What this tutorial will provide . . .

- Suggested process for developing a CDM ETL
- OHDSI ETL tools: White Rabbit, Rabbit-In-A-Hat, and Usagi
- Resources like the CDM Wiki and THEMIS
- Generation of a simple ETL
<table>
<thead>
<tr>
<th>Time</th>
<th>Agenda Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00 – 9:30</td>
<td>Overview</td>
</tr>
<tr>
<td></td>
<td>• VM Set Up</td>
</tr>
<tr>
<td></td>
<td>• What is OHDSI / CDM?</td>
</tr>
<tr>
<td></td>
<td>• What is the ETL Process?</td>
</tr>
<tr>
<td>9:30 – 10:45</td>
<td>ETL Step 1 – Design Your ETL</td>
</tr>
<tr>
<td>10:45 – 11:15</td>
<td>Coffee</td>
</tr>
<tr>
<td>11:15 – 12:30</td>
<td>ETL Step 2 – Mapping to the Vocabulary</td>
</tr>
<tr>
<td>12:30 – 13:30</td>
<td>Lunch</td>
</tr>
<tr>
<td>13:30 – 14:30</td>
<td>ETL Step 3 – Develop ETL</td>
</tr>
<tr>
<td>14:30 – 15:30</td>
<td>ETL Step 4 – Quality Control</td>
</tr>
<tr>
<td>15:30 – 16:00</td>
<td>Coffee Break</td>
</tr>
<tr>
<td>16:00 – 17:00</td>
<td>ETL Pain Points &amp; Conclusions</td>
</tr>
<tr>
<td>17:00 – 18:00</td>
<td>Beer, Wine, &amp; Snacks</td>
</tr>
</tbody>
</table>
## Instructors

<table>
<thead>
<tr>
<th>Clair Blacketer</th>
<th>Mui van Zandt</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Clair Blacketer" /></td>
<td><img src="image2.png" alt="Mui van Zandt" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Erica Voss</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3.png" alt="Erica Voss" /></td>
</tr>
</tbody>
</table>
Ground Rules

• We are recording this presentation for future use

• We may take some questions off-line if too specific
OHDSI in a Box

- PostgreSQL
  - cdm
  - webapi
- PGAdmin4
- EC2
  - Atlas
  - WebAPI
  - Tomcat
- Methods Library
  - OHDSI R packages
  - Studio
- WhiteRabbit
- Raw Lauren
- CDM Lauren (EMPTY)
- Raw Synthea
- CDM Synthea
- CDM Synpuf (100K)
- Usagi
Directions for Accessing a VM

LINK TBD

• Pick one of the rows and put your name on the second column

• Go to c:/windows/system32 and click mstc.exe

• Username: erasmusmc
• Password: 123beter
OHDSI’s Mission & Vision

To improve health by empowering a community to collaboratively generate the evidence that promotes better health decisions and better care.

A world in which observational research produces a comprehensive understanding of health and disease.

Join us on the journey

http://ohdsi.org
Current Approach: “One Study – One Script”

"What's the adherence to my drug in the data assets I own?"

Analytical method: Adherence to Drug

Application to data

Current solution:
One SAS or R script for each study

- Not scalable
- Not transparent
- Expensive
- Slow
- Prohibitive to non-expert routine use
Solution: Data Standardization Enables Systematic Research

OHDSI Tools

OMOP CDM

Adherence

Mortality

Source of Business

Safety Signals

North America Southeast Asia

China

Europe

UK

Japan

India

So Africa

Switzerland

Italy

Israel

Standardized data
CDM Version 6 Key Domains

Standardized clinical data
- Person
  - Observation_period
  - Visit_occurrence
    - Visit_detail
  - Condition_occurrence
  - Drug_exposure
  - Procedure_occurrence
  - Device_exposure
  - Measurement
  - Note
    - Note_NLP
  - Survey_conduct
  - Observation
  - Specimen
    - Fact_relationship

Standardized health system data
- Location
  - Location_history
  - Care_site
  - Provider

Standardized derived elements
- Condition_era
- Drug_era
- Dose_era

Results Schema
- Cohort
  - Cohort_definition

Standardized health economics
- Cost
  - Payer_plan_period

Standardized metadata
- CDM_source
- Metadata

Standardized vocabularies
- Concept
- Vocabulary
- Domain
- Concept_class
- Concept_relationship
- Relationship
- Concept_synonym
- Concept_ancestor
- Source_to_concept_map
- Drug_strength
Why the CDM?

Ability to pursue **cross-institutional collaborations**

Write **one program** to run on multiple data assets

**OMOP Vocabularies** has greatly increased our **ability to find relevant codes**

You truly **know your data** if you convert it to the CDM

If you know a problem with your data, you can use the **ETL to address it**

**Whole community of researchers** across diverse organizations and countries

You can use **standardized tools** developed by OHDSI like ATLAS and the Patient Level Prediction Package

The CDM brings **consistency** to observational research through standardization of many of its components

**Buy vs Build:** leverage an entire community of technical and scientific capability for **“free”**

Takes observational research towards **open science**
Why the CDM?

<table>
<thead>
<tr>
<th>Regulator</th>
<th>Data Source Owners</th>
<th>Small to Medium Enterprises</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Increased capacity to carry out <strong>studies with big geographical coverage</strong></td>
<td>• Facilitates <strong>scientific collaboration</strong> by becoming part of a thriving network</td>
<td>• Opportunity to change paradigm benefitting health of citizens</td>
</tr>
<tr>
<td>• Increased capacity to look at patients <strong>holistically</strong> across health systems</td>
<td>• <strong>Increased analysis capability</strong> thanks to a host of open source tools to use</td>
<td>• Expand your existing market</td>
</tr>
<tr>
<td>• Easier assessment of <strong>data quality</strong></td>
<td>• <strong>Faster performing studies</strong>, more studies in less time.</td>
<td>• Open source community boosts opportunities</td>
</tr>
</tbody>
</table>
• Extract, Transform, Load

• In order to get from our native/raw data into the OMOP CDM we need to design and develop and ETL process

• Goal in ETLing is standardize the format and terminology

• This tutorial
  – Will teach you best practices around designing an ETL and CDM maintenance
  – Will not teach you how to program an ETL
Data experts and CDM experts together design the ETL.

People with medical knowledge create the code mappings.

All are involved in quality control.

A technical person implements the ETL.

OHDSI Tools:
- White Rabbit
- Rabbit In a Hat
- Usagi
- White Rabbit
- ACHILLES
- Rabbit In a Hat
ETL Process

ETL creation best practices

- CDM Conversion Best Practices

This document describes some of the best practices we have developed over the years when trying to create an ETL (Extract, Transform, Load) process to convert data into the OMOP Common Data Model (CDM). We have found it best to split the process into four distinct activities:

1. Data experts and CDM experts together design the ETL
2. People with medical knowledge create the code mappings
3. A technical person implements the ETL
4. All are involved in quality control

1. Data experts and CDM experts together design the ETL

Designing the ETL requires in-depth knowledge of the source data, but it also requires knowledge of the CDM, and having someone with experience in past ETLs to the OMOP CDM can speed up the design activity. Ideally, the data and CDM experts should sit down together at the same location in a one- or two-day session.

We have developed two tools that have proven to be helpful for this activity: White Rabbit and Rabbit-in-a-Hat.
Hands On Exercises for Today

• Scan a database with White Rabbit

• Build a ETL document with Rabbit in a Hat

• Mapping Source Codes by with the OMOP Vocabulary and USAGI

• SQL to build an ETL
Data experts and CDM experts together design the ETL

People with medical knowledge create the code mappings

All are involved in quality control

A technical person implements the ETL
A Patient’s Story: Lauren

Lauren's story

“Every step of this painful journey I’ve had to convince everyone how much pain I was in.”

“My first surgery taught me that I had to be very patient with my recovery and very patient with myself in general.”

https://www.endometriosis-uk.org/laurens-story
Lauren’s Timeline

-3 Years -2 Years -1 Years / / -2 Weeks / / -3 Days / / Day 0

Endometriosis

dysmenorrhea
abdominal pain
missed work
acetaminophen
acetaminophen
acetaminophen
GP visit
pelvic exam
ultrasound
cyst of ovary

Hospital Visit

severe pain
temp 103°F
CT Scan
ambulance
Bloated abdomen
ascites
surgery
endometrioma

What data do we have?

-3 Years
-2 Years
-1 Years

missed work
acetaminophen
acetaminophen
acetaminophen

-2 Weeks

GP visit
pelvic exam
ultrasound
cyst of ovary

-3 Days

CT Scan

Day 0

surgery
Data Format

- **Synthea™** is a Synthetic Patient Population Simulator. The goal is to output synthetic, realistic (but not real), patient data and associated health records in a variety of formats.

- The resulting data is free from cost, privacy, and security restrictions. It can be used without restriction for a variety of secondary uses in academia, research, industry, and government (although a citation would be appreciated).

- [https://github.com/synthetichealth/synthea](https://github.com/synthetichealth/synthea)

## Synthea Tables

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>allergies.csv</code></td>
<td>Patient allergy data.</td>
</tr>
<tr>
<td><code>careplans.csv</code></td>
<td>Patient care plan data, including goals.</td>
</tr>
<tr>
<td><code>conditions.csv</code></td>
<td>Patient conditions or diagnoses.</td>
</tr>
<tr>
<td><code>encounters.csv</code></td>
<td>Patient encounter data.</td>
</tr>
<tr>
<td><code>imaging_studies.csv</code></td>
<td>Patient imaging metadata.</td>
</tr>
<tr>
<td><code>immunizations.csv</code></td>
<td>Patient immunization data.</td>
</tr>
<tr>
<td><code>medications.csv</code></td>
<td>Patient medication data.</td>
</tr>
<tr>
<td><code>observations.csv</code></td>
<td>Patient observations including vital signs and lab reports.</td>
</tr>
<tr>
<td><code>organizations.csv</code></td>
<td>Provider organizations including hospitals.</td>
</tr>
<tr>
<td><code>patients.csv</code></td>
<td>Patient demographic data.</td>
</tr>
<tr>
<td><code>procedures.csv</code></td>
<td>Patient procedure data including surgeries.</td>
</tr>
<tr>
<td><code>providers</code></td>
<td>Clinicians that provide patient care.</td>
</tr>
</tbody>
</table>
Raw Data

1 Patient
Lauren Data
Synthea Format

1000 Patient
Synthetic Data
Synthea Format
Tools help us get started . . .

**White Rabbit**
- performs a scan of the source data, providing detailed information on the tables, fields, and values that appear in a field

**Rabbit In a Hat**
- Uses White Rabbit scan to provide a graphical user interface to help build an ETL document
- Does not generate code
White Rabbit - Location
White Rabbit - Scan
White Rabbit - Scan
White Rabbit - Scan

<table>
<thead>
<tr>
<th>Min cell count</th>
<th>Max distinct values</th>
<th>Rows per table</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1,000</td>
<td>100,000</td>
</tr>
</tbody>
</table>
White Rabbit – Scan Report

- We already ran the scan on raw_synthea

- To open the scan while we review:
  - [https://github.com/OHDSI/Tutorial-ETL](https://github.com/OHDSI/Tutorial-ETL)
  - Materials → WhiteRabbit → ScanReport_raw_synthea.xlsx
  - Click “View Raw” to download the XLSX
### White Rabbit – Scan Report: raw_synthea

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Max length</th>
<th>N rows</th>
<th>N rows</th>
<th>Fraction empty</th>
</tr>
</thead>
<tbody>
<tr>
<td>allergies start</td>
<td>date</td>
<td>10</td>
<td>619</td>
<td>619</td>
<td>0</td>
</tr>
<tr>
<td>allergies stop</td>
<td>date</td>
<td>10</td>
<td>619</td>
<td>619</td>
<td>0.904685</td>
</tr>
<tr>
<td>allergies patient</td>
<td>character</td>
<td>36</td>
<td>619</td>
<td>619</td>
<td>0</td>
</tr>
<tr>
<td>allergies encounter</td>
<td>character</td>
<td>36</td>
<td>619</td>
<td>619</td>
<td>0</td>
</tr>
<tr>
<td>allergies code</td>
<td>character</td>
<td>9</td>
<td>619</td>
<td>619</td>
<td>0</td>
</tr>
<tr>
<td>allergies description</td>
<td>character</td>
<td>24</td>
<td>619</td>
<td>619</td>
<td>0</td>
</tr>
<tr>
<td>careplans id</td>
<td>character</td>
<td>36</td>
<td>2939</td>
<td>2939</td>
<td>0</td>
</tr>
<tr>
<td>careplans start</td>
<td>date</td>
<td>10</td>
<td>2939</td>
<td>2939</td>
<td>0.380061</td>
</tr>
<tr>
<td>careplans stop</td>
<td>date</td>
<td>10</td>
<td>2939</td>
<td>2939</td>
<td>0</td>
</tr>
<tr>
<td>careplans patient</td>
<td>character</td>
<td>36</td>
<td>2939</td>
<td>2939</td>
<td>0</td>
</tr>
<tr>
<td>careplans encounter</td>
<td>character</td>
<td>36</td>
<td>2939</td>
<td>2939</td>
<td>0</td>
</tr>
<tr>
<td>careplans code</td>
<td>character</td>
<td>15</td>
<td>2939</td>
<td>2939</td>
<td>0</td>
</tr>
<tr>
<td>careplans description</td>
<td>character</td>
<td>62</td>
<td>2939</td>
<td>2939</td>
<td>0.090507</td>
</tr>
<tr>
<td>careplans reason_cc</td>
<td>character</td>
<td>14</td>
<td>2939</td>
<td>2939</td>
<td>0.090507</td>
</tr>
<tr>
<td>careplans reason_dc</td>
<td>character</td>
<td>69</td>
<td>2939</td>
<td>2939</td>
<td>0</td>
</tr>
<tr>
<td>condition start</td>
<td>date</td>
<td>10</td>
<td>7898</td>
<td>7898</td>
<td>0</td>
</tr>
<tr>
<td>condition stop</td>
<td>date</td>
<td>10</td>
<td>7898</td>
<td>7898</td>
<td>0.458091</td>
</tr>
<tr>
<td>condition patient</td>
<td>character</td>
<td>36</td>
<td>7898</td>
<td>7898</td>
<td>0</td>
</tr>
<tr>
<td>condition encounter</td>
<td>character</td>
<td>36</td>
<td>7898</td>
<td>7898</td>
<td>0</td>
</tr>
<tr>
<td>condition code</td>
<td>character</td>
<td>15</td>
<td>7898</td>
<td>7898</td>
<td>0</td>
</tr>
<tr>
<td>condition description</td>
<td>character</td>
<td>80</td>
<td>7898</td>
<td>7898</td>
<td>0</td>
</tr>
<tr>
<td>encounter id</td>
<td>character</td>
<td>36</td>
<td>34262</td>
<td>34262</td>
<td>0</td>
</tr>
<tr>
<td>encounter start</td>
<td>date</td>
<td>10</td>
<td>34262</td>
<td>34262</td>
<td>0</td>
</tr>
<tr>
<td>encounter stop</td>
<td>date</td>
<td>10</td>
<td>34262</td>
<td>34262</td>
<td>0</td>
</tr>
<tr>
<td>encounter patient</td>
<td>character</td>
<td>36</td>
<td>34262</td>
<td>34262</td>
<td>0</td>
</tr>
<tr>
<td>encounter provider</td>
<td>character</td>
<td>36</td>
<td>34262</td>
<td>34262</td>
<td>0</td>
</tr>
<tr>
<td>encounter encounter</td>
<td>character</td>
<td>36</td>
<td>34262</td>
<td>34262</td>
<td>0</td>
</tr>
<tr>
<td>encounter code</td>
<td>character</td>
<td>9</td>
<td>34262</td>
<td>34262</td>
<td>0</td>
</tr>
<tr>
<td>encounter description</td>
<td>character</td>
<td>59</td>
<td>34262</td>
<td>34262</td>
<td>0</td>
</tr>
<tr>
<td>encounter cost</td>
<td>numeric</td>
<td>6</td>
<td>34262</td>
<td>34262</td>
<td>0</td>
</tr>
<tr>
<td>encounter reason_cc</td>
<td>character</td>
<td>15</td>
<td>34262</td>
<td>34262</td>
<td>0.661155</td>
</tr>
<tr>
<td>encounter reason_dc</td>
<td>character</td>
<td>69</td>
<td>34262</td>
<td>34262</td>
<td>0.661155</td>
</tr>
<tr>
<td>imaging_s_id</td>
<td>character</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>imaging_s_date</td>
<td>date</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### White Rabbit – Scan Report: raw_synthea

<table>
<thead>
<tr>
<th>patients</th>
<th>id</th>
<th>character varying</th>
<th>36</th>
<th>1120</th>
<th>1120</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>patients</td>
<td>birthdate</td>
<td>date</td>
<td>10</td>
<td>1120</td>
<td>1120</td>
<td>0</td>
</tr>
<tr>
<td>patients</td>
<td>deathdate</td>
<td>date</td>
<td>10</td>
<td>1120</td>
<td>1120</td>
<td>0</td>
</tr>
<tr>
<td>patients</td>
<td>ssn</td>
<td>character varying</td>
<td>11</td>
<td>1120</td>
<td>1120</td>
<td>0</td>
</tr>
<tr>
<td>patients</td>
<td>drivers</td>
<td>character varying</td>
<td>9</td>
<td>1120</td>
<td>1120</td>
<td>0</td>
</tr>
<tr>
<td>patients</td>
<td>passport</td>
<td>character varying</td>
<td>10</td>
<td>1120</td>
<td>1120</td>
<td>0</td>
</tr>
<tr>
<td>patients</td>
<td>prefix</td>
<td>character varying</td>
<td>4</td>
<td>1120</td>
<td>1120</td>
<td>0</td>
</tr>
<tr>
<td>patients</td>
<td>first</td>
<td>character varying</td>
<td>15</td>
<td>1120</td>
<td>1120</td>
<td>0</td>
</tr>
<tr>
<td>patients</td>
<td>last</td>
<td>character varying</td>
<td>16</td>
<td>1120</td>
<td>1120</td>
<td>0</td>
</tr>
<tr>
<td>patients</td>
<td>suffix</td>
<td>character varying</td>
<td>3</td>
<td>1120</td>
<td>1120</td>
<td>0</td>
</tr>
<tr>
<td>patients</td>
<td>maiden</td>
<td>character varying</td>
<td>16</td>
<td>1120</td>
<td>1120</td>
<td>0</td>
</tr>
<tr>
<td>patients</td>
<td>marital</td>
<td>character varying</td>
<td>1</td>
<td>1120</td>
<td>1120</td>
<td>0</td>
</tr>
<tr>
<td>patients</td>
<td>race</td>
<td>character varying</td>
<td>8</td>
<td>1120</td>
<td>1120</td>
<td>0</td>
</tr>
<tr>
<td>patients</td>
<td>ethnicity</td>
<td>character varying</td>
<td>16</td>
<td>1120</td>
<td>1120</td>
<td>0</td>
</tr>
<tr>
<td>patients</td>
<td>gender</td>
<td>character varying</td>
<td>1</td>
<td>1120</td>
<td>1120</td>
<td>0</td>
</tr>
<tr>
<td>patients</td>
<td>birthplace</td>
<td>character varying</td>
<td>21</td>
<td>1120</td>
<td>1120</td>
<td>0</td>
</tr>
<tr>
<td>patients</td>
<td>address</td>
<td>character varying</td>
<td>36</td>
<td>1120</td>
<td>1120</td>
<td>0</td>
</tr>
<tr>
<td>patients</td>
<td>city</td>
<td>character varying</td>
<td>21</td>
<td>1120</td>
<td>1120</td>
<td>0</td>
</tr>
<tr>
<td>patients</td>
<td>state</td>
<td>character varying</td>
<td>13</td>
<td>1120</td>
<td>1120</td>
<td>0</td>
</tr>
<tr>
<td>patients</td>
<td>zip</td>
<td>character varying</td>
<td>5</td>
<td>1120</td>
<td>1120</td>
<td>0</td>
</tr>
</tbody>
</table>

### Overview Tab
<table>
<thead>
<tr>
<th>Id</th>
<th>Frequency</th>
<th>birthdate</th>
<th>Frequency</th>
<th>deathdate</th>
<th>Frequency</th>
<th>ssn</th>
<th>Frequency</th>
<th>drivers</th>
<th>Frequency</th>
<th>passport</th>
<th>Frequency</th>
<th>prefix</th>
<th>Frequency</th>
<th>first</th>
<th>Frequency</th>
<th>last</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>List trunc</td>
<td>List trunc</td>
<td>List trunc</td>
<td>List trunc</td>
<td>List trunc</td>
<td>List trunc</td>
<td>List trunc</td>
<td>List trunc</td>
<td>List trunc</td>
<td>List trunc</td>
<td>List trunc</td>
<td>List trunc</td>
<td>List trunc</td>
<td>List trunc</td>
<td>List trunc</td>
<td>List trunc</td>
<td>List trunc</td>
<td></td>
</tr>
<tr>
<td>1908-08-11</td>
<td>11</td>
<td>1000</td>
<td>List trunc</td>
<td>List trunc</td>
<td>186</td>
<td>235</td>
<td>Mr.</td>
<td>453</td>
<td>List trunc</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1910-09-22</td>
<td>11</td>
<td>List trunc</td>
<td>List trunc</td>
<td>List trunc</td>
<td>List trunc</td>
<td>List trunc</td>
<td>List trunc</td>
<td>List trunc</td>
<td>List trunc</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1914-10-06</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1920-10-03</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1929-07-01</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1913-05-01</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1922-08-11</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Patients Tab**
Now Your Turn:
Scan Lauren’s Data

- Click on WhiteRabbit shortcut
- Go into the WhiteRabbit folder
- Open WhiteRabbit.jar
Now Your Turn: Scan Lauren’s Data

- Connect to Lauren’s Data

  ![Database Icon](raw_lauren)

  **Source data location**
  - Data type: PostgreSQL
  - Server location: localhost/ETL
  - User name: postgres
  - Password: ohdsi
  - Database name: raw_lauren

- Test connection
Now Your Turn:
Scan Lauren’s Data

• Go to the “Scan” tab
• Press “Add all in DB” button, set “Min cell count” to 0, and then “Scan tables”

• Open ScanReport.xlsx
White Rabbit

• White Rabbit creates an export of information about the source data

• The scan can be used to:
  – Learn about your source data
  – Needed for Rabbit In a Hat
Rabbit in a Hat

- Can read and display a White Rabbit scan document
- Provides a graphical interface to allow a user to connect source data to tables
Rabbit in a Hat

- We will use the ScanReport_raw_synthea.xlsx for this:
  - https://github.com/OHDSI/Tutorial-ETL
  - Materials ➔ WhiteRabbit ➔ ScanReport_raw_synthea.xlsx
  - Click “View Raw” to download the XLSX

- Save it to the desktop
- Open in Rabbit in a Hat
The scan tells Rabbit in a Hat what is in the raw database

- Orange Tables = Raw
- Blue Tables = CDM
Rabbit in a Hat

Together

- person
- observation_period
- condition_occurrence

On your Own

- drug_exposure

Generate document
Resources

• Important links to keep in mind when working on an ETL:

  – CDM Wiki
    https://github.com/OHDSI/CommonDataModel/wiki
    Information about the CDM structure and conventions to follow can be found here

  – OHDSI Forums
    http://forums.ohdsi.org/
    http://forums.ohdsi.org/c/cdm-builders
    OHDSI is an active community, your questions may have already been asked on the forum however if not do not be afraid to ask it yourself!

  – CDM Examples
    About 100 CDMs currently exist

  – THEMIS Working Group
    https://github.com/OHDSI/Themis
Rabbit in a Hat

• The full ETL document: https://ohdsi.github.io/ETL-Synthea/
Some Parting Thoughts On ETL

• Vocabulary will tell a source record where to go.
  – Example, just because it is a condition code and in a condition table does not mean it will end up in CONDITION_OCCURRENCE

  ICD9 781.1 - Abnormal weight gain

• STEM Table in Rabbit In a Hat
Data experts and CDM experts together design the ETL.

People with medical knowledge create the code mappings.

All are involved in quality control.

A technical person implements the ETL.

ETL Documentation
Standardizing Terminologies

SOURCE_CODE
XYZ
i.e. ICPC-1 Dutch codes, ICD9, etc.

STANDARD_CONCEPT_ID
123456789
i.e. SNOMED for conditions and RxNorm for drugs

• What is standardize:

  1. TABLE_CONCEPT_ID
     standard concept the source code maps to, **used for analysis**
  2. TABLE_SOURCE_CONCEPT_ID
     concept representation of the source code, **helps maintain tie to raw data**

• Ways to get a source code to standard code:

  1. OMOP Vocabulary
  2. USAGI
OMOP Vocab

• There are two standard queries to help us use the OMOP Vocabulary:
  – SOURCE_TO_STANDARD.sql
  – SOURCE_TO_SOURCE.sql

• [https://github.com/OHDSI/Tutorial-ETL](https://github.com/OHDSI/Tutorial-ETL)
  – Materials ➔ Queries
OMOP Vocab

• If your source data’s codes are in the OMOP Vocab you can use it to translate to a standard

• Synthea already speaks standard terminology:
  – Conditions = SNOMED
  – Drugs = RxNorm
  – Procedures = SNOMED
  – Observations = SNOMED
SELECT * 
FROM RAW_LAUREN.CONDITIONS 
WHERE ENCOUNTER = '70'

<table>
<thead>
<tr>
<th>START</th>
<th>STOP</th>
<th>PATIENT</th>
<th>ENCOUNTER</th>
<th>CODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/6/2010</td>
<td></td>
<td>1</td>
<td>70</td>
<td>2665990000</td>
<td>Dysmenorrhea</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CONDITION_CONCEPT_ID</th>
<th>CONDITION_SOURCE_CONCEPT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
WITH CTE_VOCAB_MAP AS (  
SELECT c.concept_code AS SOURCE_CODE, c.concept_id AS SOURCE_CONCEPT_ID, c.concept_name AS SOURCE_CODE_DESCRIPTION,  
c.vocabulary_id AS SOURCE_VOCABULARY_ID, c.domain_id AS SOURCE_DOMAIN_ID, c.concept_class_id AS SOURCE_CONCEPT_CLASS_ID,  
c.VALID_START_DATE AS SOURCE_VALID_START_DATE, c.VALID_END_DATE AS SOURCE_VALID_END_DATE, c.INVALID_REASON AS SOURCE_INVALID_REASON,  
c1.concept_id AS TARGET_CONCEPT_ID, c1.concept_name AS TARGET_CONCEPT_NAME, c1.vocabulary_id AS TARGET_VOCABULARY_ID,  
c1.domain_id AS TARGET_DOMAIN_ID, c1.concept_class_id AS TARGET_CONCEPT_CLASS_ID, c1.INVALID_REASON AS TARGET_INVALID_REASON,  
c1.standard_concept AS TARGET_STANDARD_CONCEPT  
FROM CONCEPT C  
JOIN CONCEPT_RELATIONSHIP CR  
ON C.CONCEPT_ID = CR.CONCEPT_ID_1  
AND CR.INVALID_REASON IS NULL  
AND cr.relationship_id = 'Maps to'  
JOIN CONCEPT C1  
ON CR.CONCEPT_ID_2 = C1.CONCEPT_ID  
AND C1.INVALID_REASON IS NULL  
UNION  
SELECT source_code, SOURCE_CONCEPT_ID, SOURCE_CODE_DESCRIPTION, source_vocabulary_id, c1.domain_id AS SOURCE_DOMAIN_ID,  
c2.concept_class_id AS SOURCE_CONCEPT_CLASS_ID, c1.VALID_START_DATE AS SOURCE_VALID_START_DATE,  
c1.VALID_END_DATE AS SOURCE_VALID_END_DATE, stcm.INVALID_REASON AS SOURCE_INVALID_REASON, target_concept_id,  
c2.concept_name AS TARGET_CONCEPT_NAME, target_vocabulary_id, c2.domain_id AS TARGET_DOMAIN_ID,  
c2.concept_class_id AS TARGET_CONCEPT_CLASS_ID, c2.INVALID_REASON AS TARGET_INVALID_REASON,  
c2.standard_concept AS TARGET_STANDARD_CONCEPT  
FROM source_to_concept_map stcm  
LEFT OUTER JOIN CONCEPT c1  
ON c1.concept_id = stcm.source_concept_id  
LEFT OUTER JOIN CONCEPT c2  
ON c2.concept_id = stcm.target_concept_id  
WHERE stcm.INVALID_REASON IS NULL  
)  
SELECT TARGET_CONCEPT_ID, TARGET_CONCEPT_NAME, TARGET_DOMAIN_ID  
FROM CTE_VOCAB_MAP  
WHERE SOURCE_CODE = '266599000'  
AND TARGET_STANDARD_CONCEPT = 'S'
WITH CTE_VOCAB_MAP AS (  
SELECT c.concept_code AS SOURCE_CODE, c.concept_id AS SOURCE_CONCEPT_ID, c.concept_name AS SOURCE_CODE_DESCRIPTION, c.vocabulary_id AS SOURCE_VOCABULARY_ID, c.domain_id AS SOURCE_DOMAIN_ID, c.concept_class_id AS SOURCE_CONCEPT_CLASS_ID, c.VALID_START_DATE AS SOURCE_VALID_START_DATE, c.VALID_END_DATE AS SOURCE_VALID_END_DATE, c.INVALID_REASON AS SOURCE_INVALID_REASON, c1.concept_id AS TARGET_CONCEPT_ID, c1.concept_name AS TARGET_CONCEPT_NAME, c1.vocabulary_id AS TARGET_VOCABULARY_ID, c1.domain_id AS TARGET_DOMAIN_ID, c1.concept_class_id AS TARGET_CONCEPT_CLASS_ID, c1.INVALID_REASON AS TARGET_INVALID_REASON, c1.standard_concept AS TARGET_STANDARD_CONCEPT 
FROM CONCEPT C 
JOIN CONCEPT_RELATIONSHIP CR 
ON C.CONCEPT_ID = CR.CONCEPT_ID_1 
AND CR.invalid_reason IS NULL 
AND cr.relationship_id = 'Maps to'  
JOIN CONCEPT C1 
ON CR.CONCEPT_ID_2 = C1.CONCEPT_ID 
AND C1.INVALID_REASON IS NULL  
UNION 
SELECT source_code, SOURCE_CONCEPT_ID, SOURCE_CODE_DESCRIPTION, source_vocabulary_id, c1.domain_id AS SOURCE_DOMAIN_ID, c2.concept_class_id AS SOURCE_CONCEPT_CLASS_ID, c1.VALID_START_DATE AS SOURCE_VALID_START_DATE, c1.VALID_END_DATE AS SOURCE_VALID_END_DATE, c1.INVALID_REASON AS SOURCE_INVALID_REASON, c1.standard_concept AS TARGET_STANDARD_CONCEPT 
FROM source_to_concept_map stcm 
LEFT OUTER JOIN CONCEPT c1 
ON c1.concept_id = stcm.source_concept_id 
LEFT OUTER JOIN CONCEPT c2 
ON c2.CONCEPT_ID = stcm.target_concept_id 
WHERE stcm.INVALID_REASON IS NULL 
)
SELECT TARGET_CONCEPT_ID, TARGET_CONCEPT_NAME, TARGET_DOMAIN_ID
FROM CTE_VOCAB_MAP
WHERE SOURCE_CODE = '266599000'
AND TARGET_STANDARD_CONCEPT = 'S'

Look in the OMOP Vocabulary for a map
Source to Standard

WITH CTE_VOCAB_MAP AS (  
SELECT c.concept_code AS SOURCE_CODE, c.concept_id AS SOURCE_CONCEPT_ID, c.concept_name AS SOURCE_CODE_DESCRIPTION, c.vocabulary_id AS SOURCE_VOCABULARY_ID, c.domain_id AS SOURCE_DOMAIN_ID, c.concept_class_id AS SOURCE_CONCEPT_CLASS_ID, c.VALID_START_DATE AS SOURCE_VALID_START_DATE, c.VALID_END_DATE AS SOURCE_VALID_END_DATE, c.INVALID_REASON AS SOURCE_INVALID_REASON, 
  c1.concept_id AS TARGET_CONCEPT_ID, c1.concept_name AS TARGET_CONCEPT_NAME, c1.vocabulary_id AS TARGET_VOCABULARY_ID,  
  c1.domain_id AS TARGET_DOMAIN_ID, c1.concept_class_id AS TARGET_CONCEPT_CLASS_ID, 
  c1.standard_concept AS TARGET_STANDARD_CONCEPT  
FROM CONCEPT C 
JOIN CONCEPT_RELATIONSHIP CR ON C.CONCEPT_ID = CR.CONCEPT_ID_1 
AND CR.invalid_reason IS NULL 
AND cr.relationship_id = 'Maps to' 
JOIN CONCEPT C1 ON CR.CONCEPT_ID_2 = C1.CONCEPT_ID 
AND C1.INVALID_REASON IS NULL 
UNION 
SELECT source_code, SOURCE_CONCEPT_ID, SOURCE_CODE_DESCRIPTION, source_vocabulary_id, c1.domain_id AS SOURCE_DOMAIN_ID,  
  c2.concept_class_id AS SOURCE_CONCEPT_CLASS_ID, c1.VALID_START_DATE AS SOURCE_VALID_START_DATE,  
  c1.VALID_END_DATE AS SOURCE_VALID_END_DATE, stcmINVALID_REASON AS SOURCE_INVALID_REASON,target_concept_id,  
  c2.CONCEPT_NAME AS TARGET_CONCEPT_NAME, target_vocabulary_id, c2.domain_id AS TARGET_DOMAIN_ID,  
  c2.concept_class_id AS TARGET_CONCEPT_CLASS_ID, c2.INVALID_REASON AS TARGET_INVALID_REASON,  
  c2.standard_concept AS TARGET_STANDARD_CONCEPT  
FROM source_to_concept_map stcm  
LEFT OUTER JOIN CONCEPT c1  
ON c1.concept_id = stcm.source_concept_id  
LEFT OUTER JOIN CONCEPT c2  
ON c2.concept_id = stcm.target_concept_id  
WHERE stcm.INVALID_REASON IS NULL )

SELECT TARGET_CONCEPT_ID, TARGET_CONCEPT_NAME, TARGET_DOMAIN_ID  
FROM CTE_VOCAB_MAP  
WHERE SOURCE_CODE = '266599000' 
AND TARGET_STANDARD_CONCEPT = 'S'
WITH CTE_VOCAB_MAP AS (  
SELECT c.concept_code AS SOURCE_CODE, c.concept_id AS SOURCE_CONCEPT_ID, c.concept_name AS SOURCE_CODE_DESCRIPTION, c.vocabulary_id AS SOURCE_VOCABULARY_ID, c.domain_id AS SOURCE_DOMAIN_ID, c.CONCEPT_CLASS_ID AS SOURCE_CONCEPT_CLASS_ID, c.VALID_START_DATE AS SOURCE_VALID_START_DATE, c.VALID_END_DATE AS SOURCE_VALID_END_DATE, c.INVALID_REASON AS SOURCE_INVALID_REASON, c1.concept_id AS TARGET_CONCEPT_ID, c1.concept_name AS TARGET_CONCEPT_NAME, c1.VOCABULARY_ID AS TARGET_VOCABULARY_ID, c1.domain_id AS TARGET_DOMAIN_ID, c1.concept_class_id AS TARGET_CONCEPT_CLASS_ID, c1.INVALID_REASON AS TARGET_INVALID_REASON, c1.standard_concept AS TARGET_STANDARD_CONCEPT  
FROM CONCEPT C  
JOIN CONCEPT_RELATIONSHIP CR  
ON C.CONCEPT_ID = CR.CONCEPT_ID_1  
AND CR.INVALID_REASON IS NULL  
AND cr.relationship_id = 'Maps to'  
JOIN CONCEPT C1  
ON CR.CONCEPT_ID_2 = C1.CONCEPT_ID  
AND C1.INVALID_REASON IS NULL  
UNION  
SELECT source_code AS SOURCE_CODE, source_concept_id AS SOURCE_CONCEPT_ID, source_code_description AS SOURCE_CODE_DESCRIPTION, source_vocabulary_id AS SOURCE_VOCABULARY_ID, source_domain_id AS SOURCE_DOMAIN_ID, source_valid_start_date AS SOURCE_VALID_START_DATE, source_valid_end_date AS SOURCE_VALID_END_DATE, source_invalid_reason AS SOURCE_INVALID_REASON, target_concept_id, target_vocabulary_id, target_domain_id, target_concept_class_id, target_invalid_reason, target_standard_concept  
FROM source_to_concept_map stcm  
LEFT OUTER JOIN CONCEPT c1  
ON c1.concept_id = stcm.source_concept_id  
LEFT OUTER JOIN CONCEPT c2  
ON c2.CONCEPT_ID = stcm.target_concept_id  
WHERE stcm.INVALID_REASON IS NULL  
)  
SELECT TARGET_CONCEPT_ID, TARGET_CONCEPT_NAME, TARGET_DOMAIN_ID  
FROM CTE_VOCAB_MAP  
WHERE SOURCE_CODE = '266599000'  
AND TARGET_STANDARD_CONCEPT = 'S'
Mapping a Lauren Row to CONCEPT_ID: Source to Standard

<table>
<thead>
<tr>
<th>START</th>
<th>STOP</th>
<th>PATIENT</th>
<th>ENCOUNTER</th>
<th>CODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/6/2010</td>
<td></td>
<td>1</td>
<td>70</td>
<td>266599000</td>
<td>Dysmenorrhea</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TARGET_CONCEPT_ID</th>
<th>TARGET_CONCEPT_NAME</th>
<th>TARGET_DOMAIN_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>194696</td>
<td>Dysmenorrhea</td>
<td>Condition</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CONDITION_CONCEPT_ID</th>
<th>CONDITION_SOURCE_CONCEPT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>194696</td>
<td></td>
</tr>
</tbody>
</table>
WITH CTE_VOCAB_MAP AS (  
    SELECT c.concept_code AS SOURCE_CODE, c.concept_id AS SOURCE_CONCEPT_ID,  
           c.CONCEPT_NAME AS SOURCE_CODE_DESCRIPTION, c.vocabulary_id AS SOURCE_VOCABULARY_ID,  
           c.domain_id AS SOURCE_DOMAIN_ID, c.concept_class_id AS SOURCE_CONCEPT_CLASS_ID,  
           c.VALID_START_DATE AS SOURCE_VALID_START_DATE, c.VALID_END_DATE AS SOURCE_VALID_END_DATE,  
           c.invalid_reason AS SOURCE_INVALID_REASON, c.concept_id AS TARGET_CONCEPT_ID,  
           c.concept_name AS TARGET_CONCEPT_NAME, c.vocabulary_id AS TARGET_VOCABULARY_ID,  
           c.domain_id AS TARGET_DOMAIN_ID, c.concept_class_id AS TARGET_CONCEPT_CLASS_ID,  
           c.INVALID_REASON AS TARGET_INVALID_REASON, c.STANDARD_CONCEPT AS TARGET_STANDARD_CONCEPT  
    FROM CONCEPT c  
    UNION  
    SELECT source_code, SOURCE_CODE_DESCRIPTION, source_vocabulary_id,  
           c1.domain_id AS SOURCE_DOMAIN_ID, c2.CONCEPT_CLASS_ID AS SOURCE_CONCEPT_CLASS_ID,  
           c1.VALID_START_DATE AS SOURCE_VALID_START_DATE, c1.VALID_END_DATE AS SOURCE_VALID_END_DATE,  
           stcm.INVALID_REASON AS SOURCE_INVALID_REASON, target_concept_id,  
           c2.CONCEPT_NAME AS TARGET_CONCEPT_NAME, target_vocabulary_id, c2.domain_id AS TARGET_DOMAIN_ID,  
           c2.concept_class_id AS TARGET_CONCEPT_CLASS_ID, c2.INVALID_REASON AS TARGET_INVALID_REASON,  
           c2.standard_concept AS TARGET_STANDARD_CONCEPT  
    FROM source_to_concept_map stcm  
    LEFT OUTER JOIN CONCEPT c1  
        ON c1.concept_id = stcm.source_concept_id  
    LEFT OUTER JOIN CONCEPT c2  
        ON c2.CONCEPT_ID = stcm.target_concept_id  
    WHERE stcm.INVALID_REASON IS NULL  
  )  
SELECT *  
FROM CTE_VOCAB_MAP  
/*EXAMPLE FILTERS*/  
WHERE SOURCE_CODE = '266599000'  
AND SOURCE_VOCABULARY_ID = 'SNOMED'
WITH CTE_VOCAB_MAP AS (  
SELECT c.concept_code AS SOURCE_CODE, c.concept_id AS SOURCE_CONCEPT_ID,  
c.CONCEPT_NAME AS SOURCE_CODE_DESCRIPTION, c.vocabulary_id AS SOURCE_VOCABULARY_ID,  
c.domain_id AS SOURCE_DOMAIN_ID, c.concept_class_id AS SOURCE_CONCEPT_CLASS_ID,  
c.VALID_START_DATE AS SOURCE_VALID_START_DATE, c.VALID_END_DATE AS SOURCE_VALID_END_DATE,  
c.invalid_reason AS SOURCE_INVALID_REASON, c.concept_name AS TARGET_CONCEPT_NAME,  
c.domain_id AS TARGET_DOMAIN_ID, c.concept_class_id AS TARGET_CONCEPT_CLASS_ID,  
c.INVALID_REASON AS TARGET_INVALID_REASON, c.STANDARD_CONCEPT AS TARGET_STANDARD_CONCEPT  
FROM CONCEPT c  
UNION  
SELECT source_code, SOURCE_CONCEPT_ID, SOURCE_CODE_DESCRIPTION, source_vocabulary_id,  
c1.domain_id AS SOURCE_DOMAIN_ID, c2.CONCEPT_CLASS_ID AS SOURCE_CONCEPT_CLASS_ID,  
c1.VALID_START_DATE AS SOURCE_VALID_START_DATE, c1.VALID_END_DATE AS SOURCE_VALID_END_DATE,  
c2.INVALID_REASON, target_concept_id  
FROM source_to_concept_map stcm  
LEFT OUTER JOIN CONCEPT c1 ON c1.concept_id = stcm.source_concept_id  
LEFT OUTER JOIN CONCEPT c2 ON c2.CONCEPT_ID = stcm.target_concept_id  
WHERE stcm.INVALID_REASON IS NULL  
)  
SELECT *  
FROM CTE_VOCAB_MAP  
/*EXAMPLE FILTERS*/  
WHERE SOURCE_CODE = '266599000'  
AND SOURCE_VOCABULARY_ID = 'SNOMED'
Mapping a Lauren Row to CONCEPT_ID: Source to Source

<table>
<thead>
<tr>
<th>START</th>
<th>STOP</th>
<th>PATIENT</th>
<th>ENCOUNTER</th>
<th>CODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/6/2010</td>
<td></td>
<td>1</td>
<td>70</td>
<td>266599000</td>
<td>Dysmenorrhea</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TARGET_CONCEPT_ID</th>
<th>TARGET_CONCEPT_NAME</th>
<th>TARGET_DOMAIN_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>194696</td>
<td>Dysmenorrhea</td>
<td>Condition</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CONDITION_CONCEPT_ID</th>
<th>CONDITION_SOURCE_CONCEPT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>194696</td>
<td>194696</td>
</tr>
</tbody>
</table>
Mapping a Lauren Row to CONCEPT_ID: Source to Source

<table>
<thead>
<tr>
<th>START</th>
<th>STOP</th>
<th>PATIENT</th>
<th>ENCOUNTER</th>
<th>CODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/6/2010</td>
<td></td>
<td>1</td>
<td>70</td>
<td>266599000</td>
<td>Dysmenorrhea</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TARGET_CONCEPT_ID</th>
<th>TARGET_CONCEPT_NAME</th>
<th>TARGET_DOMAIN_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>194696</td>
<td>Dysmenorrhea</td>
<td>Condition</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CONDITION_CONCEPT_ID</th>
<th>CONDITION_SOURCE_CONCEPT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>194696</td>
<td>194696</td>
</tr>
</tbody>
</table>

They are the same CONCEPT_IDs because the source codes in Synthea are also standard codes!
Mapping Source Codes – Your turn

• Let’s open PostgreSQL
  – Open up pgAdmin4 using the icon on the task bar
  – Expand the server list and right-click on PostgreSql 10 and choose Connect Server from the drop-down menu
  – When it asks for a password, type in ohdsi
Mapping Source Codes – Your turn

- Expand:
  a. Databases
  b. ETL
  c. Schemas

- Right click on cdm_synthea and choose Query Tool from the drop-down menu
Mapping Source Codes – Your turn

<table>
<thead>
<tr>
<th>CODE</th>
<th>DESCRIPTION</th>
<th>CODE TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>C83.3</td>
<td>Diffuse large B-cell lymphoma</td>
<td>ICD10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(not ICD10CM)</td>
</tr>
</tbody>
</table>

https://github.com/OHDSI/Tutorial-ETL/tree/master/materials/Queries
Mapping Source Codes – Your turn

<table>
<thead>
<tr>
<th>CODE</th>
<th>DESCRIPTION</th>
<th>CODE TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>C83.3</td>
<td>Diffuse large B-cell lymphoma</td>
<td>ICD10 (not ICD10CM)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CONDITION_CONCEPT_ID</th>
<th>CONDITION_SOURCE_CONCEPT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>44808122</td>
<td>45600549</td>
</tr>
</tbody>
</table>

https://github.com/OHDSI/Tutorial-ETL/tree/master/materials/Queries
Usagi

• When the Vocabulary does not have your source codes you will need to create a map to OMOP Vocabulary Concepts

• Usagi is Japanese for rabbit and was named after the first mapping exercise it was used for; mapping source codes used in a Japanese dataset into OMOP Vocabulary concepts

• Usagi software tool to help with process of mapping source codes to OMOP concepts

• Usagi performs text similarity between your source codes and what is in the OMOP Vocabulary
Usagi Process

1. Get a copy of the **Vocabulary** from ATHENA
2. Download **Usagi**
3. Have Usagi **build an index** on the Vocabulary
4. **Load your source codes** and let Usagi process them
5. **Review and update suggest mappings** with someone who has medical knowledge
6. **Export codes** into the SOURCE_TO_CONCEPT_MAP
Usagi Process

1. Get a copy of the **Vocabulary** from ATHENA

http://athena.ohdsi.org
Usagi Process

1. Get a copy of the **Vocabulary** from ATHENA

![Image of a folder containing CSV files and other files]

- CONCEPT.csv
- CONCEPT_ANCESTOR.csv
- CONCEPT_CLASS.csv
- CONCEPT_CPT4.csv
- CONCEPT_RELATIONSHIP.csv
- CONCEPT_SYNONYM.csv
- cpt.bat
- cpt.sh
- cpt4.jar
- DOMAIN.csv
- DRUG_STRENGTH.csv
- readme.txt
- RELATIONSHIP.csv
- VOCABULARY.csv
Usagi Process

2. Download Usagi

https://github.com/OHDSI/Usagi
3. Have Usagi **build an index** on the Vocabulary
4. Load your source codes, let Usagi process them

• Generate an XLSX of distinct codes from source system with descriptions and frequency

• If the codes are not in English, use Google Translate to convert
4. **Load your source codes, let Usagi process them**

- Import the codes into Usagi
5. Review and update suggest mappings with someone who has medical knowledge
5. Review and update suggest mappings with someone who has medical knowledge
Usagi Process

5. **Review and update suggest mappings with someone who has medical knowledge**
Usagi Process

5. **Review and update suggest mappings with someone who has medical knowledge**

![Usagi Process UI](image)
Usagi Process

5. **Review and update suggest mappings with someone who has medical knowledge**

- It is okay to map to zero or 0 – “No matching concept”
- A source code might end up being mapped to two concepts
- You might have what the system considers one domain but the OMOP Vocabulary lumps into another domain
6. Export codes into the SOURCE_TO_CONCEPT_MAP

- After you have completed, you will export the relationships.

- When exporting you will give a Vocabulary ID, for example JNJ_JMDC_PROVIDERS would signify a Johnson & Johnson map for the database JMDC for provider codes.

<table>
<thead>
<tr>
<th>source_code</th>
<th>source_concept_id</th>
<th>source_vocab_id</th>
<th>source_code_description</th>
<th>target_concept_id</th>
<th>target_vocab_id</th>
<th>valid_start_date</th>
<th>valid_end_date</th>
<th>invalid_reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>R74.02</td>
<td>0</td>
<td>TEST_VOCAB</td>
<td>Acute pharyngitis</td>
<td>25297</td>
<td>SNOMED</td>
<td>1/1/1970</td>
<td>12/31/2099</td>
<td></td>
</tr>
</tbody>
</table>

R74.02 - Acute pharyngitis = 25297 - Acute pharyngitis
Usagi Process

6. Export codes into the SOURCE_TO_CONCEPT_MAP

- You then load your generated maps into the empty Vocabulary table.
Usagi – Your Turn

1. Get a copy of the **Vocabulary** from ATHENA

2. Download **Usagi**

3. Have Usagi **build an index** on the Vocabulary

4. **Load your source codes** and let Usagi process them

5. **Review and update suggest mappings** with someone who has medical knowledge

6. **Export codes** into the SOURCE_TO_CONCEPT_MAP
Now Your Turn: Open Usagi

- Click on Usagi shortcut
- Go into the Usagi-1.1.6 folder
- Open Usagi.jar
Usagi – Your Turn

• We have provided a small subset of codes to try to map

https://github.com/OHDSI/Tutorial-ETL/

-> Materials -> Usagi ->

DUTCH_ICPC_CONDITION_CODES_TO_MAP.xlsx

• These condition codes are in Dutch ICPC codes and need to be mapped to standard concepts
Usagi – Your Turn

• Your mission:
  – Download the codes to map
  – Translate codes to English
  – Import codes into Usagi
  – Map to standard concepts
  – Export SOURCE_TO_CONCEPT_MAP table

• For help review the User Guide:
Data experts and CDM experts together design the ETL

People with medical knowledge create the code mappings

All are involved in quality control

A technical person implements the ETL
ETL Implementation

There are multiple tools available to implement your ETL

In this example we created a builder using SQL and R, though your choice will largely depend on the size and complexity of the ETL design
ETL Implementation

General Flow of Implementation

A good rule of thumb is to always create the PERSON table first.

The VISIT_OCCURRENCE table must be created before the standardized clinical data tables as they all refer to the VISIT_OCCURRENCE_ID.
In this example we will not go over the VISIT_OCCURRENCE creation, though a link to how that was done will be provided later in the presentation.
ETL Implementation

First, let’s review the logic we decided on for how the PERSON table should be created.

Navigate in your browser to: https://ohdsi.github.io/ETL-Synthea/Person.html
First, let’s review the logic we decided on for how the PERSON table should be created.

<table>
<thead>
<tr>
<th>Gender:</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>gender_concept_id</td>
<td>gender</td>
<td>When gender = ‘M’ then set gender_concept_id to 8507, when gender = ‘F’ then set to 8532</td>
<td>Drop any rows with missing/unknown gender</td>
</tr>
<tr>
<td>Birthdate:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>year_of_birth</td>
<td>birthdate</td>
<td>Take year from birthdate</td>
<td></td>
</tr>
<tr>
<td>month_of_birth</td>
<td>birthdate</td>
<td>Take month from birthdate</td>
<td></td>
</tr>
<tr>
<td>day_of_birth</td>
<td>birthdate</td>
<td>Take day from birthdate</td>
<td></td>
</tr>
<tr>
<td>birth_datetime</td>
<td>birthdate</td>
<td>With midnight as time 00:00:00</td>
<td></td>
</tr>
<tr>
<td>Race:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>race_concept_id</td>
<td>race</td>
<td>When race = ‘WHITE’ then set as 8527, when race = ‘BLACK’ then set as 8516, when race = ‘ASIAN’ then set as 8515, otherwise set as 0</td>
<td></td>
</tr>
<tr>
<td>Ethnicity:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ethnicity_concept_id</td>
<td>race ethnicity</td>
<td>When race = ‘HISPANIC’, or when ethnicity in (‘CENTRAL_AMERICAN’, ‘DOMINICAN’, ‘MEXICAN’, ‘PUERTO_RICAN’, ‘SOUTH_AMERICAN’) then set as 38003563, otherwise set as 0</td>
<td></td>
</tr>
</tbody>
</table>
How should the PERSON table logic be implemented in SQL?

To open the query while we review:

https://github.com/OHDSI/Tutorial-ETL
Materials → Implementation →
Insert_Person_Lauren.sql

You can either view it directly in GitHub or download it and open it in pgAdmin4
ETL Implementation

How should the PERSON table logic be implemented in SQL?

```sql
truncate cdm_lauren.person;
insert into cdm_lauren.person (  
    person_id,  
    ...  
    ethnicity_source_concept_id
)
select  
    row_number() over (order by p.id) as person_id,
    case upper(p.gender)  
        when 'M' then 8507  
        when 'F' then 8532  
    end as gender_concept_id,
    date_part('year', p.birthdate) as year_of_birth,
    date_part('month', p.birthdate) as month_of_birth,
    date_part('day', p.birthdate) as day_of_birth,
    p.birthdate as birth_datetime,
    case upper(p.race)  
        when 'WHITE' then 8527  
        when 'BLACK' then 8516  
        when 'ASIAN' then 8515  
        else 0  
    end as race_concept_id,
    case  
        when upper(p.race) = 'HISPANIC'  
        then 38003563 else 0  
    end as ethnicity_concept_id,
    ...
```
Let’s review the logic we decided on for how the PERSON table should be created.

<table>
<thead>
<tr>
<th>Gender:</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>gender_concept_id</td>
<td>gender</td>
<td>When gender = ‘M’ then set gender_concept_id to 8507, when gender = ‘F’ then set to 8532</td>
<td>Drop any rows with missing/unknown gender.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Birthdate:</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>year_of_birth</td>
<td>birthdate</td>
<td>Take year from birthdate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>month_of_birth</td>
<td>birthdate</td>
<td>Take month from birthdate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>day_of_birth</td>
<td>birthdate</td>
<td>Take day from birthdate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>birth_datetime</td>
<td>birthdate</td>
<td>With midnight as time 00:00:00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Race:</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>race_concept_id</td>
<td>race</td>
<td>When race = ‘WHITE’ then set as 8527, when race = ‘BLACK’ then set as 8516, when race = ‘ASIAN’ then set as 8515, otherwise set as 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ethnicity:</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ethnicity_concept_id</td>
<td>race ethnicity</td>
<td>When race = ‘HISPANIC’, or when ethnicity in (‘CENTRAL_AMERICAN’, ‘DOMINICAN’, ‘MEXICAN’, ‘PUERTO_RICAN’, ‘SOUTH_AMERICAN’) ) then set as 38003563, otherwise set as 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ETL Implementation

How should the PERSON table logic be implemented in SQL?

```sql
truncate cdm_lauren.person;
insert into cdm_lauren.person (person_id, ethnic_id, ethnicity_source_concept_id)

select
  new_number(lower(order by p.id)) as person_id,
  case upper(p.gender)
    when 'M' then 8507
    when 'F' then 8532
  end as gender_concept_id,
  date_part('year', p.birthdate) as year_of_birth,
  date_part('month', p.birthdate) as month_of_birth,
  date_part('day', p.birthdate) as day_of_birth,
  when upper(p.race) = 'BLACK' then 8516
  when 'ASIAN' then 8515
  else 0
end as race_concept_id,
  when upper(p.race) = 'HISPANIC' then 38003563 else 0
end as ethnicity_concept_id,
...
```

<table>
<thead>
<tr>
<th>gender_concept_id</th>
<th>gender</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8507</td>
<td>'M'</td>
<td>When gender = 'M' then set gender_concept_id to 8507, when gender = 'F' then set to 8532</td>
</tr>
<tr>
<td>8532</td>
<td>'F'</td>
<td></td>
</tr>
<tr>
<td>8516</td>
<td>'BLACK'</td>
<td></td>
</tr>
<tr>
<td>8515</td>
<td>'ASIAN'</td>
<td></td>
</tr>
<tr>
<td>38003563</td>
<td>'HISPANIC'</td>
<td>Drop any rows with missing/unknown gender.</td>
</tr>
</tbody>
</table>

92
ETL Implementation

How should the PERSON table logic be implemented in SQL?

```sql
TRUNCATE cdm_lauren.person;

INSERT INTO cdm_lauren.person (person_id, ...
  ethnicity_source_concept_id
)

SELECT
  new_number(10, order by p.id) as person_id,
  CASE UPPER(p.gender)
    WHEN 'M' THEN 8507
    WHEN 'F' THEN 8532
  END as gender_concept_id,
  date_part('year', p.birthdate) as year_of_birth,
  ...
FROM cdm_lauren.person

DROP any rows with missing/unknown gender.
```

When gender = 'M' then set gender_concept_id to 8507, when gender = 'F' then set to 8532.
ETL Implementation

How should the PERSON table logic be implemented in SQL?

```sql
... 
end as gender_concept_id,
date_part('year', p.birthdate) as year_of_birth,
date_part('month', p.birthdate) as month_of_birth,
date_part('day', p.birthdate) as day_of_birth,
p.birthdate as birth_datetime,
case upper(p.race) 
  when 'WHITE' then 8527 
  when 'BLACK' then 8516 
  when 'ASIAN' then 8515 
else 0 
end as race_concept_id,
case 
  when upper(p.race) = 'HISPANIC'
    gender_concept_id 8507 
  when gender = 'M' then set gender_concept_id to 8507, when gender = 'F' then set to 8532
    gender_concept_id 0

from raw_lauren.patients p
where p.gender is not null;
```

Gender

Drop any rows with missing/unknown gender.
### ETL Implementation

Let’s review the logic we decided on for how the PERSON table should be created.

<table>
<thead>
<tr>
<th>Column</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>gender_concept_id</td>
<td>gender</td>
</tr>
<tr>
<td>gender</td>
<td>When gender = ‘M’ then set gender_concept_id to 8507, when gender = ‘F’ then set to 8532</td>
</tr>
<tr>
<td>birthdate</td>
<td>Drop any rows with missing/unknown gender.</td>
</tr>
<tr>
<td>year_of_birth</td>
<td>birthdate</td>
</tr>
<tr>
<td>month_of_birth</td>
<td>birthdate</td>
</tr>
<tr>
<td>day_of_birth</td>
<td>birthdate</td>
</tr>
<tr>
<td>birth_datetime</td>
<td>birthdate</td>
</tr>
<tr>
<td></td>
<td>With midnight as time 00:00:00</td>
</tr>
<tr>
<td>race_concept_id</td>
<td>race</td>
</tr>
<tr>
<td>race</td>
<td>When race = ‘WHITE’ then set as 8527, when race = ‘BLACK’ then set as 8516, when race = ‘ASIAN’ then set as 8515, otherwise set as 0</td>
</tr>
<tr>
<td>ethnicity_concept_id</td>
<td>race ethnicity</td>
</tr>
<tr>
<td>race ethnicity</td>
<td>When race = ‘HISPANIC’, or when ethnicity in (‘CENTRAL_AMERICAN’, ‘DOMINICAN’, ‘MEXICAN’, ‘PUERTO_RICAN’, ‘SOUTH_AMERICAN’) then set as 38003563, otherwise set as 0</td>
</tr>
</tbody>
</table>
ETL Implementation

How should the PERSON table logic be implemented in SQL?

```sql
truncate cdm_lauren.person;
insert into cdm_lauren.person (person_id, ... ethnicity_source_concept_id )
select
    row_number() over(order by p.id) as person_id,
    case upper(p.gender)
        when 'M' then 8507
        when 'F' then 8532
    end as gender_concept_id,
    date_part('year', p.birthdate) as year_of_birth,
    date_part('month', p.birthdate) as month_of_birth,
    date_part('day', p.birthdate) as day_of_birth,
    p.birthdate as birth_datetime,
    date_part('year', p.birthdate) * 10000 + date_part('month', p.birthdate) * 100 + date_part('day', p.birthdate) as birthdate
from person p
```

<table>
<thead>
<tr>
<th>Year of Birth</th>
<th>Birthdate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Birthdate</td>
<td>Birthdate</td>
<td>Take year from birthdate</td>
</tr>
<tr>
<td>Month of Birth</td>
<td>Birthdate</td>
<td>Take month from birthdate</td>
</tr>
<tr>
<td>Day of Birth</td>
<td>Birthdate</td>
<td>Take day from birthdate</td>
</tr>
<tr>
<td>Birth Datetime</td>
<td>Birthdate</td>
<td>With midnight as time 00:00:00</td>
</tr>
<tr>
<td></td>
<td>38003563</td>
<td></td>
</tr>
</tbody>
</table>

end as ethnicity_concept_id,
How should the PERSON table logic be implemented in SQL?

```sql
truncate cdm_lauren.person;
insert into cdm_lauren.person (person_id, ...
  ethnicity_source_concept_id
)
select row_number() over(order by p.id) as person_id,
case upper(p.gender)
  when 'M' then 8507
  when 'F' then 8532
end as gender_concept_id,
date_part('year', p.birthdate) as year_of_birth,
date_part('month', p.birthdate) as month_of_birth,
date_part('day', p.birthdate) as day_of_birth,
p.birthdate as birth_datetime,
date_part('year', p.birthdate) - 1900 as year_of_birth
  birth_datetime
  birthdate
  Take year from birthdate
  month_of_birth
  birthdate
  Take month from birthdate
day_of_birth
  birthdate
  Take day from birthdate
  birthDatetime
  birthdate
  With midnight as time 00:00:00
...
Let’s review the logic we decided on for how the PERSON table should be created.

### Gender:

<table>
<thead>
<tr>
<th>gender_concept_id</th>
<th>gender</th>
<th>When gender = ‘M’ then set gender_concept_id to 8507, when gender = ‘F’ then set to 8532</th>
<th>Drop any rows with missing/unknown gender.</th>
</tr>
</thead>
</table>

### Birthdate:

<table>
<thead>
<tr>
<th>year_of_birth</th>
<th>birthdate</th>
<th>Take year from birthdate</th>
</tr>
</thead>
<tbody>
<tr>
<td>month_of_birth</td>
<td>birthdate</td>
<td>Take month from birthdate</td>
</tr>
<tr>
<td>day_of_birth</td>
<td>birthdate</td>
<td>Take day from birthdate</td>
</tr>
<tr>
<td>birth_datetime</td>
<td>birthdate</td>
<td>With midnight as time 00:00:00</td>
</tr>
</tbody>
</table>

### Race:

<table>
<thead>
<tr>
<th>race_concept_id</th>
<th>race</th>
<th>When race = ‘WHITE’ then set as 8527, when race = ‘BLACK’ then set as 8516, when race = ‘ASIAN’ then set as 8515, otherwise set as 0</th>
</tr>
</thead>
</table>

### Ethnicity:

<table>
<thead>
<tr>
<th>ethnicity_concept_id</th>
<th>race ethnicity</th>
<th>When race = ‘HISPANIC’, or when ethnicity in (‘CENTRAL_AMERICAN’, ‘DOMINICAN’, ‘MEXICAN’, ‘PUERTO_RICAN’, ‘SOUTH_AMERICAN’) then set as 38003563, otherwise set as 0</th>
</tr>
</thead>
</table>
ETL Implementation

How should the PERSON table logic be implemented in SQL?

```sql
truncate cdm_lauren.person;
insert into cdm_lauren.person (
    person_id,
    ...
    ethnicity_source_concept_id
)
    select
        row_number() over(order by p.id) as person_id,
        case upper(p.gender)
            when 'M' then 8507
            when 'F' then 8532
        end as race_concept_id,
        case upper(p.race)
            when 'WHITE' then 8527
            when 'BLACK' then 8516
            when 'ASIAN' then 8515
            else 0
        end as race_concept_id,
    ...
```

Race

<table>
<thead>
<tr>
<th>race_concept_id</th>
<th>race</th>
</tr>
</thead>
<tbody>
<tr>
<td>8527</td>
<td>WHITE</td>
</tr>
<tr>
<td>8516</td>
<td>BLACK</td>
</tr>
<tr>
<td>8515</td>
<td>ASIAN</td>
</tr>
</tbody>
</table>

When race = ‘WHITE’ then set as 8527,
when race = ‘BLACK’ then set as 8516,
when race = ‘ASIAN’ then set as 8515,
on otherwise set as 0.
Let’s review the logic we decided on for how the PERSON table should be created.

### Gender:

<table>
<thead>
<tr>
<th>gender_concept_id</th>
<th>gender</th>
<th>When gender = ‘M’ then set gender_concept_id to 8507, when gender = ‘F’ then set to 8532</th>
<th>Drop any rows with missing/unknown gender.</th>
</tr>
</thead>
</table>

### Birthdate:

<table>
<thead>
<tr>
<th>year_of_birth</th>
<th>birthdate</th>
<th>Take year from birthdate.</th>
</tr>
</thead>
<tbody>
<tr>
<td>month_of_birth</td>
<td>birthdate</td>
<td>Take month from birthdate.</td>
</tr>
<tr>
<td>day_of_birth</td>
<td>birthdate</td>
<td>Take day from birthdate.</td>
</tr>
<tr>
<td>birth_datetime</td>
<td>birthdate</td>
<td>With midnight as time 00:00:00</td>
</tr>
</tbody>
</table>

### Race:

<table>
<thead>
<tr>
<th>race_concept_id</th>
<th>race</th>
<th>When race = ‘WHITE’ then set as 8527, when race = ‘BLACK’ then set as 8516, when race = ‘ASIAN’ then set as 8515, otherwise set as 0</th>
</tr>
</thead>
</table>

### Ethnicity:

<table>
<thead>
<tr>
<th>ethnicity_concept_id</th>
<th>race ethnicity</th>
<th>When race = ‘HISPANIC’, or when ethnicity in (‘CENTRAL_AMERICAN’, ‘DOMINICAN’, ‘MEXICAN’, ‘PUERTO_RICAN’, ‘SOUTH_AMERICAN’) then set as 38003563, otherwise set as 0</th>
</tr>
</thead>
</table>
How should the PERSON table logic be implemented in SQL?

```sql
truncate cdm_lauren.person;
insert into cdm_lauren.person (person_id, ... ethnicity_source_concept_id)

select row_number() over(order by p.id) as person_id,
    case upper(p.gender)
        when 'M' then 8507
        when 'F' then 8532
    end as gender_concept_id,
    date_part('year', p.birthdate) as year_of_birth,
    date_part('month', p.birthdate) as month_of_birth,
    date_part('day', p.birthdate) as day_of_birth,

<table>
<thead>
<tr>
<th>ethnicity_concept_id</th>
<th>race ethnicity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>When race = 'HISPANIC', or when ethnicity in ('CENTRAL_AMERICAN', 'DOMINICAN', 'MEXICAN', 'PUERTO_RICAN', 'SOUTH_AMERICAN') then set as 38003563, otherwise set as 0</td>
</tr>
</tbody>
</table>
```

```sql
case
    when upper(p.race) = 'HISPANIC'
    then 38003563
    else 0
end as ethnicity_concept_id,
```
ETL Implementation

Now let’s run the code and create the PERSON table in the cdm_lauren schema

1. Download the query from:
   
   https://github.com/OHDSI/Tutorial-ETL
   
   Materials → Implementation → Insert_Person_Lauren.sql

2. Open up pgAdmin4 using the icon on the task bar
ETL Implementation

3. Expand the server list and right-click on **PostgreSql 10** and choose **Connect Server** from the drop-down menu.

4. When it asks for a password, type in **ohdsi**.
ETL Implementation

5. Expand:
   a. Databases
   b. ETL
   c. Schemas

6. Right click on cdm_lauren and choose Query Tool from the drop-down menu
7. Paste the sql code to create the PERSON table into the query window and press F5 or

NOTE:
• The ‘truncate’ statement at the beginning deletes anything that is in the table already without deleting the table itself (helpful if you make a mistake)

How would you check that your PERSON table was created?
First, let’s review the logic we decided on for how the CONDITION_OCCURRENCE table should be created.

Navigate in your browser to: https://ohdsi.github.io/ETL-Synthea/Condition_occurrence.html
First, let’s review the logic we decided on for how the CONDITION_OCCURRENCE table should be created.

<table>
<thead>
<tr>
<th>Person:</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>person_id</td>
<td>patient</td>
<td>Map by mapping person.person_source_value to patient. Find person.person_id by mapping encounters.patient to person.person_source_value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Condition Concept:</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>condition_concept_id</td>
<td>code</td>
<td>Use code to lookup target_concept_id in CTE_TARGET_VOCAB_MAP: select ctvm.target_concept_id from conditions c join cte_target_vocab_map ctvm on ctvm.source_code = c.code and ctvm.target_domain_id = ‘Condition’ and ctvm.target_vocabulary_id = ‘SNOMED’</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Condition Source Concept:</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>condition_source_concept_id</td>
<td>code</td>
<td>Use code to lookup source_concept_id in CTE_SOURCE_VOCAB_MAP: select csvm.source_concept_id from cte_source_vocab_map csvm join conditions c on csvm.source_code = c.code and csvm.source_vocabulary_id = ‘SNOMED’</td>
</tr>
</tbody>
</table>
How should the CONDITION_OCCURRENCE logic be implemented in SQL?

To open the query while we review:

https://github.com/OHDSI/Tutorial-ETL
Materials → Implementation →
Insert.Condition.Occurrence.Lauren.sql

You can either view it directly in GitHub or download it and open it in pgAdmin4
ETL Implementation

```
TRUNCATE cdm_lauren.condition_occurrence;

INSERT INTO cdm_lauren.condition_occurrence
(
  condition_status_concept_id
)
SELECT
  row_number() OVER (ORDER BY p.person_id) AS condition_occurrence_id,
  p.person_id,
  CASE WHEN src_to_standvdm.target_concept_id IS NULL
    THEN 0
    ELSE src_to_standvdm.target_concept_id
  END AS target_concept_id,
  c.start AS condition_start_date,
  c.start AS condition_start_datetime,
  c.stop AS condition_end_date,
  c.stop AS condition_end_datetime,
  32020 AS condition_type_concept_id,
  cast(null AS varchar) AS stop_reason,
  cast(null AS integer) AS provider_id,
  1 AS visit_occurrence_id,
  0 AS visit_detail_id,
  c.code AS condition_source_value,
) AS condition_source_concept_id
FROM
    
SELECT src_to_standvdm.target_concept_id
FROM cdm_synthea.source_to_source_vocabulary_map src_to_standvdm
WHERE src_to_standvdm.source_code = c.code
AND src_to_standvdm.source_vocabulary_id = 'SNOMED'
)
AS condition_source_concept_id,
NULL condition_status_source_value,
0 condition_status_concept_id
FROM raw_lauren.conditions c
LEFT JOIN cdm_synthea.source_to_standard_vocabulary_map src_to_standvdm
ON src_to_standvdm.source_code = c.code
AND src_to_standvdm.target_domain_id = 'Condition'
AND src_to_standvdm.target_vocabulary_id = 'SNOMED'
AND src_to_standvdm.target_standard_concept = 's'
AND src_to_standvdm.target_invalid_reason IS NULL
JOIN cdm_lauren.person p
ON c.patient = p.person_source_value;
```
How should the CONDITION_OCCURRENCE logic be implemented in SQL?

```sql
select row_number() over (order by p.person_id) as condition_occurrence_id,
       p.person_id,
       case when srcstdvm.target_concept_id is null
             then 0
             else srcstdvm.target_concept_id
       end
from cdm_lauren.condition_occurrence c
join cdm_lauren.person p
on c.patient = p.person_source_value;
```

Map by mapping
person.person_source_value to patient.  
Find person.person_id by mapping
encounters.patient to
person.person_source_value.
How should the CONDITION_OCCURRENCE logic be implemented in SQL?

```sql
select row_number() over(order by p.person_id) as condition_occurrence_id,
    case when srctostdvm.target_concept_id is null
        then 0
        else srctostdvm.target_concept_id
    end as target_concept_id,

from cdm_person p
left join cdm_synthea.source_to_standard_vocab_map srctostdvm
    on srctostdvm.source_code  = c.code
    and srctostdvm.target_domain_id = 'Condition'
    and srctostdvm.target_vocabulary_id = 'SNOMED'
    and srctostdvm.target_standard_concept = 'S'
    and srctostdvm.target_invalid_reason IS NULL

Use code to lookup target_concept_id in CTE_TARGET_VOCAB_MAP: select ctvm.target_concept_id from conditions c join cte_target_vocab_map ctvm on ctvm.source_code = c.code and ctvm.target_domain_id = 'Condition' and ctvm.target_vocabulary_id = 'SNOMED'
```
ETL Implementation

How should the CONDITION_OCCURRENCE logic be implemented in SQL?

```sql
select
    row_number() over(order by p.person_id) as condition_occurrence_id,
    p.person_id,
    c.code as condition_source_value,
    ( case when source_concept_id is null then 0 else source_concept_id end ) as source_concept_id
from ( select
        srctosrvcvm.source_concept_id
    from cdm_synthea.source_to_source_vocab_map srctosrvcvm
    where srctosrvcvm.source_code = c.code
    and srctosrvcvm.source_vocabulary_id = 'SNOMED'
    ) a
and
null condition status as source_value
from
    condition_source_concept_id
code
    Use code to lookup source_concept_id in CTE_SOURCE_VOCAB_MAP: select
    csvm.source_concept_id from
    cte_source_vocab_map csvm join
    conditions c on csvm.source_code = c.code and csvm.source_vocabulary_id = 'SNOMED'
```
Now let’s run the code and create the CONDITION_OCCURRENCE table in the cdm_lauren schema

Download the query from:

https://github.com/OHDSI/Tutorial-ETL

Materials ➔ Implementation ➔
Insert.Condition_Occurrence_Lauren.sql

**NOTE:** Make sure you have created the PERSON table in the cdm_lauren schema or this sql script will not work
ETL Implementation – Your Turn

1. Review the OBSERVATION_PERIOD logic

2. Think through how that could be represented using SQL

**Note:** If you are not a SQL programmer don’t worry! Feel free to use this time to explore the Achilles tool through the browser at [http://localhost/achilles/#/Synthea/dashboard](http://localhost/achilles/#/Synthea/dashboard)
ETL Implementation – Your Turn

```sql
truncate table cdm_lauren.observation_period;

insert into cdm_lauren.observation_period (observation_period_id, person_id, observation_period_start_date, observation_period_end_date, period_type_concept_id)
select 1 as observation_period_id, p.person_id, 
min(e.start) as observation_period_start_date, 
max(e.stop) as observation_period_end_date, 
44814724 as period_type_concept_id
from cdm_lauren.person p
join raw_lauren.encounters e
on p.person_source_value = e.patient
group by p.person_id;
```

https://github.com/OHDSI/Tutorial-ETL
Materials → Implementation→ Insert_Observation_Period_Lauren.sql
ETL Implementation – Your Turn

```
truncate table cdm_lauren.observation_period;

insert into cdm_lauren.observation_period (observation_period_id, person_id, observation_period_start_date, observation_period_end_date, period_type_concept_id)

select 1 as observation_period_id, p.person_id, 
       min(e.start) as observation_period_start_date, 
       max(e.stop) as observation_period_end_date, 
       e.period_type_concept_id 
from cdm_lauren.person p 
join raw_lauren.encounters e 
on p.person_source_value = e.patient 
group by p.person_id;
```

https://github.com/OHDSI/Tutorial-ETL
Materials → Implementation → Insert_Observation_Period_Lauren.sql
Resources

• The full Synthea builder can be found here: 
  https://github.com/OHDSI/ETL-Synthea

• Another example of a R/SQL builder for a much larger database: 
  https://github.com/OHDSI/ETL-HealthVerityBuilder

• A builder created using .NET: 
  https://github.com/OHDSI/ETL-CDMBuilder

• A builder created using the AWS lambda functionality: 
  https://github.com/OHDSI/ETL-lambdabuilder 
  (in development)
Data experts and CDM experts together design the ETL

People with medical knowledge create the code mappings

All are involved in quality control

A technical person implements the ETL
What tools are available to check that the CDM logic was implemented correctly?

Achilles HEEL

Rabbit-in-a-Hat Test Case Framework
Achilles

Achilles is a data characterization and quality tool available for download here:

https://github.com/OHDSI/Achilles

For an example of how it was run for our sample data, that R script is located here:

Achilles

Synthea
Dashboard

CDM Summary

Source name: Synthea
Number of 1.13k persons:

Population by Gender

Age at First Observation

Cumulative Observation

Persons With Continuous Observation By Month

122
Achilles Heel

Achilles heel is a report generated by the Achilles application that will run a series of data quality checks on the CDM.

<table>
<thead>
<tr>
<th>Message Type</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR</td>
<td>410-Number of condition occurrence records outside valid observation period; count (n=134) should not be &gt;</td>
</tr>
<tr>
<td>ERROR</td>
<td>610-Number of procedure occurrence records outside valid observation period; count (n=11) should not be &gt;</td>
</tr>
<tr>
<td>ERROR</td>
<td>710-Number of drug exposure records outside valid observation period; count (n=241) should not be &gt; 0</td>
</tr>
<tr>
<td>ERROR</td>
<td>712-Number of drug exposure records with invalid provider_id; count (n=29,518) should not be &gt; 0</td>
</tr>
<tr>
<td>ERROR</td>
<td>810-Number of observation records outside valid observation period; count (n=134) should not be &gt; 0</td>
</tr>
<tr>
<td>ERROR</td>
<td>812-Number of observation records with invalid provider_id; count (n=8,518) should not be &gt; 0</td>
</tr>
<tr>
<td>ERROR</td>
<td>909-Number of drug eras outside valid observation period; count (n=55) should not be &gt; 0</td>
</tr>
<tr>
<td>ERROR</td>
<td>1,009-Number of condition eras outside valid observation period; count (n=134) should not be &gt; 0</td>
</tr>
<tr>
<td>NOTIFICATION</td>
<td>[GeneralPopulationOnly] Not all deciles represented at first observation</td>
</tr>
<tr>
<td>NOTIFICATION</td>
<td>Unmapped data over percentage threshold in Measurement</td>
</tr>
<tr>
<td>NOTIFICATION</td>
<td>Unmapped data over percentage threshold in DrugExposure</td>
</tr>
<tr>
<td>NOTIFICATION</td>
<td>Unmapped data over percentage threshold in Observation</td>
</tr>
<tr>
<td>NOTIFICATION</td>
<td>99+ percent of persons have exactly one observation period</td>
</tr>
<tr>
<td>NOTIFICATION</td>
<td>percentage of non-numerical measurement records exceeds general population threshold</td>
</tr>
<tr>
<td>NOTIFICATION</td>
<td>Unmapped data over percentage threshold in Condition</td>
</tr>
</tbody>
</table>

Showing 1 to 15 of 25 entries
The application has a feature called ‘Generate ETL Test Framework’. This feature allows you to create ‘fake’ people as a way to test your ETL logic.
The test framework creates a series of R functions that enables you to specify your ‘fake’ people and records in the same structure as your source data using the scan report as a guide.

```r
add_patients <- function(id, birthdate, deathdate, ssn, drivers, passport, prefix, first, last, suffix, maiden, marital, state, zip) {
  defaults <- get('patients', envir = frameworkContext$defaultValues)
  fields <- c()
  values <- c()
  if (missing(id)) {
    id <- defaults$id
  }
  if (!is.null(id)) {
    fields <- c(fields, "id")
    values <- c(values, if (is.null(id)) "NULL" else if (is(id, "subQuery")) paste0("("", as.character(id), ")") else paste0("("", as.character(id), ")")
  }
  if (missing(birthdate)) {
    birthdate <- defaults$birthdate
  }
  if (!is.null(birthdate)) {
    fields <- c(fields, "birthdate")
    values <- c(values, if (is.null(birthdate)) "NULL" else if (is(birthdate, "subQuery")) paste0("("", as.character(birthdate), ")")
  }
  if (missing(deathdate)) {
    deathdate <- defaults$deathdate
  }
  if (!is.null(deathdate)) {
    fields <- c(fields, "deathdate")
    values <- c(values, if (is.null(deathdate)) "NULL" else if (is(deathdate, "subQuery")) paste0("("", as.character(deathdate), ")")
  }
}
```
An example of how this was done for the Synthea data is available from:

https://github.com/OHDSI/Tutorial-ETL/tree/master/materials/Unit%20Tests

The file that creates the test cases as a series of insert statement is RunSyntheaTestCases.r
Let’s revisit the PERSON table logic:

<table>
<thead>
<tr>
<th>gender_concept_id</th>
<th>gender</th>
<th>Condition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>When gender = ‘M’ then set gender_concept_id to 8507, when gender = ‘F’ then set to 8532</td>
<td>Drop any rows with missing/unknown gender.</td>
</tr>
</tbody>
</table>

How could we create a test case for this?

```r
createPersonTests <- function () {

    patient <- createPatient()
    declareTest(id = patient$id, description = "Drop patients with no gender, id is PERSON_SOURCE_VALUE")
    add_patients(id = patient$id, gender = NULL)
    expect_no_person(person_source_value = patient$id)
}
```

-- 1: Drop patients with no gender, id is PERSON_SOURCE_VALUE

```sql
INSERT INTO synthea_test.[patients] (id, birthdate, ssn, prefix, first, last, marital, race, ethnicity, birthplace, address, city, state, zip) VALUES ('1', '1926-02-23', '999-41-5589', 'Mr.', 'Benito209', 'Marks830', 'M', 'white', 'irish', 'Boston', '192 MacGyver Dam', 'Boston', 'Massachusetts', '02108');
```
ETL Maintenance

- Changed or Updated Raw Data?
- Bug Found?
- New Vocab?
- CDM Update?

All are involved in quality control

A technical person implements the ETL

Updated CDM

ETL Documentation

ETL

A technical person implements the ETL
ETL Maintenance

Let’s Revisit Ethnicity

```sql
truncate cdm_lauren.person;
insert into cdm_lauren.person (person_id,
    ...
    ethnicity_source_concept_id)

select
    row_number() over(order by p.id) as person_id,
    case upper(p.gender)
        when 'M' then 8507
        when 'F' then 8532
    end as gender_concept_id,
    date_part('year', p.birthdate) as year_of_birth,
    date_part('month', p.birthdate) as month_of_birth,
    date_part('day', p.birthdate) as day_of_birth,
    ethnicity_concept_id, race, ethnicity
    when upper(p.race) = 'HISPANIC'
    then 38003563 else 0
    end as ethnicity_concept_id,
```
Final Hard Lessons Learned
You don’t need to map all terms to get good data coverage!
Comfort with Data Loss

• If there is data that is not of research quality or there are methods to adjust, use the ETL to standardize that

<table>
<thead>
<tr>
<th>Example Patient Drop Counts from a CDM Build</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason to Drop Someone</td>
</tr>
<tr>
<td>Unknown gender</td>
</tr>
<tr>
<td>Implausible year of birth - past</td>
</tr>
<tr>
<td>Implausible year of birth - post earliest observation period</td>
</tr>
<tr>
<td>Gender changes</td>
</tr>
</tbody>
</table>
Thank you!

This tutorial would not have been possible without the contribution of many collaborators in the OHDSI Community.

We like to thank Amazon Web Services for their valuable technical support and resources.
Acknowledgements

Anthony Molinaro who wrote the Synthea CDM Builder

James Wiggins who helps us prepare an AWS instance for use today

Pusheen the Cat

http://pusheen.com/
Second Annual EUROPEAN OHDSI SYMPOSIUM
March 29th 2019
Tutorials 30th and 31st

The Journey from Data to Evidence
Erasmus MC Rotterdam The Netherlands
www.ohdsi-europe.org