# Data Network Dashboards

This document is currently under construction

2020-03-03

# Contents

# Preface

**This document is currently under construction**

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 proposed solution aggregates ACHILLES results files from databases in the network and displays the descriptive statistics through graphical dashboards. The 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.

## Goals

This manual aims to document the procedures to install and configure the chasrts and dashboards choosen to compare the OHDSI databases.

….

## 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 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

## License

The system is open-source under the license ….

The book is written in RMarkdown using the bookdown package.

## 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 1

# 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.

## 1.1 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 1.1). 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:

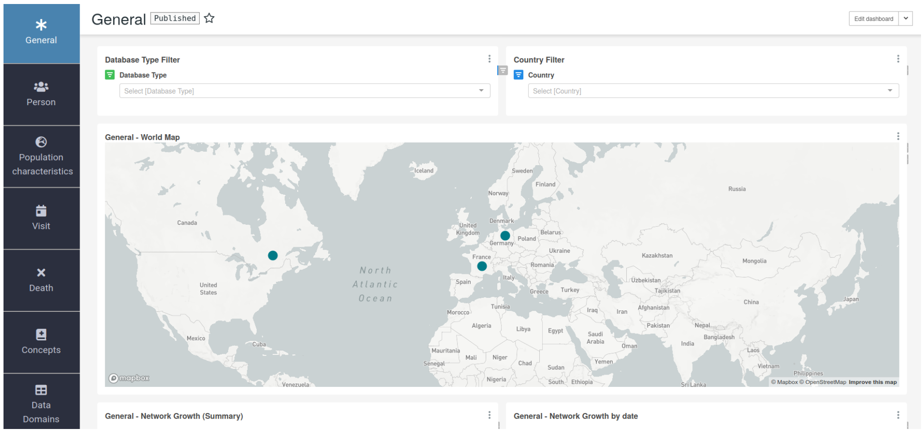


Figure 1.1: Example of a dashboards tool presenting the databases available in the network (simulated data)

- **General**: dashboards that shows the databases types per country, the distribution of data source types, the growth of the Network including 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 2

# Installation

Make sure that you have docker and docker-compose installed in your machine. Then, please follow these steps:

- Please enter in the ''docker" directory and create your `.env` file here, using `.env-example` as reference. For local installation, you can just copy the `.env-example` content to a new file. Note: In case of port errors in the next steps, the problem could be related to a port already in use by your system that you defined here and it is busy, chose other.
- Tip the following commands in the command line:
  1. Clone the Apache Superset repository:

     ```
     git clone https://github.com/apache/incubator-superset
      ../superset
     cp ../superset/contrib/docker/superset_config.py ../superset
     ```

  2. Init the Apache Superset (This creates a user, so it is necessary to interact with the console):

     ```
     docker-compose run --rm superset ./docker-init.sh
     ```

  3. Init the Dashboard Layout (This creates a user, so it is necessary to interact with the console):

     ```
     docker-compose run --rm dashboard_viewer ./docker-init.sh
     ```

  4. Finally, bring up the containers

     ```
     docker-compose up -d
     ```

To check if everything is ok, please wait 2 minutes and tip `docker ps` and the following containers need to be running:

```
... 0.0.0.0:8088->8088/tcp   dashboard_viewer_superset_1
```

```
... 0.0.0.0:8000->8000/tcp   dashboard_viewer_dashboard_viewer_1
... 0.0.0.0:6379->6379/tcp   dashboard_viewer_redis_1
... 5432/tcp                 dashboard_viewer_postgres_1
```

Now, you have a clean setup running in your machine. To try the application using synthetic data, please continue to follow the steps in the ‘’Demo” section.

## 2.1 Insert Concepts

The concepts table are not in the repository due to its dimension. Therefore, to insert this table in the installation, you should perform the following steps:

1. Download concept.csv file from here (todo)
2. Copy the file to the /tmp directory inside of the postgres container

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

3. Enter in the dashboard_viewer_postgres_1 container:

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

4. Enter in the achilles database:

   ```
   psql achilles
   ```

5. Create the table in the database using this command:

   ```
       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):

   ```
   COPY public.concept from '/tmp/concept.csv' WITH DELIMITER ','
       CSV HEADER;
   ```

7. Alter table ownership:

   ```
   -- <user> : defined in the .env file
   ALTER TABLE public.concept OWNER TO <user>;
   ```

8. Create index in table:

```
CREATE INDEX achilles_results_analysis_id_index ON
    achilles_results (analysis_id);
CREATE INDEX achilles_results_source_index ON achilles_results
    (data_source_id);
CREATE INDEX concept_concept_id_index ON concept (concept_id);
CREATE INDEX concept_concept_name_index ON concept
    (concept_name);
```

## 2.2 Import dashboards

TO DO

## 2.3 Dummy data

TO DO

# Chapter 3

# General

VERY INCOMPLETE Discuss the goal of this dashboard... TO DO

## 3.1 Database Type Filter

This filter which is a type of chart in Superset was designed to be used in the dashboard aiming the filtering of the data based on the field ‘’database_type’‘from the table’‘data_source’‘. It is important to give the alias’‘Type” to this field in the select operations because Superset does not recognize as the same field otherwise.

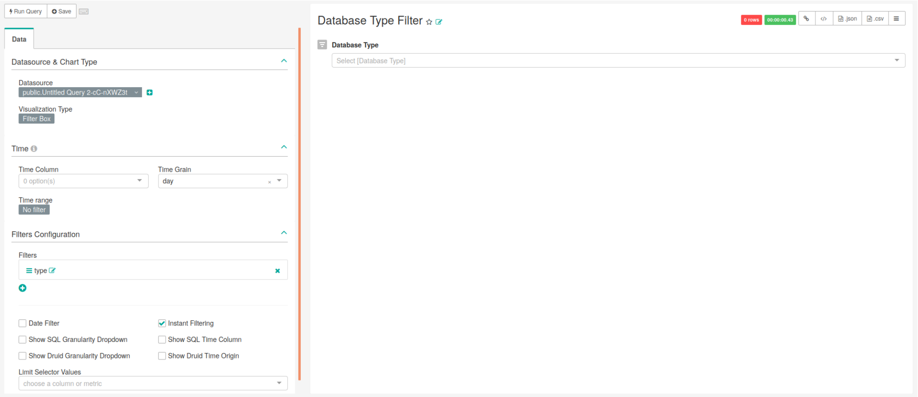
### 3.1.1 SQL query

```sql
--  Country and database type filters
SELECT source.name,
       country.country AS Country,
       database_type AS Type,
       source.slug
FROM public.data_source AS source INNER JOIN public.country
    AS country ON source.country_id=country.id;
```

### 3.1.2 Chart settings

The main characteristics of this chart are presented in Figure 3.1, being the following:

- todo



Datasource: public.Untitled Query 2-cC-nXWZ3t

Visualization Type: Filter Box

Time Column: 0 option(s)

Time Grain: day

Time range: No filter

Filters: type

☐ Date Filter ☑ Instant Filtering

☐ Show SQL Granularity Dropdown ☐ Show SQL Time Column

☐ Show Druid Granularity Dropdown ☐ Show Druid Time Origin

Limit Selector Values: choose a column or metric

Database Type: Select [Database Type]

Figure 3.1: Superset chart creation: Settings for creating the database type filter.

## 3.2 Country Filter

Discuss what is important to see in this chart... TO DO

### 3.2.1 SQL query

```sql
--  Country and database type filters
SELECT source.name,
       country.country AS Country,
       database_type AS Type,
       source.slug
FROM public.data_source AS source INNER JOIN public.country
    AS country ON source.country_id=country.id;
```

### 3.2.2 Chart settings

TO DO

## 3.3 General - World Map

Discuss what is important to see in this chart... TO DO

### 3.3.1 SQL query

```
--    General - World Map
SELECT  name,
        slug,
        release_date,
        database_type AS Type,
        latitude,
        longitude,
        link,
        country AS Country,
        continent
FROM public.data_source AS source INNER JOIN public.country
    AS country ON source.country_id=country.id;
```

### 3.3.2 Chart settings

TO DO

## 3.4 General - Network Growth (Summary)

Discuss what is important to see in this chart... TO DO

### 3.4.1 SQL query

```
-- 108    General - Network Growth (Summary)
SELECT data.source,
       data.country AS Country,
       data.database_type AS Type,
       --cast(stratum_1 as INTEGER )*30 AS Days,
       data.release_date - cast(stratum_1 AS INTEGER) * INTERVAL
        '1 month' as Time,
       count_value                   AS count
FROM (
     SELECT source.name              AS source,
            achilles.analysis_id     AS analysis_id,
            achilles.stratum_1,
            achilles.stratum_2,
            achilles.stratum_3,
            achilles.stratum_4,
            achilles.stratum_5,
            achilles.count_value,
            country.country,
            source.database_type,
            source.release_date
     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
     ) data
WHERE analysis_id = 108;
```

### 3.4.2 Chart settings

TO DO

## 3.5 General - Network Growth by Date

Discuss what is important to see in this chart... TO DO

### 3.5.1 SQL query

```
-- 108    General - Network Growth by Date
SELECT data.source,
       data.country AS Country,
       data.database_type AS Type,
       cast(stratum_1 as Integer)*30 AS DAY,
       count_value                   AS count
FROM (
     SELECT source.name              AS source,
            achilles.analysis_id     AS analysis_id,
            achilles.stratum_1,
            achilles.stratum_2,
            achilles.stratum_3,
            achilles.stratum_4,
            achilles.stratum_5,
            achilles.count_value,
            country.country,
            source.database_type
     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
     ) data
WHERE analysis_id = 108;
```

### 3.5.2 Chart settings

TO DO

## 3.6 General - Patients per Country

Discuss what is important to see in this chart... TO DO

### 3.6.1 SQL query

```
-- 1    General - Patients per Country
SELECT source.name,
       country.country AS Country,
       source.database_type AS Type,
       count_value AS patient_count,
       source.slug
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;
```

### 3.6.2 Chart settings

TO DO

## 3.7 General - Database Types per Country

Discuss what is important to see in this chart... TO DO

### 3.7.1 SQL query

```
-- 1    General - Database types per Country
SELECT source.name,
       country.country AS Country,
       database_type AS Type,
       count_value AS "Nr_patients",
       source.slug
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;
```

### 3.7.2 Chart settings

TO DO

# Chapter 4

# Person

Discuss the goal of this dashboard... TO DO Reuses the ‘’Database Type Filter” CSS Template change

## 4.1 Person - Patients by age

Discuss what is important to see in this chart... TO DO

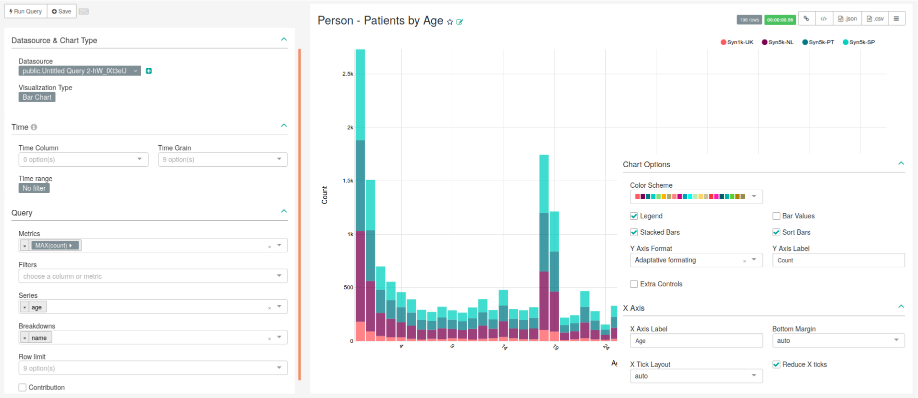
### 4.1.1 SQL query

```sql
-- 101  Person - Patients by age
SELECT source.name,
    cast(stratum_1 as int) as Age,
    count_value as count,
    source.slug
FROM public.achilles_results AS achilles INNER JOIN
    public.data_source AS source ON achilles.data_source_id=source.id
WHERE analysis_id = 101;
```

### 4.1.2 Chart settings

The main characteristics of this chart are presented in Figure 4.1, being the following:

- todo



Datasource & Chart Type

Datasource: public.Untitled Query 2-hW_lXt3eU

Visualization Type: Bar Chart

Time

Time Column: 0 option(s)

Time Grain: 9 option(s)

Time range: No filter

Query

Metrics: MAX(count)

Filters: choose a column or metric

Series: age

Breakdowns: name

Row limit: 9 option(s)

□ Contribution

Chart Options

Color Scheme

☑ Legend □ Bar Values

☑ Stacked Bars ☑ Sort Bars

Y Axis Format: Adaptative formating

Y Axis Label: Count

□ Extra Controls

X Axis

X Axis Label: Age

Bottom Margin: auto

X Tick Layout: auto

☑ Reduce X ticks

Figure 4.1: Settings for creating chart representing patient by age (bar chart). Image changed to contain information hidden in the customize menu.

## 4.2 Person - Births by Year

Discuss what is important to see in this chart... TO DO

### 4.2.1 SQL query

```
-- 3  Person - Births by year
SELECT source.name,
       stratum_1 AS "Birth_year",
       count_value AS count,
       source.slug
FROM public.achilles_results AS achilles INNER JOIN
    public.data_source AS source ON achilles.data_source_id=source.id
WHERE analysis_id = 3;
```

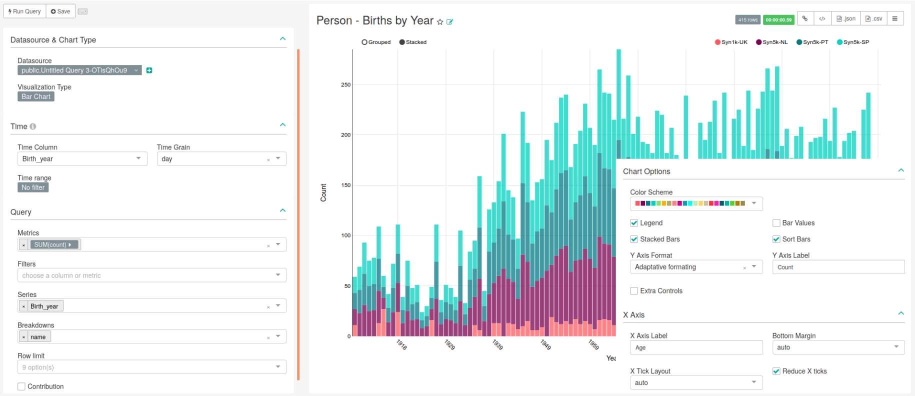
### 4.2.2 Chart settings

The main characteristics of this chart are presented in Figure 4.2, being the following:

- todo

## 4.3 Person - Gender Distribution

Discuss what is important to see in this chart... TO DO



Datasource & Chart Type

Datasource: public.Untitled Query 3-OTisQhOu9

Visualization Type: Bar Chart

Time

Time Column: Birth_year

Time Grain: day

Time range: No filter

Query

Metrics: SUM(count)

Filters: choose a column or metric

Series: Birth_year

Breakdowns: name

Row limit: 9 option(s)

□ Contribution

Chart Options

Color Scheme

☑ Legend

□ Bar Values

☑ Stacked Bars

☑ Sort Bars

Y Axis Format: Adaptative formating

Y Axis Label: Count

□ Extra Controls

X Axis

X Axis Label: Age

Bottom Margin: auto

X Tick Layout: auto

☑ Reduce X ticks

Figure 4.2: Settings for creating chart representing births by year (bar chart). Image changed to contain information hidden in the customize menu.

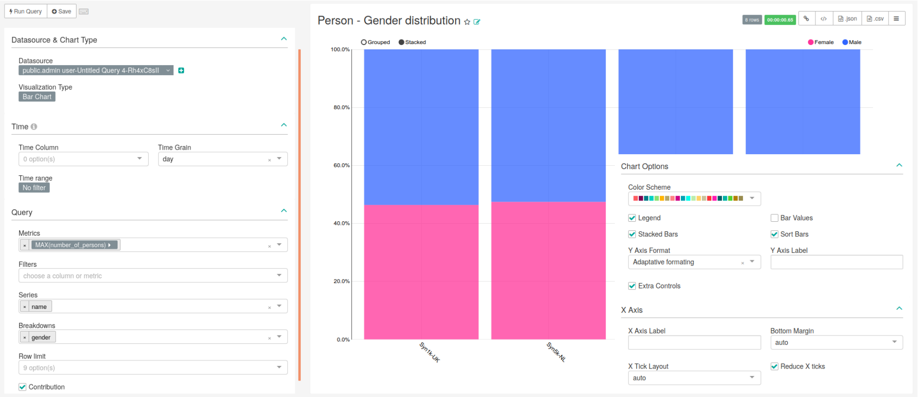
### 4.3.1 SQL query

```
-- 2  Person - Gender distribution
SELECT source.name,
       concept_name AS Gender,
       count_value AS Number_of_persons,
       source.slug
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;
```

### 4.3.2 Chart settings

The main characteristics of this chart are presented in Figure 4.3, being the following:

- todo



Datasource & Chart Type

Datasource: public.admin user-Untitled Query 4-Rh4xC8sII

Visualization Type: Bar Chart

Time

Time Column: 0 option(s)

Time Grain: day

Time range: No filter

Query

Metrics: MAX(number_of_persons)

Filters: choose a column or metric

Series: name

Breakdowns: gender

Row limit: 9 option(s)

☑ Contribution

Chart Options

Color Scheme

☑ Legend

□ Bar Values

☑ Stacked Bars

☑ Sort Bars

Y Axis Format: Adaptative formating

Y Axis Label:

☑ Extra Controls

X Axis

X Axis Label:

Bottom Margin: auto

X Tick Layout: auto

☑ Reduce X ticks

Figure 4.3: Settings for creating chart representing the gender distribution (bar chart). Image changed to contain information hidden in the customize menu.

# Chapter 5

# Population Characteristics

Discuss the goal of this dashboard... TO DO Reuses the ‘’Database Type Filter”

## 5.1 Population characteristics - Patients in Observation Period per month (whole month)

Discuss what is important to see in this chart... TO DO

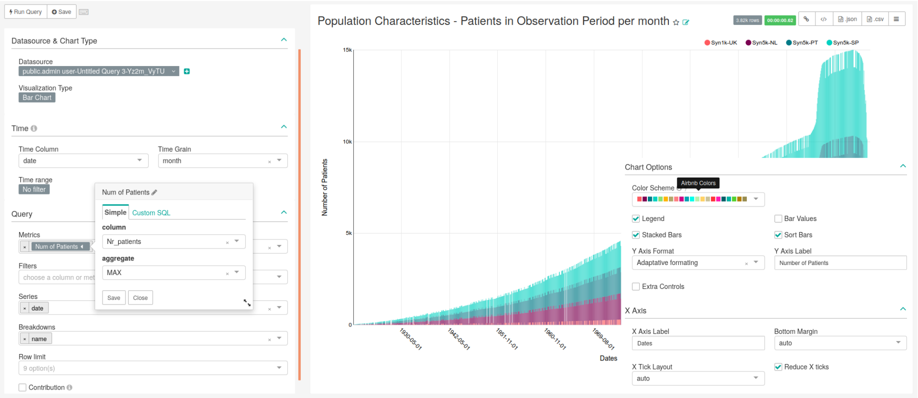
### 5.1.1 SQL query

```sql
-- 110    Population characteristics - Patients in Observation
    Period per month (whole month)
SELECT source.name,
       to_date(stratum_1, 'YYYYMM') as Date,
       count_value as "Nr_patients",
       source.slug
FROM public.achilles_results AS achilles INNER JOIN
    public.data_source AS source ON achilles.data_source_id=source.id
WHERE analysis_id = 110;
```

### 5.1.2 Chart settings

The main characteristics of this chart are presented in Figure 5.1, being the following:

- todo



Datasource & Chart Type

Datasource: public.admin user-Untitled Query 3-Yz2m_VyTU

Visualization Type: Bar Chart

Time

Time Column: date

Time Grain: month

Time range: No filter

Query

Metrics: Num of Patients

Num of Patients — column: Nr_patients; aggregate: MAX

Filters: choose a column or met

Series: date

Breakdowns: name

Row limit: 9 option(s)

☐ Contribution

Chart Options

Color Scheme

☑ Legend

☐ Bar Values

☑ Stacked Bars

☑ Sort Bars

Y Axis Format: Adaptative formating

Y Axis Label: Number of Patients

☐ Extra Controls

X Axis

X Axis Label: Dates

Bottom Margin: auto

X Tick Layout: auto

☑ Reduce X ticks

Figure 5.1: Settings for creating chart representing patient in observation per month (bar chart). Image changed to contain information hidden in the customize menu.

## 5.2 Population characteristics - Observation Period End Dates

Discuss what is important to see in this chart... TO DO

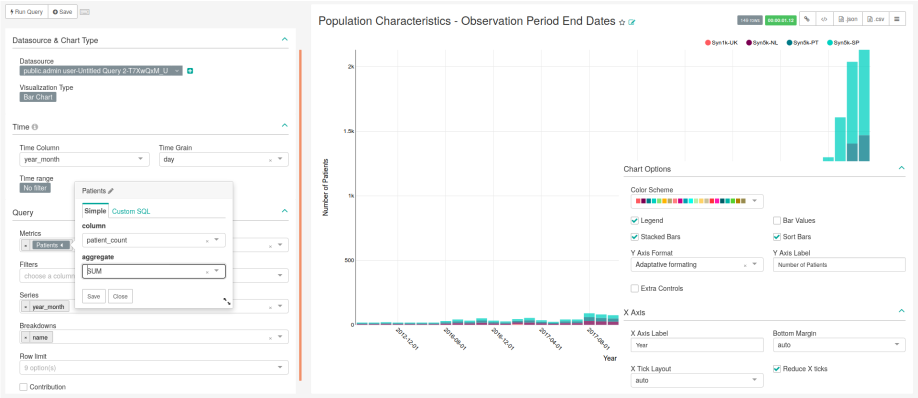
### 5.2.1 SQL query

```
-- 112  Population characteristics - Observation Period End Dates
SELECT source.name,
       to_date(stratum_1, 'YYYYMM') as year_month,
       count_value as patient_count
FROM public.achilles_results AS achilles INNER JOIN
    public.data_source AS source ON achilles.data_source_id=source.id
where analysis_id = 112;
```

### 5.2.2 Chart settings

The main characteristics of this chart are presented in Figure 5.2, being the following:

- todo



Datasource & Chart Type

Datasource: public.admin user-Untitled Query 2-T7XwQxM_U

Visualization Type: Bar Chart

Time

Time Column: year_month

Time Grain: day

Time range: No filter

Query

Metrics: Patients

Patients — Simple | Custom SQL

column: patient_count

aggregate: SUM

Save | Close

Filters: choose a column

Series: year_month

Breakdowns: name

Row limit: 9 option(s)

□ Contribution

Chart Options

Color Scheme

☑ Legend

□ Bar Values

☑ Stacked Bars

☑ Sort Bars

Y Axis Format: Adaptative formating

Y Axis Label: Number of Patients

□ Extra Controls

X Axis

X Axis Label: Year

Bottom Margin: auto

X Tick Layout: auto

☑ Reduce X ticks

Figure 5.2: Settings for creating chart representing the number of patients at the end of their observation period (bar chart). Image changed to contain information hidden in the customize menu.

## 5.3 Population characteristics - Observation Period Start Dates

Discuss what is important to see in this chart... TO DO

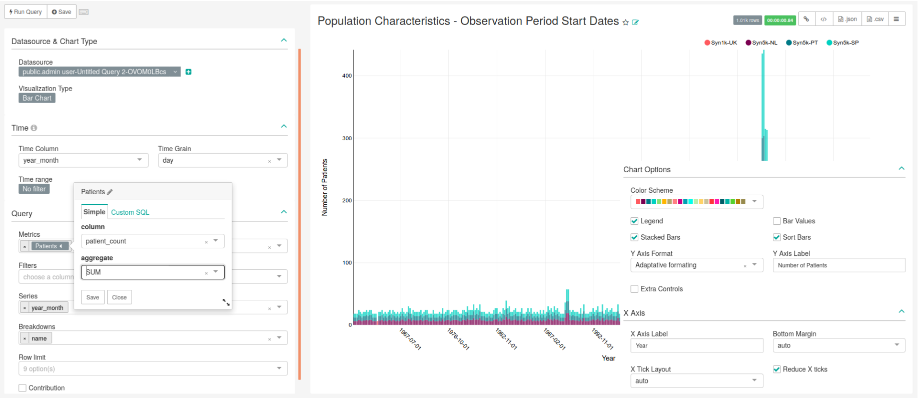
### 5.3.1 SQL query

```
-- 111  Population characteristics - Observation Period Start Dates
SELECT source.name,
       to_date(stratum_1, 'YYYYMM') as year_month,
       count_value as patient_count
FROM public.achilles_results AS achilles INNER JOIN
    public.data_source AS source ON achilles.data_source_id=source.id
where analysis_id = 111;
```

### 5.3.2 Chart settings

The main characteristics of this chart are presented in Figure 5.3, being the following:

- todo



Datasource & Chart Type

Datasource: public.admin user-Untitled Query 2-OVOM0LBcs

Visualization Type: Bar Chart

Time

Time Column: year_month

Time Grain: day

Time range: No filter

Query

Metrics: Patients

Patients

Simple | Custom SQL

column: patient_count

aggregate: SUM

Save | Close

Filters: choose a column

Series: year_month

Breakdowns: name

Row limit: 9 option(s)

□ Contribution

Chart Options

Color Scheme

☑ Legend

□ Bar Values

☑ Stacked Bars

☑ Sort Bars

Y Axis Format: Adaptative formating

Y Axis Label: Number of Patients

□ Extra Controls

X Axis

X Axis Label: Year

Bottom Margin: auto

X Tick Layout: auto

☑ Reduce X ticks

Figure 5.3: Settings for creating chart representing the number of patients at the start of their observation period (bar chart). Image changed to contain information hidden in the customize menu.

# Chapter 6

# Visit

## 6.1 Visit - Types

Discuss what is important to see in this chart... TO DO

### 6.1.1 SQL query

```sql
-- 201  Visit - Types
SELECT source.name,
       concept_name AS "Observation",
       count_value AS "Nr_Observations",
       source.slug
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 stratum_1 = CAST(concept_id AS text)
WHERE analysis_id = 201;
```

# Chapter 7

# Death

## 7.1 Death - By Month per Thousand People

Discuss what is important to see in this chart… TO DO

### 7.1.1 SQL query

```
-- 502  Death - By Month per Thousand People
SELECT source.name,
    to_date(stratum_1, 'YYYYMM') as Date,
    count_value as count,
    source.slug
FROM public.achilles_results AS achilles INNER JOIN
    public.data_source AS source ON achilles.data_source_id=source.id
WHERE analysis_id = 502;
```

## 7.2 Death - Number of Records

Discuss what is important to see in this chart… TO DO

### 7.2.1 SQL query

```
-- 501  Death - Number of Records
SELECT source.name,
    count_value as count,
    source.slug
FROM public.achilles_results AS achilles INNER JOIN
    public.data_source AS source ON achilles.data_source_id=source.id
WHERE analysis_id = 501;
```

# Chapter 8

# Concepts

Discuss the organization of this dashboard One tab uses the same query

## 8.1 Concepts General Tab

### 8.1.1 SQL query

```
SELECT
    q1.concept_id AS concept_id,
    q1.concept_name AS concept_name,
    q1.domain_id,
    source.name,
    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 q1
    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;
```

### 8.1.2 Charts

Here are all the charts presented in this dashboard using the previous query

#### 8.1.2.1 Concept Browser Table

#### 8.1.2.2 # of Occurrences

#### 8.1.2.3 # of Patients

#### 8.1.2.4 Entity Type Filter

#### 8.1.2.5 Concept Filter

#### 8.1.2.6 Number of Concepts

## 8.2 Concepts Domains Tab

Discuss what is important to see in this chart... TO DO

### 8.2.1 SQL query

```
-- Concepts Domains
SELECT source.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 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 stratum_2 = CAST(concept_id AS
    text)
WHERE analysis_id IN (405, 605, 705, 805, 1805, 2105)
GROUP BY source.name, 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
```

# Chapter 9

# Data Domains

## 9.1 Data Domains - Number of Records per Peson

Discuss what is important to see in this chart... TO DO

### 9.1.1 SQL query

```
-- 201, 401, 501, 601, 701, 801, 1801, 2101, 2201   Data domains -
    Number of records per peson
SELECT
    source.name,
    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",
    source.slug
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.slug
HAVING analysis_id IN (201, 401, 501, 601, 701, 801, 1801, 2101,
    2201)
```