count_person
    FROM achilles_results
    JOIN concept ON cast(stratum_1 AS BIGINT)=concept_id
    WHERE analysis_id in (201, 301, 401, 601, 701, 801, 901, 1001, 1801)
    UNION (SELECT  analysis_id,
                   stratum_1 concept_id,
                   data_source_id,
                   concept_name,
                   domain_id,
                   0 as count_value,
                   sum(count_value) as count_person
            FROM  achilles_results
            JOIN concept on cast(stratum_1 as BIGINT)=concept_id
            WHERE analysis_id in (202, 401, 601, 701, 801, 901, 1001, 1801)
            GROUP BY analysis_id,stratum_1,data_source_id,concept_name,domain_id) ) as
    INNER JOIN public.data_source AS source ON q1.data_source_id=source.id
GROUP BY q1.concept_id,q1.concept_name,q1.domain_id,source.name, acronym
ORDER BY "Person_count" desc
```

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Table
  - Time
    - Time range: No filter
  - Query
    - Query Mode: Raw Records
    - Columns: source_name, concept_id, concept_name, domain_id, magnitude_persons, magnitude_occurrences
- Customize Tab
  - Options
    - Table Timestamps Format: %Y-%m-%d %H:%M:%S | 2019-01-14 01:32:10
    - Page Length: 50
    - Search Box: on
    - Emit Filter Events: on

6.7 Provenance

This Dashboard shows the provenance of the data in the different data domains.

CSS

To hide the dashboard header insert the following css code to the `CSS` field on the edit page:

```
.dashboard > div:not(.dashboard-content) {  /* dashboard header */
  display: none;
}
```

With this every time you want to edit the dashboard layout you have to either comment the CSS inserted or remove it so the "Edit Dashboard" button can show again.

Data Source Filter

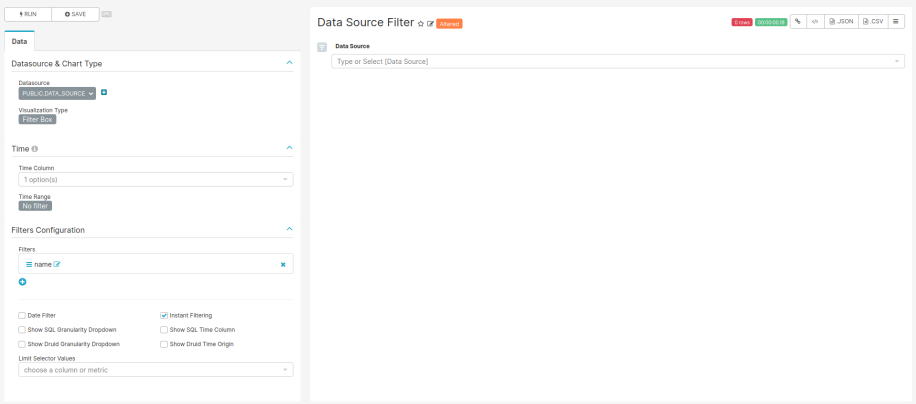


Figure 6.26: Settings for creating the Data Source filter chart

For the filter to work the name of the fields to filter should match in all tables used on the charts of this dashboard.

SQL query

No SQL query, use the sql table `data_source` of the `achilles` database.

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Filter Box

- Time
  - Time range: No filter
- Filters Configuration
  - Filters:
    - name
  - Date Filter: off
  - Instant Filtering: on

Condition & Drug & Procedure & Device & Measurement & Observation Types {#dataProvenanceCharts}

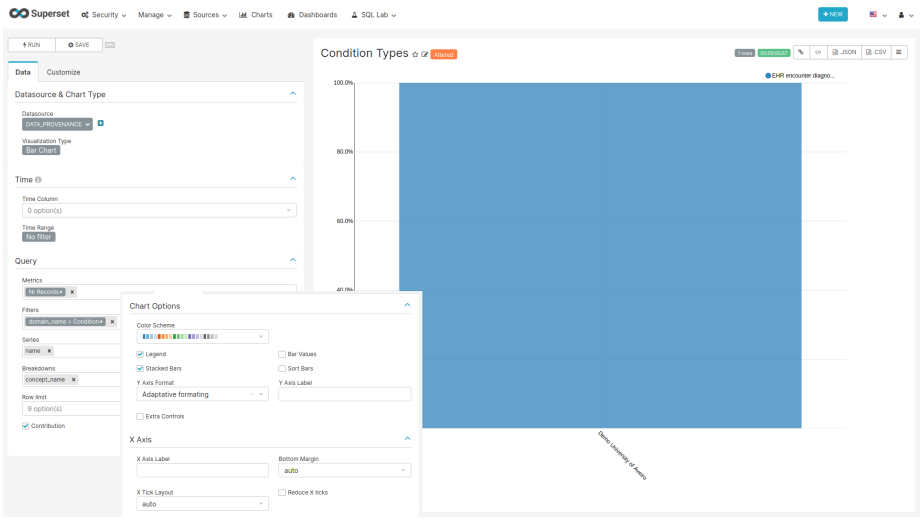


Figure 6.27: Settings for creating the Condition, Drug, Procedure, Device, Measurement and Observation charts

SQL query

All 6 charts use the same sql query.

```
SELECT source.name,
    source.acronym,
    CASE WHEN analysis_id = 405 THEN 'Condition'
    WHEN analysis_id = 605 THEN 'Procedure'
    WHEN analysis_id = 705 THEN 'Drug'
    WHEN analysis_id = 805 THEN 'Observation'
    WHEN analysis_id = 1805 THEN 'Measurement'
    WHEN analysis_id = 2105 THEN 'Device'
    ELSE 'Other' END AS domain_name,
    concept_name,
    SUM(count_value) AS num_records
```

```
FROM public.achilles_results AS achilles
INNER JOIN public.data_source AS source ON achilles.data_source_id=source.id
INNER JOIN public.concept AS c1 ON CAST(stratum_2 AS BIGINT) = concept_id
WHERE analysis_id IN (405,605,705,805,1805,2105)
GROUP BY source.name, source.acronym, concept_name,
    CASE WHEN analysis_id = 405 THEN 'Condition'
    WHEN analysis_id = 605 THEN 'Procedure'
    WHEN analysis_id = 705 THEN 'Drug'
    WHEN analysis_id = 805 THEN 'Observation'
    WHEN analysis_id = 1805 THEN 'Measurement'
    WHEN analysis_id = 2105 THEN 'Device'
    ELSE 'Other' END
```

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Bar Chart
  - Time
    - Time range: No filter
  - Query
    - Metrics: SUM(num_records) with label Nr Records
    - Filters: domain_name=Condition or domain_name=Drug or domain_name=Procedure or domain_name=Device or domain_name=Measurement or domain_name=Observation
    - Series: name
    - Breakdowns: concept_name
    - Contribution: on
- Customize Tab
  - Chart Options
    - Stacked Bars: on

6.8 Data Domains

CSS

To hide the dashboard header insert the following css code to the `CSS` field on the edit page:

```
.dashboard > div:not(.dashboard-content) {  /* dashboard header */
  display: none;
}
```

With this every time you want to edit the dashboard layout you have to either comment the CSS inserted or remove it so the “Edit Dashboard” button can show again.

Data Source Filter

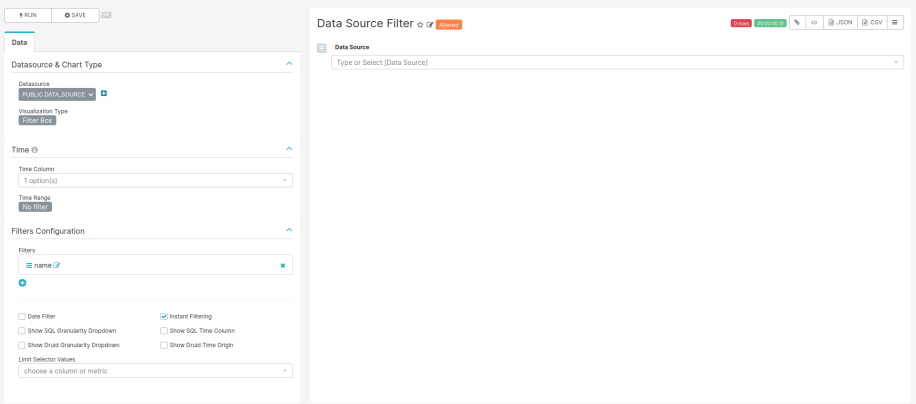


Figure 6.28: Settings for creating the Data Source filter chart

For the filter to work the name of the fields to filter should match in all tables used on the charts of this dashboard.

SQL query

No SQL query, use the sql table `data_source` of the `achilles` database.

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Filter Box
  - Time
    - Time range: No filter
  - Filters Configuration
    - Filters:
      - name
    - Date Filter: off
    - Instant Filtering: on

Average Number of Records per Person {#avgRecordsPerPerson}

SQL query

```
SELECT
    source.name,
    source.acronym,
```

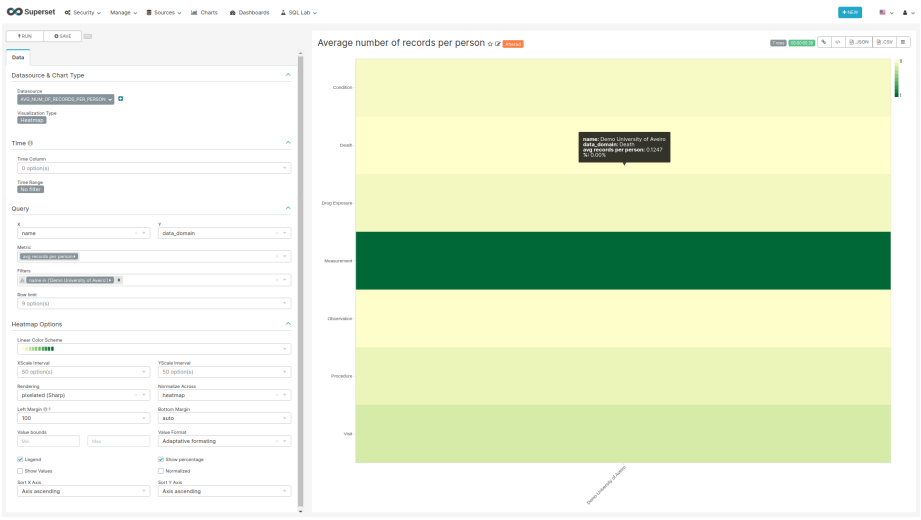


Figure 6.29: Settings for creating the Data Source filter chart

```
    CASE
    WHEN analysis_id = 201 THEN 'Visit'
    WHEN analysis_id = 401 THEN 'Condition'
    WHEN analysis_id = 501 THEN 'Death'
    WHEN analysis_id = 601 THEN 'Procedure'
    WHEN analysis_id = 701 THEN 'Drug Exposure'
    WHEN analysis_id = 801 THEN 'Observation'
    WHEN analysis_id = 1801 THEN 'Measurement'
    WHEN analysis_id = 2101 THEN 'Device'
    WHEN analysis_id = 2201 THEN 'Note'
    END AS Data_Domain,
    SUM(count_value) /AVG(num_persons) AS "records_per_person"
FROM public.achilles_results AS achilles
INNER JOIN public.data_source AS source ON achilles.data_source_id=source.id
INNER JOIN (
  SELECT data_source_id , count_value as num_persons
  FROM achilles_results
  WHERE analysis_id = 1) counts ON achilles.data_source_id = counts.data_source_id
GROUP BY analysis_id, source.name, source.acronym
HAVING analysis_id IN (201, 401, 501, 601, 701, 801, 1801, 2101, 2201)
```

Chart settings

- Data Tab
  - Datasource & Chart Type

    - Visualization Type: Heatmap
  - Time
    - Time range: No filter
  - Query
    - X: name
    - Y: data_domain
    - Metric: AVG(records_per_person) with a label avg records per person
    - Row limit: None
  - Heatmap Options
    - Left Margin: 100
    - Show Percentage: off

6.9 Per Database

Label Colors

In order to obtain the colors blue and rose in the chart representing the gender distribution, add the following JSON entry to the JSON object of the `JSON Metadata` field on the edit dashboard page:

```
"label_colors": {
    "Male": "#3366FF",
    "Female": "#FF3399"
}
```

CSS

To hide the dashboard header insert the following css code to the `CSS` field on the edit page:

```
.dashboard > div:not(.dashboard-content) {  /* dashboard header */
  display: none;
}
```

With this every time you want to edit the dashboard layout you have to either comment the CSS inserted or remove it so the "Edit Dashboard" button can show again.

Data Source Filter

For the filter to work the name of the fields to filter should match in all tables used on the charts of this dashboard.

SQL query

No SQL query, use the sql table `data_source` of the `achilles` database.

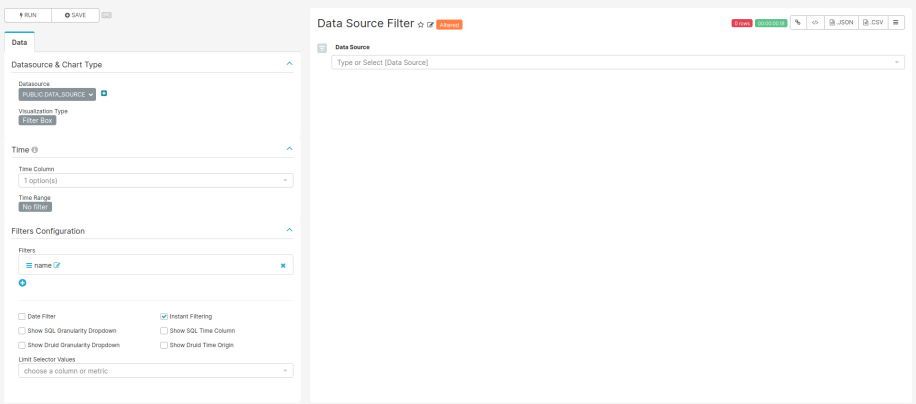


Figure 6.30: Settings for creating the Data Source filter chart

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Filter Box
  - Time
    - Time range: No filter
  - Filters Configuration
    - Filters:
      - name
    - Date Filter: off
    - Instant Filtering: on

Demographics Tab

Number of Patients

```
SELECT
  achilles_results.count_value,
  data_source.name,
  data_source.acronym
FROM achilles_results
JOIN data_source ON achilles_results.data_source_id=data_source.id
WHERE analysis_id = 1
```

SQL query

Chart settings

- Data Tab

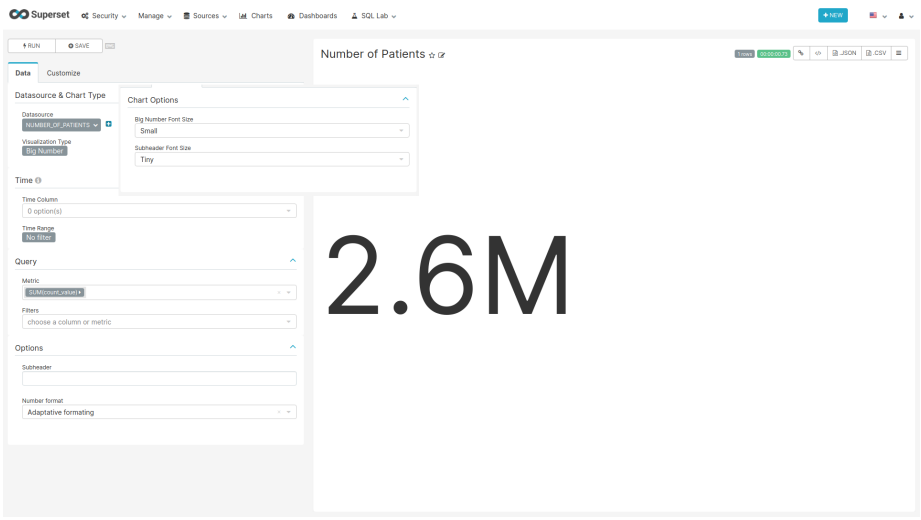


Figure 6.31: Settings for creating the Number of Patients chart

  - Datasource & Chart Type
    - Visualization Type: Big Number
  - Time Time range: No filter
  - Query
    - Metric: sum(count_value)
- Customize Tab
  - Big Number Font Size: Small
  - Subheader Font Size: Tiny

Gender Table

```
SELECT source.name as name,
       source.acronym,
       concept_name as gender,
       count_value
FROM public.achilles_results AS achilles
INNER JOIN public.data_source AS source ON achilles.data_source_id=source.id
INNER JOIN public.concept ON CAST(stratum_1 AS BIGINT) = concept_id
WHERE analysis_id = 2
```

SQL Query {#genderTableQuery}

Chart settings

- Data Tab
  - Datasource & Chart Type

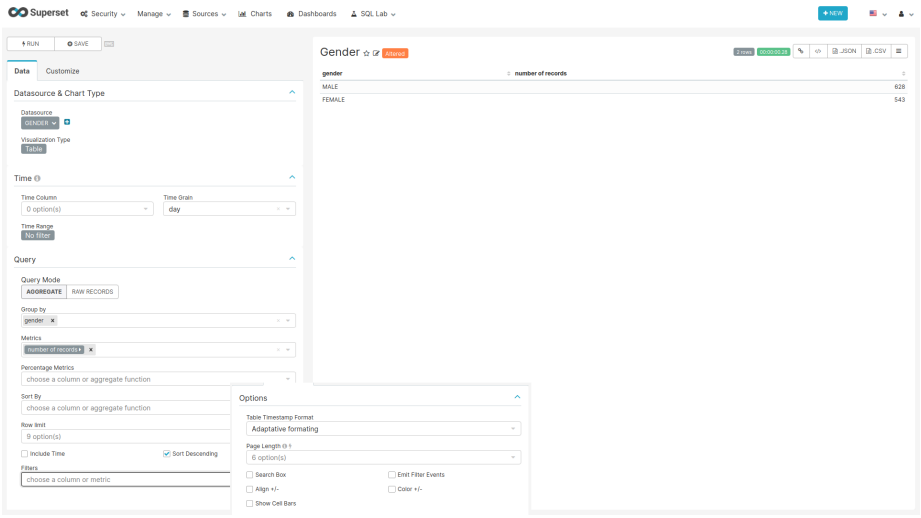


Figure 6.32: Settings for creating the Gender Table chart

    - Visualization Type: Table
  - Time
    - Time range: No filter
  - Query
    - Query Mode: Aggregate
    - Group by: gender
    - Metrics: SUM(count_value) with label number of records
    - Row lmit: None
- Customize Tab
  - Options
    - Show Cells Bars: off

Gender Pie

SQL query Same as Gender Table query

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Pie Chart
  - Time
    - Time range: No filter
  - Query
    - Metric: SUM(count_value)
    - Group by: gender

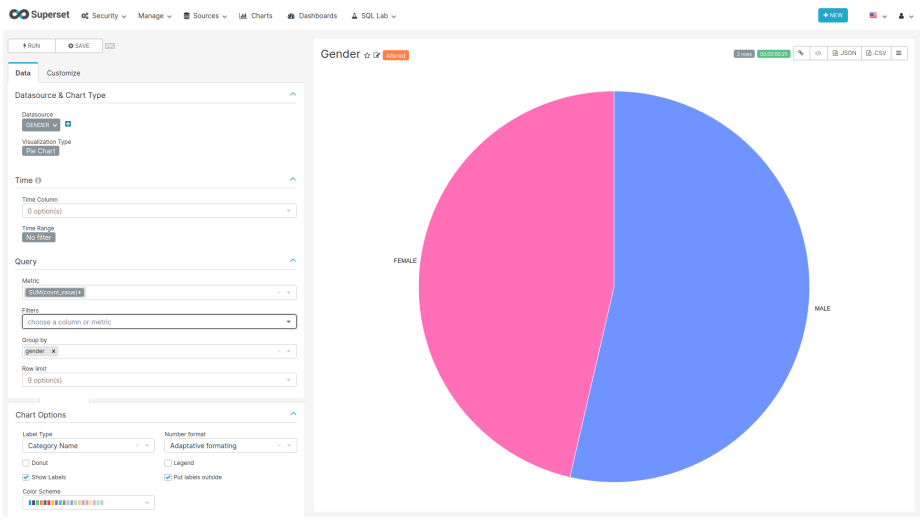


Figure 6.33: Settings for creating the Gender Pie chart

        - Row limit: None
- Customize Tab
    - Chart Options
        - Legend: off

Age at first observation - Table

Same chart as the one used on the Person dashboard.

Age at first observation - Bars

Same chart as the one used on the Person dashboard.

Year of Birth

Same chart as the one used on the Person dashboard.

Data Domains Tab

Average Number of Records per Person

Same chart as the one used on the Data Domains dashboard.

Total Number of Records

```
SELECT
data_source.name,
```

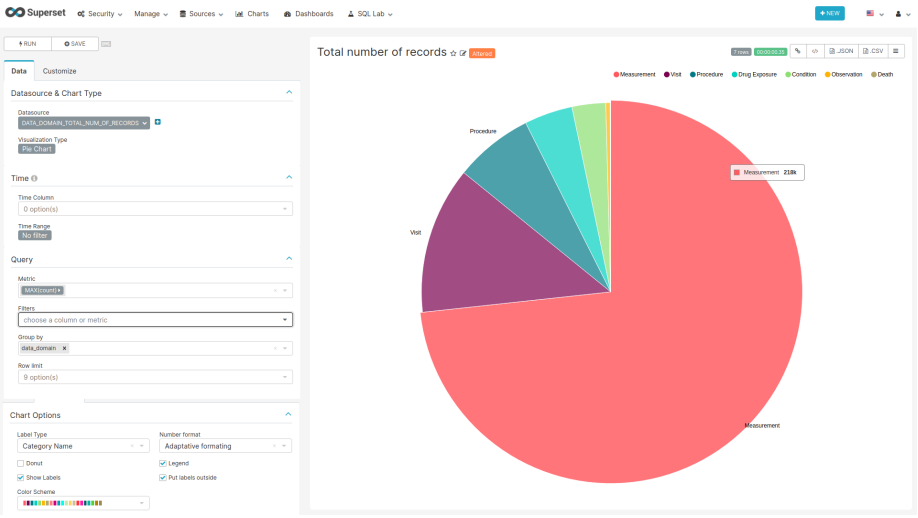


Figure 6.34: Settings for creating the Total Number of Records chart

```
data_source.acronym,
    CASE
    WHEN analysis_id = 201 THEN 'Visit'
    WHEN analysis_id = 401 THEN 'Condition'
    WHEN analysis_id = 501 THEN 'Death'
    WHEN analysis_id = 601 THEN 'Procedure'
    WHEN analysis_id = 701 THEN 'Drug Exposure'
    WHEN analysis_id = 801 THEN 'Observation'
    WHEN analysis_id = 1801 THEN 'Measurement'
    WHEN analysis_id = 2101 THEN 'Device'
    WHEN analysis_id = 2201 THEN 'Note'
    END AS Data_Domain,
    SUM(count_value) AS "count"
FROM achilles_results
JOIN data_source ON achilles_results.data_source_id=data_source.id
GROUP BY name, acronym, analysis_id
HAVING analysis_id IN (201, 401, 501, 601, 701, 801, 1801, 2101, 2201)
```

SQL query

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Pie Chart
  - Time

    - Time range: No filter
  - Query
    - Metric: MAX(count)
    - Group by: data_domain
    - Row limit: None

Data Provenance Tab

Same six charts used on the Provenance dashboard.

Observation Period Tab

Number of Patitents in Observation Period

Same chart used on the Observation Period dashboard.

Cumulative Observation Period

The cumulative observation time plot shows the percentage of patients that have more that X days of observation time.

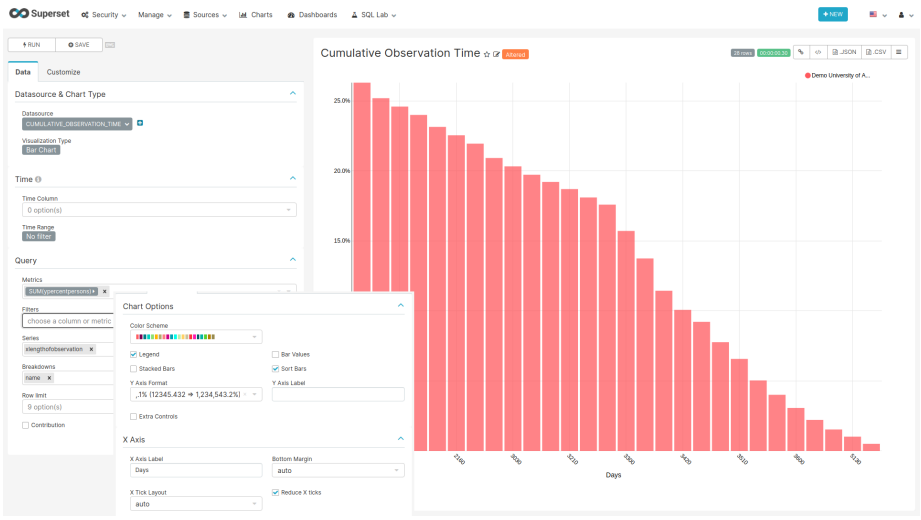


Figure 6.35: Settings for creating the Total Number of Records chart

```
SELECT
  name,
  acronym,
  xLengthOfObservation,
  round(cumulative_sum / total, 5) as yPercentPersons
```

```
FROM (
  SELECT data_source_id, CAST(stratum_1 AS INTEGER) * 30 AS xLengthOfObservation, SUM(count_value
  FROM achilles_results
  WHERE analysis_id = 108
) AS cumulative_sums
JOIN (
  SELECT data_source_id, count_value as total
  FROM achilles_results
  WHERE analysis_id = 1
) AS totals
ON cumulative_sums.data_source_id = totals.data_source_id
JOIN data_source ON cumulative_sums.data_source_id = data_source.id
ORDER BY name, xLengthOfObservation
```

SQL Query

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Bar Chart
  - Time
    - Time range: No filter
  - Query
    - Metrics: SUM(ypercentpersons)
    - Series: xlengthofobservation
    - Breakdowns: name
    - Row limit: None
- Customize Tab
  - Chart Options
    - Sort Bars: on
    - Y Axis Fomat: ,.1% (12345.432 => 1,234,543.2%)
    - Y Axis Label: Number of Patients
  - X Axis
    - X Axis Label: Days
    - Reduce X ticks: on

Visit Tab

Visit Type Graph

```
SELECT
  data_source.name,
  data_source.acronym,
  concept.concept_name,
```

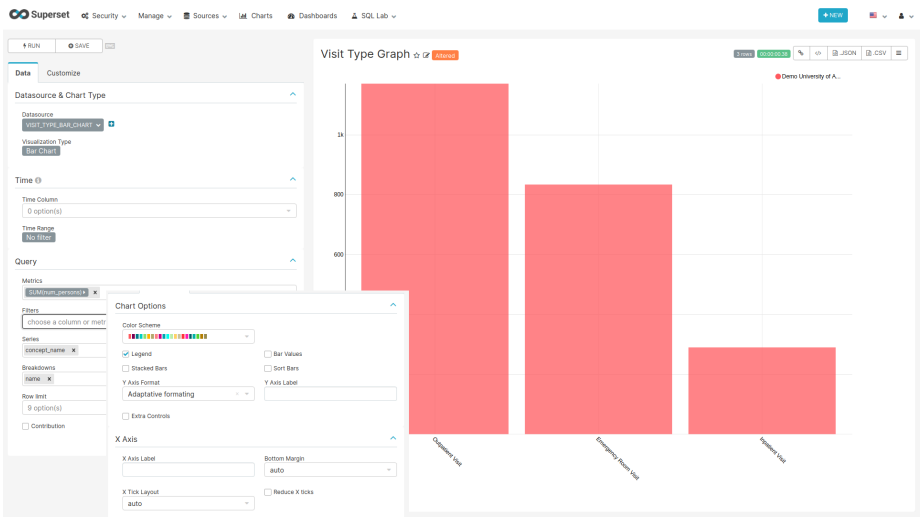


Figure 6.36: Settings for creating the Visit Type Graph chart

```
  achilles_results.count_value AS num_persons
FROM (SELECT * FROM achilles_results WHERE analysis_id = 200) AS achilles_results
JOIN data_source ON achilles_results.data_source_id = data_source.id
JOIN concept ON CAST(achilles_results.stratum_1 AS BIGINT) = concept.concept_id
```

SQL Query

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Bar Chart
  - Time
    - Time range: No filter
  - Query
    - Metrics: SUM(num_persons)
    - Series: concept_name
    - Breakdowns: name
    - Row limit: None

Visit Type Table

```
SELECT
  name,
  acronym,
```

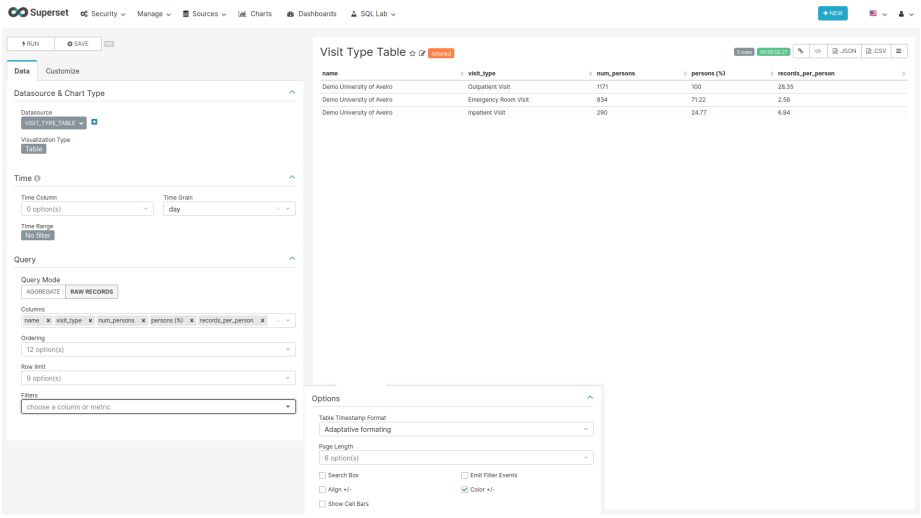


Figure 6.37: Settings for creating the Visit Type Table chart

```
  concept.concept_name,
  ar1.count_value AS num_persons,
  round(100.0 * ar1.count_value / denom.count_value, 2) AS percent_persons,
  round(1.0 * ar2.count_value / ar1.count_value, 2) AS records_per_person
FROM (
  SELECT *
  FROM achilles_results WHERE analysis_id = 200) AS ar1
  JOIN (
    SELECT *
    FROM achilles_results WHERE analysis_id = 201) AS ar2
    ON ar1.stratum_1 = ar2.stratum_1 AND ar1.data_source_id = ar2.data_source_id
  JOIN (
    SELECT *
    FROM achilles_results WHERE analysis_id = 1) AS denom
    ON ar1.data_source_id = denom.data_source_id
  JOIN data_source ON data_source.id = ar1.data_source_id
  JOIN concept ON CAST(ar1.stratum_1 AS INTEGER) = concept_id
ORDER BY ar1.data_source_id, ar1.count_value DESC
```

SQL Query

Chart Settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Table

    - Time
        * Time range: No filter
    - Query
        * Query Mode: Raw Records
        * Columns: name, visit_type, num_persons, percent_persons with label persons (%), records_per_person
        * Row limit: None
- Customize Tab
    - Options
        * Show Cell Bars: off

Concept Browser Tab

Concept Browser Table

Same chart used on the Concept Browser dashboard.

Meta Data Tab

Meta Data Table

Same chart used on the General dashboard.

6.10 Database Level Dashboard

This dashboard is an exact copy of the Per Database dashboard but several legends and fields displayed on the original are hidden either through CSS or by changing some chart settings. On the following sections we will only present the things to change on the original charts.

Label Colors

In order to obtain the colors blue and rose in the chart representing the gender distribution, add the following JSON entry to the JSON object of the `JSON Metadata` field on the edit dashboard page:

```
"label_colors": {
    "Male": "#3366FF",
    "Female": "#FF3399"
}
```

CSS

To hide the dashboard header insert the following css code to the `CSS` field on the edit page:

```
/* hides the filter badges on right side of charts */
.dashboard-filter-indicators-container {
    display: none;
}
/* hides the acronym filter */
.grid-content > .dragdroppable.dragdroppable-row > .with-popover-menu {
    display: none;
}
/*
WARNING panel 1 id hardcoded
Hides the X Axis Label of the heatmap on the Data Domains tab
*/
#TABS-nlIU6H5mcT-pane-1 g.x.axis > g.tick text {
    display: none;
}
/*
WARNING panel 2 id hardcoded
Hides the X Axis Labels of the bar charts on the Data Provenance tab
*/
#TABS-nlIU6H5mcT-pane-2 g.nv-x.nv-axis.nvd3-svg > g.nvd3.nv-wrap.nv-axis > g > g.tick.zero > text
    display: none;
}
```

With this every time you want to edit the dashboard layout you have to either comment the CSS inserted or remove it so the "Edit Dashboard" button can show again.

Data Source Filter - hidden

For the filter to work the name of the fields to filter should match in all tables used on the charts of this dashboard.

SQL query

No SQL query, use the sql table `data_source` of the `achilles` database.

Chart settings

- Data Tab
  - Datasource & Chart Type
    - Visualization Type: Filter Box
  - Time
    - Time range: No filter
  - Filters Configuration
    - Filters:
      - acronym

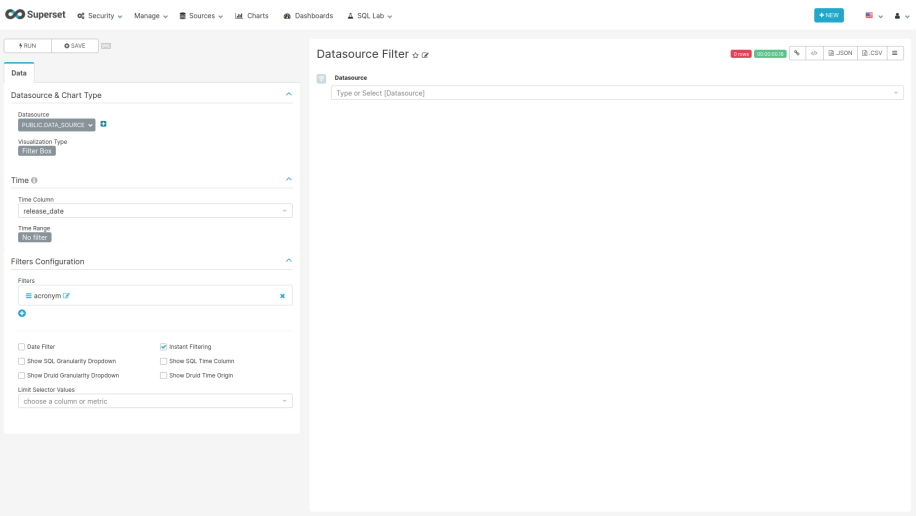


Figure 6.38: Settings for creating the Data Source filter chart

    * Date Filter: off
    * Instant Filtering: on

Demographics Tab

Number of Patients

No changes

Gender Table

No changes

Gender Pie

No changes

Age at first observation - Table

Remove the `name` field from the columns to display.

- Data Tab
  - Query
    * Columns: 0-10, 10-20, 20-30, 30-40, 40-50, 50-60, 60-70, 70-80, 80-90, 90+

Age at first observation - Bars

Remove legend.

- Customize Tab
  - Chart Options
    - Legend: off

Year of Birth

Remove legend.

- Customize Tab
  - Chart Options
    - Legend: off

Data Domains Tab

No changes

Data Provenance Tab

No changes

Observation Period Tab

Number of Patitents in Observation Period

Remove legend.

- Customize Tab
  - Chart Options
    - Legend: off

Cumulative Observation Period

Remove legend.

- Customize Tab
  - Chart Options
    - Legend: off

Visit Tab

Visit Type Graph

Remove legend.

- Customize Tab
  - Chart Options
    - Legend: off

Visit Type Table

Remove the `name` field from the columns to display.

- Data Tab
  - Query
    - Columns: visit_type, num_persons, percent_persons with label persons (%), records_per_person

Concept Browser Tab

Concept Browser Table

Remove the `source_name` field from the columns to display.

- Data Tab
  - Query
    - Columns: concept_id, concept_name, domain_id, magnitude_persons, magnitude_occurrences

Meta Data Tab

Meta Data Table

Remove the `name` field from the columns to display.

- Data Tab
  - Query
    - Columns: source_release_date, cdm_release_date, cdm_version, vocabulary_version