

# ETL Guide

## SAPhIRE Model (KTPH ) to Common Data Model V5

07.08.2018

---

Intern: Haroun Chahed

Supervisor: Dr Cynthia Sung

**Health Sciences Authority**

11 Biopolis Way #11-01 Helios  
Singapore, 138667

<b>Overview</b>	<b>2</b>
Abbreviations	2
Introduction to the SAPHIRE KTPH Database	3
Tables 3	
Schema	4
Description	5
Introduction to the CDM V5	5
General Conventions of fields	6
Representation of content through Concepts	7
Schema	7
Mapping Steps	8
<b>Mapping Specifications</b>	<b>9</b>
Step 1: Execute RELEVANT SECTIONS of "0.create_cdm_tables.sql"	9
Step 2: Execute "1.Mapping LOCATION table.sql"	10
Step 3: Execute "1.Mapping PERSON table.sql"	10
Step 4: Execute "2.Mapping VISIT_OCCURENCE table.sql"	11
Step 5: Execute "3.create_visit_id_source_mapping_table.sql"	11
Step 6: Execute "4.filling_visit_id_source_mapping_table.sql"	12
Step 7: Execute "5.Mapping CONDITION_OCCURENCE table Part 1.sql"	12
Step 8: Execute "5.Mapping CONDITION_OCCURENCE table Part 2.sql"	14
Step 9: Execute "5.Mapping DRUG_EXPOSURE table.sql"	14
Step 10: Execute "5.Mapping OBSERVATION_PERIOD.sql"	15
Step 11: Execute "6.Mapping CONDITION_ERA.sql"	15
Step 12: Execute "6.Mapping DRUG_ERA.sql"	16
Step 12: Test the Mapping	16
<b>Code Examples:</b>	<b>16</b>
1.Mapping PERSON table.sql	16
2.create_visit_id_source_mapping_table.sql	18
3.create_visit_id_source_mapping_table.sql	19
4.filling_visit_id_source_mapping_table.sql	19
5.Mapping CONDITION_OCCURENCE table Part 1.sql	19
Mapping DRUG_EXPOSURE table.sql	30
Mapping OBSERVATION_PERIOD.sql	34
6.Mapping CONDITION_ERA.sql	35
<b>References</b>	<b>38</b>

## Overview

The purpose of this document is to describe the Extract, Transform, and Load (ETL) mapping of the HSA data of Khoo Teck Puat Hospital into the Observational Medical Outcomes Partnership (OMOP) Common Data Model (CDM) V5. This document will also be useful to repeat the ETL process in case more data is added to the KTPH Database, or to replicate it with a different database model. Prior to reading this document, the reader should be familiar with the purpose and structure of the CDM.

## Abbreviations

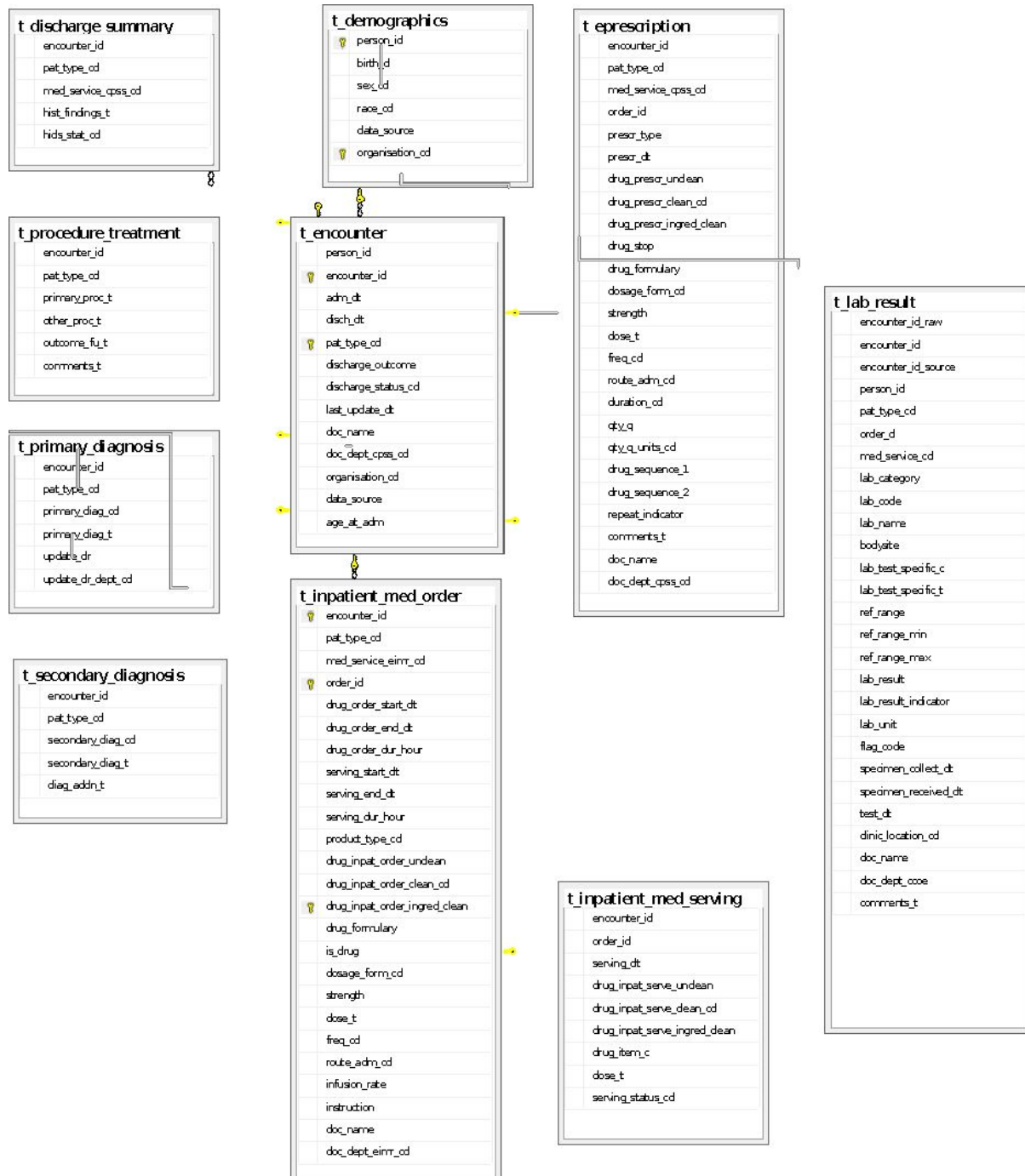
Abbreviation	Description
ETL	Extract, Transform, Load
EMR	Electronic Medical Record
OMOP	Observational Medical Outcomes Partnership
CDM	Common Data Model
KTPH	Khoo Teck Puat Hospital
OHDSI	Observational Health Data Sciences and Informatics
ICD	International Classification of Diseases
RxNorm	a normalized naming system for generic and branded drugs
SNOMED	a systematically organized computer processable collection of medical terms

## Introduction to the SAPHIRE KTPH Database

### Tables

No	Table name	Description
1	demographics	This table contains the patients' profile; contains one record per person ID with the most recent information on weight and height.
2	encounter	This table contains information about the patients' hospital admission and outpatient visits.
3	discharge_summary	This table contains the free text information on medical conditions, adverse drug reaction, treatment and procedure etc.
4	primary_diagnosis	This table describes the primary medical condition why the patient needed treatment.
5	secondary_diagnosis	This table describes the other medical condition(s) why the patient needed treatment.
6	procedure_treatment	This table describes the procedure and treatment given to patients.
7	eprescription	This table describes the medications prescribed to patients at the point of discharge or at outpatient clinics, emergency and day surgery clinic.
8	inpatient_med_order	This table describes the drugs ordered for patients during the inpatient stay.
9	inpatient_med_serving	This table describes the drugs served to patients during the inpatient stay.
10	lab_result	This table describes the laboratory tests performed and the results i.e. 1 laboratory test result per entry

## Schema



## Description

Description	Period	Count (April 2017)	Count (18 June 2018)
Discharge report– inpatient stay & day surgery– inpatient stay & day surgery			t_discharge_summary
No. of unique patients	June 2010 – Dec 2015	101,805	120,536
No. of encounters(case ids)		177,838	219,399
outpatient medications - discharge & outpatient medications			t_eprescription_dispensing
No. of unique patients	Jan 2015 – Dec 2015	127,141	191,153
No. of ordered medications		1,022,200	2,147,870
Inpatient medication order – inpatient stay & day surgery			t_inpatient_med_order
No. of unique patients	Jan 2013 – Mar 2017	60,586	60,586
No. of medications (order level)		2,022,428	2,022,428
Laboratory results			t_lab_results
No. of unique patients	Nov 2011 – June 2016	135,535	146,785
No. of rows of records of lab tests		18,352,997	19,357,717

## Introduction to the CDM V5

No single observational data source provides a comprehensive view of the clinical data a patient accumulates while receiving healthcare, and therefore none can be sufficient to meet all expected outcome analysis needs. This explains the need for assessing and analyzing multiple data sources concurrently using a common data standard. This standard is provided by the OMOP Common Data Model (CDM).

The CDM is designed to support the conduct of research to identify and evaluate associations between interventions (drug exposure, procedures, healthcare policy changes etc.) and outcomes caused by these interventions (condition occurrences, procedures, drug exposure etc.). Outcomes can be efficacious (benefit) or adverse (safety risk). Often times, specific patient cohorts (e.g., those taking a certain drug or suffering from a certain disease) may be defined for treatments or outcomes, using clinical events (diagnoses, observations, procedures, etc.) that occur in predefined temporal relationships to each other. The CDM, combined with its standardized content (via the Standardized Vocabularies), facilitates

research methods that can be systematically applied to produce meaningfully comparable and reproducible results.

## General Conventions of fields

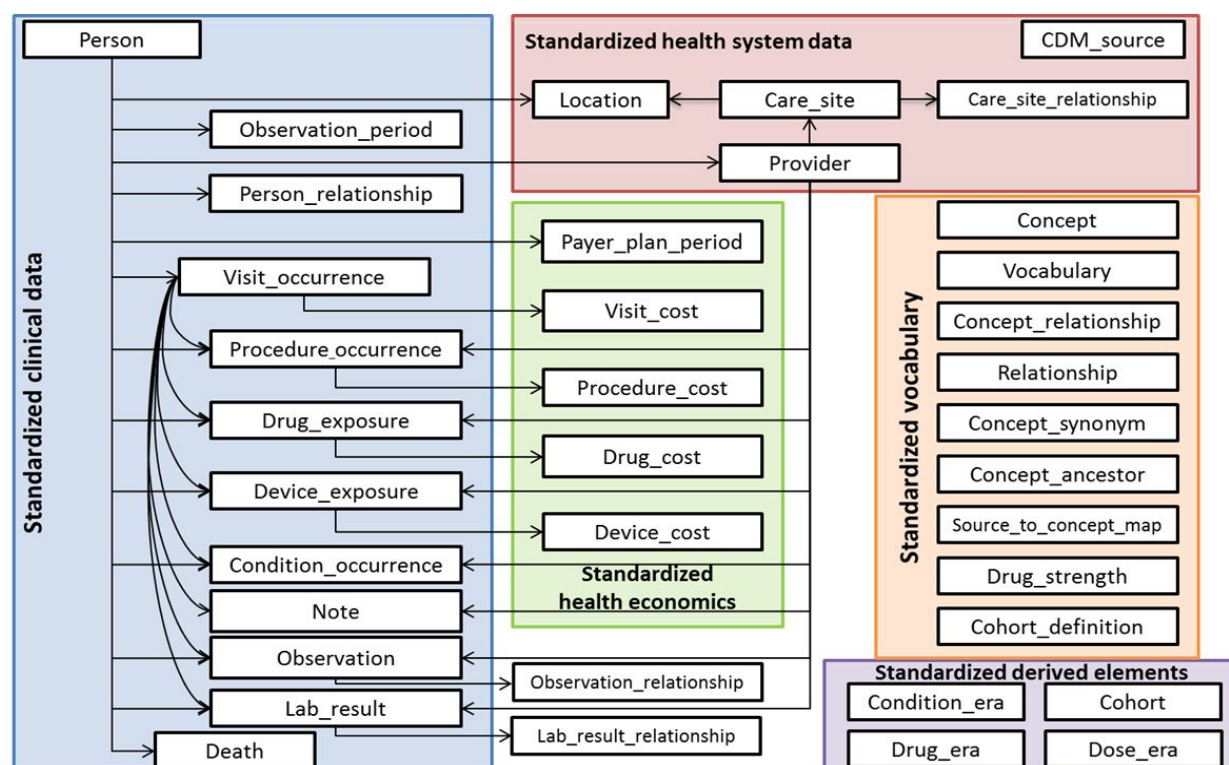
Notation	Description
_SOURCE_VALUE	Verbatim information from the source data, typically used in ETL to map to CONCEPT_ID, and not to be used by any standard analytics. For example, condition_source_value = '787.02' was the ICD-9 diagnosis code that the clinician entered into the EMR
_ID	Unique identifiers for key entities, which can serve as foreign keys to establish relationships across entities For example, person_id uniquely identifies each individual. visit_occurrence_id uniquely identifies a PERSON encounter at a point of care.
_CONCEPT_ID	Foreign key into the Standardized Vocabularies (i.e. the standard_concept attribute for the corresponding term is true), which serves as the primary basis for all standardized analytics For example, condition_concept_id = 31967 contains reference value for SNOMED concept of 'Nausea'
_SOURCE_CONCEPT_ID	Foreign key into the Standardized Vocabularies representing the concept and terminology used in the source data, when applicable For example, condition_source_concept_id = 35708202 denotes the concept of 'Nausea' in the MedDRA terminology; the analogous condition_concept_id might be 31967, since SNOMED-CT is the Standardized Vocabularies for most clinical diagnoses and findings.
_TYPE_CONCEPT_ID	Delineates the type of the original source information, standardized within the Standardized Vocabularies For example, drug_type_concept_id can allow analysts to discriminate between 'Prescription dispensed in pharmacy' (38000175) and 'Prescription written' (38000177)

## Representation of content through Concepts

In CDM data tables the meaning of the content of each record is represented using Concepts. Concepts are stored with their concept\_id as foreign keys to the CONCEPT table in the Standardized Vocabularies, which contains Concepts necessary to describe the healthcare experience of a patient. If a Standard Concept does not exist or cannot be identified, the Concept with the concept\_id 0 is used, representing a non-existing or unmappable concept. Records in the CONCEPT table contain all the detailed information about it (name, relationships, types etc.).

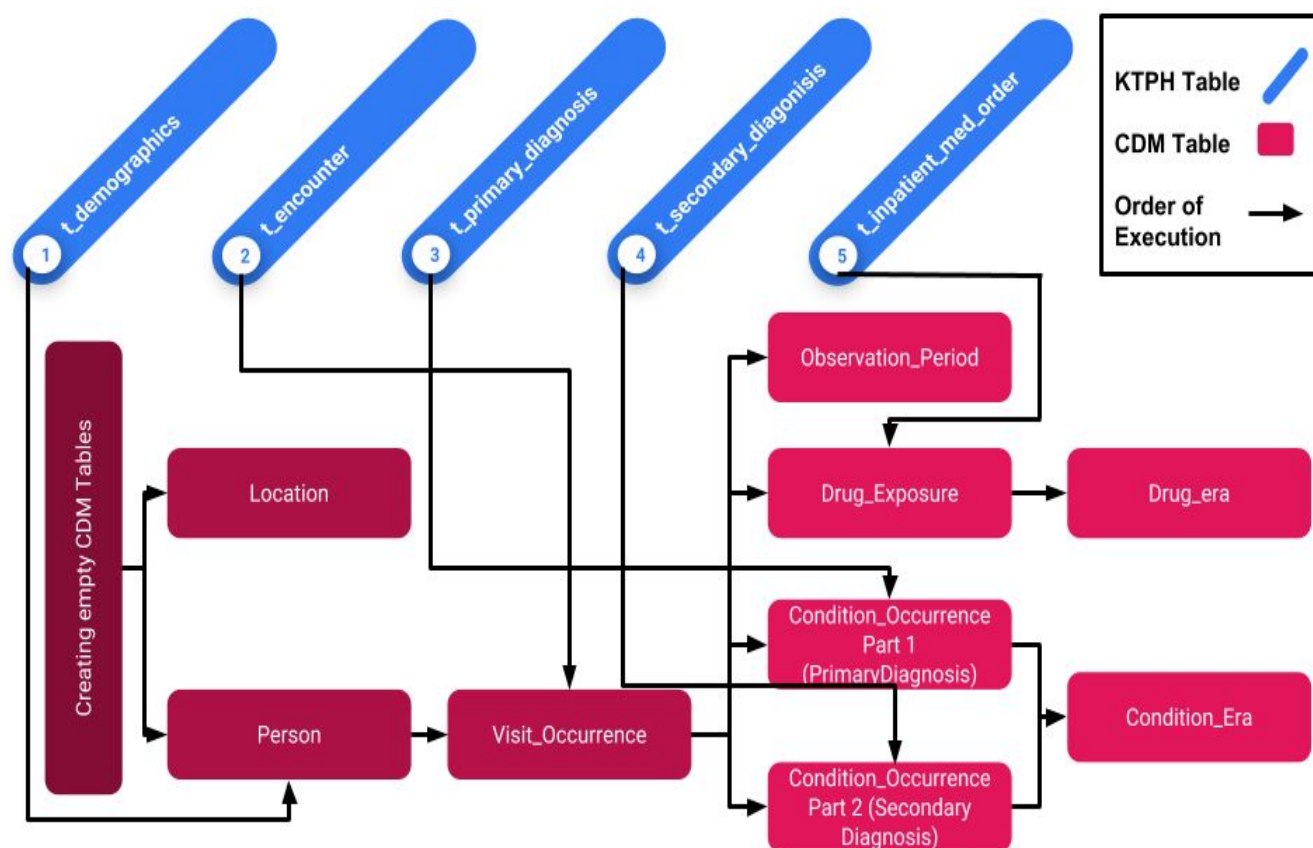
The Concept table is also accessible through a website ([Link](#)). The filters on the right allow us to make a query without writing any line of code. In other words, we can find a concept (or a number of concepts) by selecting its domain, vocabulary, and class, and deciding whether it is standard and/or valid or not.

## Schema



## Mapping Steps

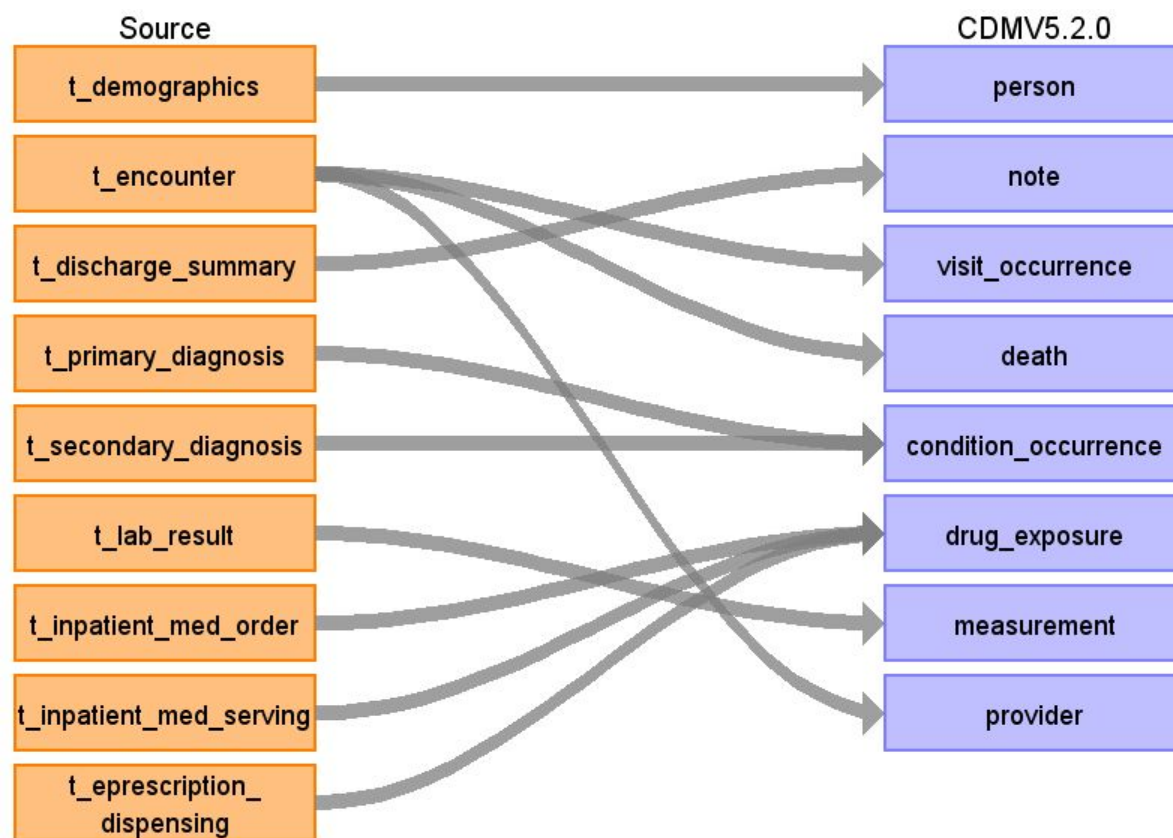




The order of the execution of the ETL steps is crucial for a valid mapping. This is because the filling of CDM tables uses data from other tables from the KTPH database AND/OR tables from the CDM. For instance, filling the table **Visit\_Occurrence (CDM)** requires data from **t\_encounter (KTPH)** that contains all the information about the date, time and other details of the visit, but also requires data from **Person (CDM)** that contains data about the patient.

In the graph above, we map some, but not all, tables of the CDM. Some of the tables on the CDM are supposed to contain data that does not exist in the KTPH database. Naturally, these tables will stay empty. However, there is one exception of one table whose data exists in the KTPH database but that we do not map to the CDM, and this table is **lab\_results**. The reason we do not map this table is that KTPH uses internal codes for their lab tests that cannot be mapped to a known dictionary, that can later be mapped to the CDM standardized dictionary for the lab test. KTPH will provide HSA in the future with a mapping from their internal codes to LOINC vocabulary. Once this mapping is available, applying ETL on the **lab\_results** table will be straightforward because a table already exists for mapping LOINC codes to OMOP concepts.

This is the mapping from the source database (KTPH tables) to the the target database (CDMV5.2.0) generated by OHDSI's Rabbit-In-a-Hat tool.



## Mapping Specifications

This section explains each step of the ETL process. This ETL is to be performed on an empty CDM (i.e., when more data are added to the KTPH, the tables of the the CDM should be deleted and the ETL process should be started from scratch). This ETL takes account for creation of the CDM, but for update.

### Step 1: Execute RELEVANT SECTIONS of “0.create\_cdm\_tables.sql”

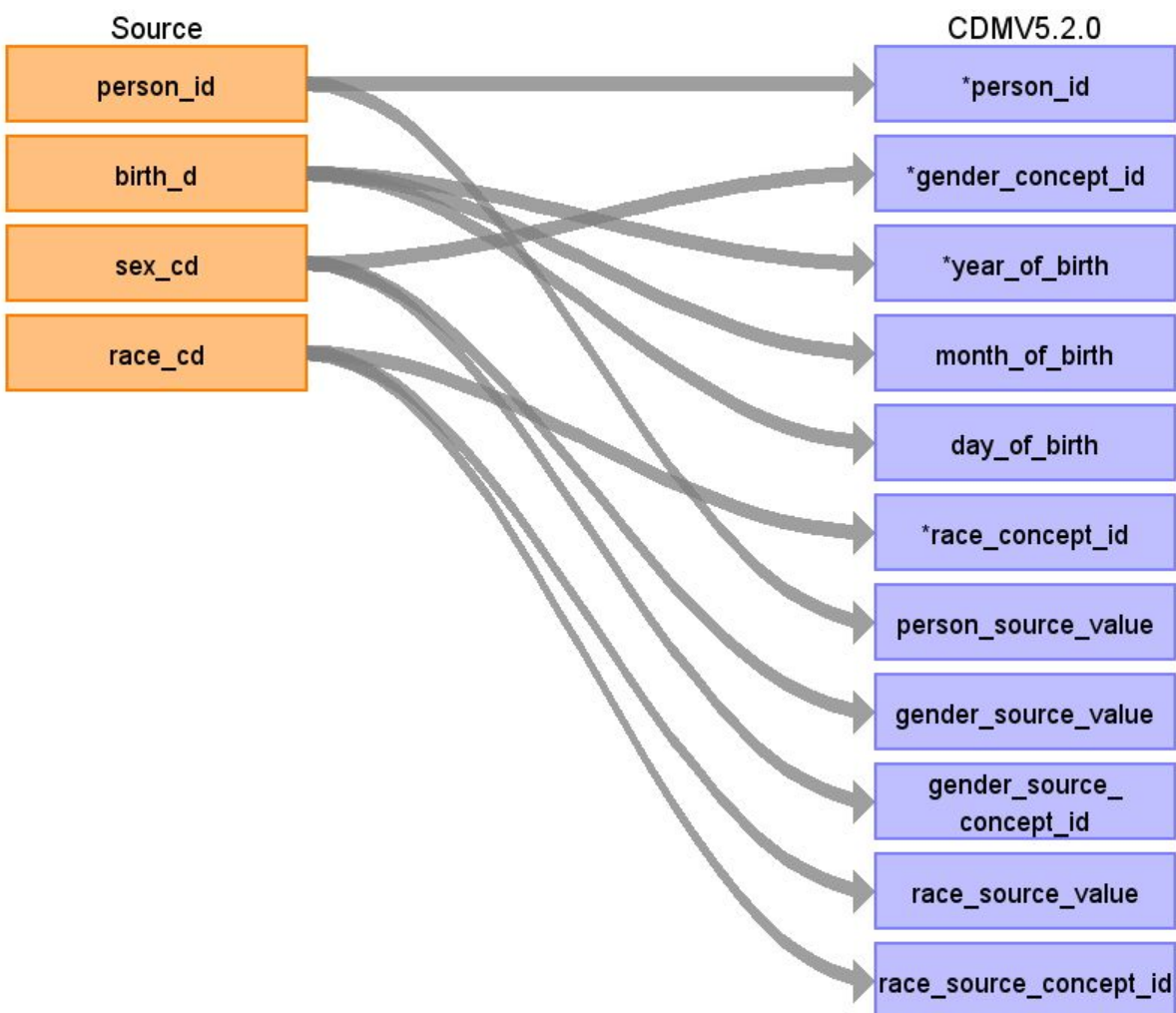
This codes drops all tables of CDM and creates new empty ones. While we will need to delete some tables and recreate them, we do not want to drop all tables. The tables of the standardized vocabulary (refer to schema above) will not need to be updated (of course, unless OHDSI updates them). This is to say that it will not be necessary to recreate them. If we end up deleting these tables, then we will need to download them from Athena ([Link](#)) and import them to the server using the import wizard. Ideally, we should recreate only the tables that we need to map from the KTPH database.

## Step 2: Execute “1.Mapping LOCATION table.sql”

This code fills the location table with the hospitals KTPH and NUH. The LOCATION table represents a generic way to capture physical location or address information of Persons and Care Sites.

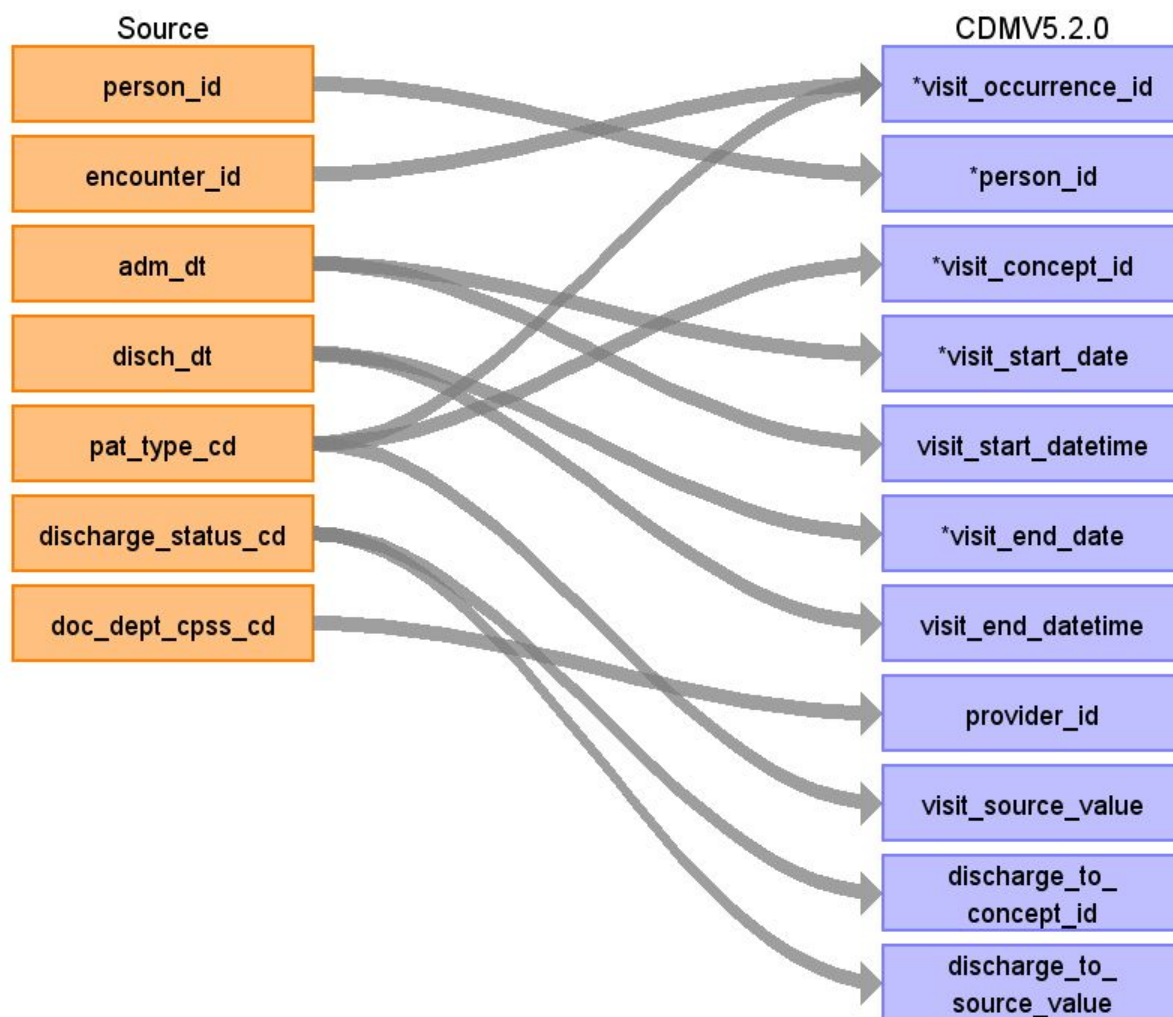
## Step 3: Execute “1.Mapping PERSON table.sql”

This table fills the Person table from data in t\_demographics. The Person Domain contains records that uniquely identify each patient in the source data who is time at-risk to have clinical observations recorded within the source systems. The mapping is as follows:



### Step 4: Execute “2.Mapping VISIT\_OCCURENCE table.sql”

This code fills the Visit\_Occurrence table with data from t\_encounter. The VISIT\_OCCURRENCE table contains the spans of time a Person continuously receives medical services from one or more providers at a Care Site in a given setting within the health care system. Visits are classified into 4 types: outpatient care, inpatient confinement, emergency room, and long-term care. Persons may transition between these settings over the course of an episode of care (for example, treatment of a disease onset). The mapping is as follows:



### Step 5: Execute “3.create\_visit\_id\_source\_mapping\_table.sql”

This code creates an empty table with three columns: encounter\_id, pat\_type\_cd, and visit\_occurrence\_id. The first two columns are the primary key of t\_encounter (SAPHIRE KTPH), and the third is the primary key of Visit\_occurrence (OHDSI KTPH). This table makes

it easier to link the primary keys of the two databases. Usually, the primary key of a database is a numeric value in one column. The CDM has one column for source value, that is supposed to contain this source ID. However, in this case t\_encounter has two columns that make up its primary key (encounter\_id and pat\_type\_cd). Hence, the one column that visit\_occurrence has for the Id source value is not enough to contain both columns. Therefore, we create this extra table that will allow us to connect the primary key of t\_encounter with the primary key of visit\_occurrence. Indeed, adding this one table makes sure that the mapping is sound and complete (i.e., it preserves and accurately describes the data).

### Step 6: Execute “4.filling\_visit\_id\_source\_mapping\_table.sql”

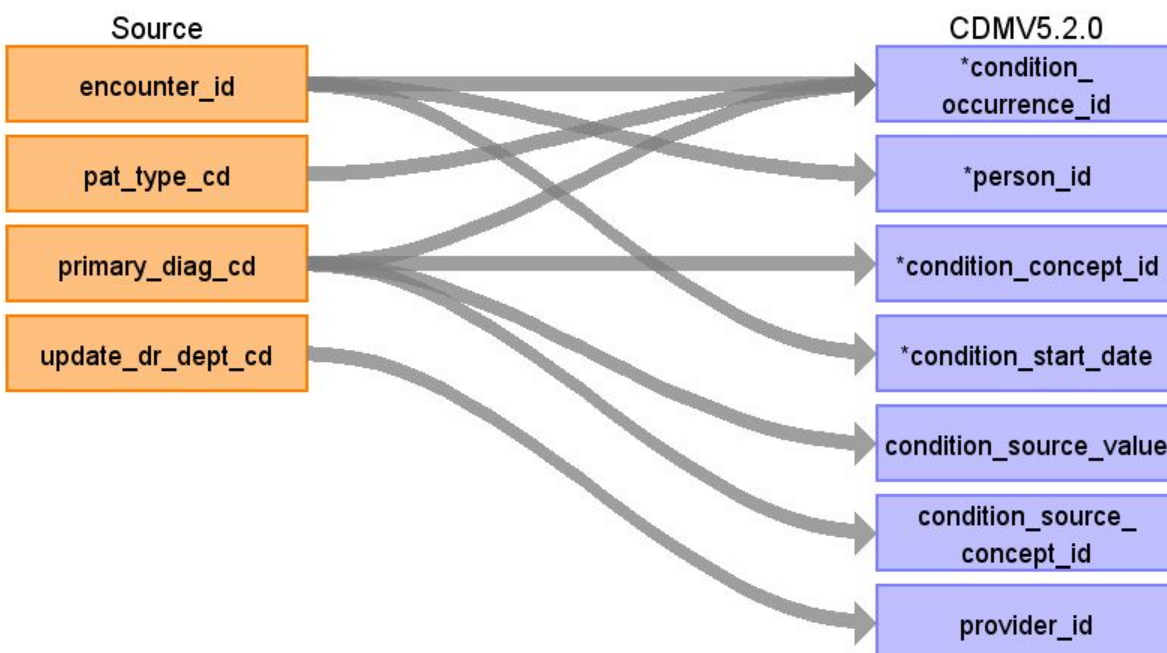
This table fills the table visit\_id\_source\_mapping\_table explained in Step 5.

### Step 7: Execute “5.Mapping CONDITION\_OCCURENCE table Part 1.sql”

This code fills the table condition\_occurrence with data from primary\_diagnosis. Conditions are records of a Person suggesting the presence of a disease or medical condition stated as a diagnosis, a sign or a symptom, which is either recorded by a Provider or reported by the patient. Conditions are recorded in different sources and levels of standardization, for example:

- Medical claims data include diagnoses coded in ICD-9-CM that are submitted as part of a reimbursement claim for health services and
- EMRs may capture Person Conditions in the form of diagnosis codes or symptoms.

The mapping is as follows:

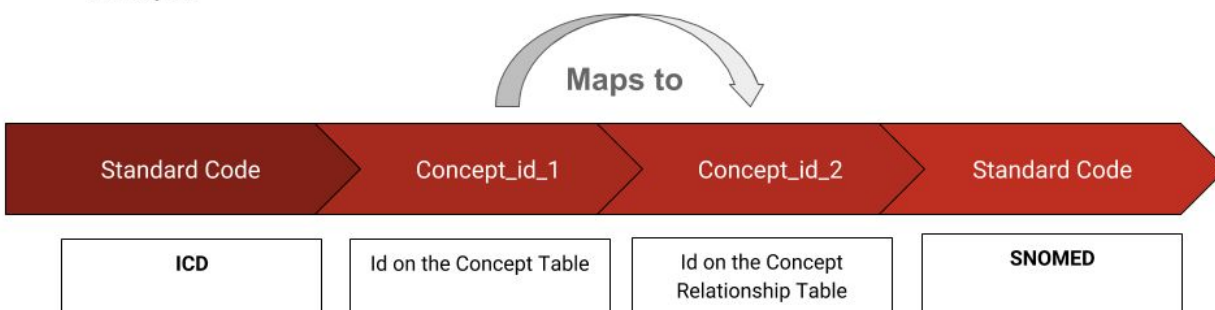




To map the diagnosis codes in `t_primary_diagnosis` to the standardized `concept_ids` in `condition_occurrence`, we have to go through the steps in the chart below. First, we use the Concept table of the standardized vocabulary to match each ICD code with a concept ID. We can find the exact specific ID of each ICD code because we know its `concept_code` and `vocabulary_id`, which are a primary key of the Concept table. Then we use the Concept\_Relationship table to map the `concept_id` of the ICD code to a standardized `concept_id` (corresponding to a SNOMED code).

## Description of mapping

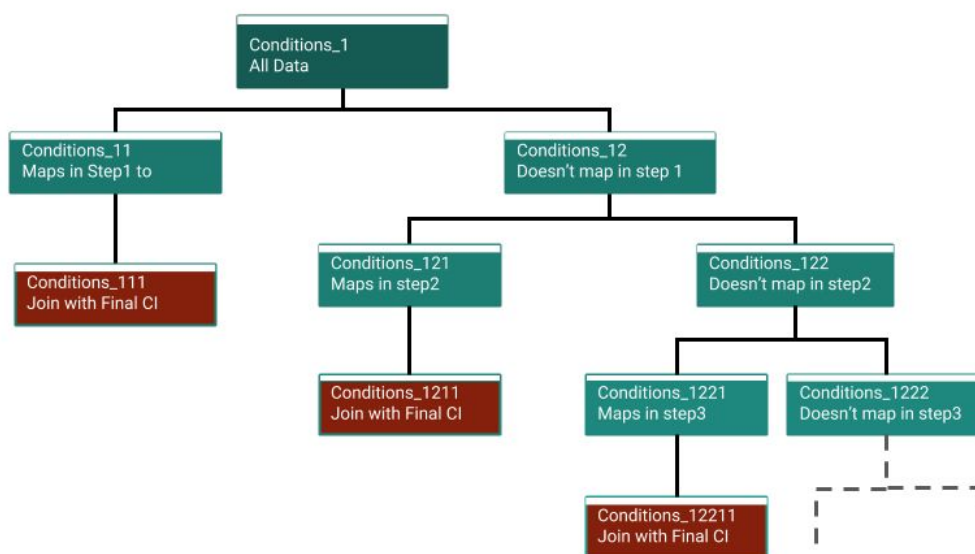
**Example:**



Unfortunately, the ICD codes are not all in standard formats. For example, when the ICD-9CM codes were extracted from the KTPH database, zeros at the beginning of the code were dropped. The location of decimal points in the code often were shifted by one digit. The incorrect codes could be identified by examining the corresponding text of the code. In the `secondary_diagnosis_cd`, KTPH added a T. before some codes, which presumably signified `tertiary_diagnosis_codes`; these had to be stripped off to match to the ICD code. In some cases, the KTPH code had more granularity (more digits after the decimal point) than exists in the OMOP Standard Vocabularies, and these then have to be mapped to the parent concept. Therefore, we must make a few alterations to them before mapping them by joining them with the `concept_code` of the ICD codes on the Concept table of the standardized vocabulary. We apply these modification sequentially as indicated in the graph below. In other words, we first map all the KTPH ICD codes to the Concept ICD codes. Then we take those that have not been mapped, we make modification 1 to them and map them to the Concept ICD codes. Then we take those that are still not mapped, we make modification 2 to them and map them to the Concept ICD codes. So on and so forth, until all the KTPH diagnosis codes are mapped.

Mapping Concept\_id\_1 to Concept\_id\_2 is accomplished by the OMOP Concept\_Relationship Table in the Standardized Vocabulary (the section of the CDM schema that is provided filled by OHDSI). To download this table along with all other tables in the Standardized Vocabulary (Concept, Concept\_Ancessor, Relationship, Vocabulary, etc.), we can press the Download button on [Athena](#) and follow the instructions.

## Mapping Condition Occurrence Table



Conditions\_1 contains all the data. Conditions\_11 contains the data that has been mapped in the first step. Condition\_12 contains the data that has not been mapped in the first step. Conditions\_121 contains the data that has not been mapped in the first step, but that has been mapped in the second. Conditions\_122 contains the data that has not been mapped neither in the first nor in the second step. By checking which ICD code belongs to what table, we can know at what step exactly this ICD code was mapped. The Condition\_Occurrence table should finally contains all the data in the table colored in red (Conditions\_111, Conditions\_1211, Conditions\_12211, Conditions\_122211...)

### Step 8: Execute “5.Mapping CONDITION\_OCCURENCE table Part 2.sql”

This code performs the same operations describes in step 7, but on the KTPH table t\_secondary\_diagnosis, not t\_primary\_diagnosis.

### Step 9: Execute “5.Mapping DRUG\_EXPOSURE table.sql”

This code fills the table Drug\_Exposure with data from t\_impatient\_med\_order.

The drug exposure domain captures records about the utilization of a Drug when ingested or otherwise introduced into the body. A Drug is a biochemical substance formulated in

such a way that when administered to a Person it will exert a certain physiological effect. Drugs include prescription and over-the-counter medicines, vaccines, and large-molecule biologic therapies. Drug Exposure is inferred from clinical events associated with orders, prescriptions written, pharmacy dispensings, procedural administrations, and other patient-reported information, for example:

- The “Prescription” section of an EMR captures prescriptions written by physicians or from electronic ordering systems
- The “Medication list” section of an EMR for both non-prescription products and medications prescribed by other providers
- Prescriptions filled at dispensing providers such as pharmacies, and then captured in reimbursement claim systems
- Drugs administered as part of a Procedure, such as chemotherapy or vaccines.

To map the drugs in `t_inpatient_med_order` to standardized `concept_id` (that happen to be RxNorm codes) we will have to use the drug names and match them with the `Concept_name` column of the `Concept` table of the standardized vocabulary. The reason we cannot use the `concept_code` like we did when mapping diagnosis, is that the SAPHIRE KTPH database for drugs lists the drug ingredient (text), not a code..

Some drugs given at KTPH are combination products, and these combinations were represented by the drug ingredients connected by a hyphen. The OMOP Standard Vocabulary for drugs does not represent combination products in this manner and they do not exist on the `Concept_table`. To match these combinations to the correct concept id in the OMOP concept table, we manually created a table on OHDSI\_KTPH called `manually_mapped_drugs` that contains all the combination products and the corresponding standardized `concept_id` they map to. We use this table to map the drug combinations.

## Step 10: Execute “5.Mapping OBSERVATION\_PERIOD.sql”

This table fills `Observation_period` with data from `visit_occurrence`. The `OBSERVATION_PERIOD` table contains records which uniquely define the spans of time for which a Person is at-risk to have clinical events recorded within the source systems, even if no events in fact are recorded (healthy patient with no healthcare interactions). This table is different from `condition_occurrence`, because it captures the period between the first and last `condition_occurrence` record of a patient.

## Step 11: Execute “6.Mapping CONDITION\_ERA.sql”

This table fills `Condition_Era` with data from `Condition_Occurrence`. A Condition Era is defined as a span of time when the Person is assumed to have a given condition. Similar to



Drug Eras, Condition Eras are chronological periods of Condition Occurrence. Combining individual Condition Occurrences into a single Condition Era serves two purposes:

- It allows aggregation of chronic conditions that require frequent ongoing care, instead of treating each Condition Occurrence as an independent event.
- It allows aggregation of multiple, closely timed doctor visits for the same Condition to avoid double-counting the Condition Occurrences.

## Step 12: Execute “6.Mapping DRUG\_ERA.sql”

This tab fills Drug\_Era with data from Drug\_Exposure. A Drug Era is defined as a span of time when the Person is assumed to be exposed to a particular active ingredient. A Drug Era is not the same as a Drug Exposure: Exposures are individual records corresponding to the source when Drug was delivered to the Person, while successive periods of Drug Exposures are combined under certain rules to produce continuous Drug Eras.

## Step 12: Test the Mapping

We check whether all data was mapped using the test codes on the folder “ETL\_SQL\_CODE”. It should also be easy at this point to write fresh code to test the mapping.

## Code Examples

### 1.Mapping PERSON table.sql

```
/* map patient who have at least a year of birth */
INSERT INTO OHDSI_KTPH.dbo.person (gender_concept_id,
                                   year_of_birth,
                                   month_of_birth,
                                   day_of_birth,
                                   race_concept_id,
                                   location_id,
                                   ethnicity_concept_id,
                                   person_source_value,
                                   gender_source_value,
                                   race_source_value,
                                   ethnicity_source_value)

SELECT
-- person_id is auto-incremented
CASE
    WHEN sex_cd = 'M' THEN 8507
    WHEN sex_cd = 'F' THEN 8532
```

```

    ELSE 8551
    END AS gender_concept_id,
-- 'M' and 'F' are already the OMOP standardized values for sex so we just
map them to their concept_ids
-- since there are few values (3), there is no need to use the
Concept_Relationship table
-- we just include the values manually in the code
DATEPART(YEAR, birth_d) AS year_of_birth,
DATEPART(MONTH, birth_d) AS month_of_birth,
DATEPART(DAY, birth_d) AS day_of_birth,
CASE
    WHEN race_cd = 'C' THEN 38003579 -- Chinese
    WHEN race_cd = 'E' THEN 100 -- Eurasian
    WHEN race_cd = 'I' OR race_cd = 'S' THEN 100 -- Indian or Sikh
    WHEN race_cd = 'J' THEN 38003584 -- Japanese
    WHEN race_cd = 'M' THEN 38003587 -- Malay
    WHEN race_cd = 'N' THEN 8527 -- White
    ELSE 0 -- Other
    END AS race_concept_id,
1, -- location_id for KTPH - check OHDSI_KTPH.dbo.location for other
location_ids
CASE
    WHEN race_cd = 'C' THEN 38003579 -- Chinese
    WHEN race_cd = 'E' THEN 100 -- Eurasian
    WHEN race_cd = 'I' OR race_cd = 'S' THEN 100 -- Indian or Sikh
    WHEN race_cd = 'J' THEN 38003584 -- Japanese
    WHEN race_cd = 'M' THEN 38003587 -- Malay
    WHEN race_cd = 'N' THEN 8527 -- White
    ELSE 0 -- Other
    END AS ethnicity_concept_id,
person_id, -- source value
sex_cd, -- source value
race_cd, -- source value
race_cd-- source value

FROM Sapphire_KTPH.dbo.t_demographics
WHERE DATEPART(YEAR, birth_d) IS NOT NULL
-- each patient on the CMD must have a valid date of birth;

```

## 2.create\_visit\_id\_source\_mapping\_table.sql

```

INSERT INTO OHDSI_KTPH.dbo.visit_occurrence (person_id,

visit_concept_id,

visit_start_date,

visit_start_datetime,

visit_end_date,

visit_end_datetime,

provide_id - does not apply to KTPH*/

visit_source_value,

visit_type_concept_id)

SELECT
    p.person_id,
    CASE
        WHEN te.pat_type_cd = 'I' or te.pat_type_cd = 'EL' THEN 9201 /*
Impatient or Elective Impatient */
        WHEN te.pat_type_cd = 'O' THEN 9202 /* outpatient */
        WHEN te.pat_type_cd = 'E' or te.pat_type_cd = 'EM' THEN 9203 /*
Emergency */
        ELSE 0 /* unknown */
    END AS visit_concept_id,

    CONVERT(date, te.adm_dt),
    CONVERT(time, te.adm_dt),
    CONVERT(date, te.disch_dt),
    CONVERT(time, te.disch_dt),
    te.pat_type_cd,
    44818518

FROM Sapphire_KTPH.dbo.t_encounter te INNER JOIN OHDSI_KTPH.dbo.person p
ON

```

```

p.person_source_value = te.person_id
-- JOIN serves to retrieve the person_id on OHDSI.dbo.person table, and not
the one on Saphire_KTPH.dbo.t_demographics
WHERE te.adm_dt IS NOT NULL
AND te.disch_dt IS NOT NULL
-- each visit record must have valid admission and discharge dates;

```

### 3.create\_visit\_id\_source\_mapping\_table.sql

```

CREATE TABLE OHDSI_KTPH.dbo.visit_id_source_mapping (
/* This table is not part of the OMOP CDM */
/* This table serves to link visit_occurrence_id (the primary key of visit
occurrence) */
/* to encounter_id and pat_type_cd (the primary key of t_encounter) */
    encounter_id VARCHAR(100),
    pat_type_cd VARCHAR (100),
    visit_occurrence_id INTEGER IDENTITY(0,1) PRIMARY KEY)
-- IDENTITY(0,1) is an auto increment function;

```

### 4.filling\_visit\_id\_source\_mapping\_table.sql

```

INSERT INTO OHDSI_KTPH.dbo.visit_id_source_mapping(encounter_id,

    pat_type_cd)

SELECT
    te.encounter_id,
    te.pat_type_cd
    -- primary key of t_encounter
    -- primary key of visit_occurrence is auto_incremented
FROM Saphire_KTPH.dbo.t_encounter te INNER JOIN OHDSI_KTPH.dbo.person p
ON

p.person_source_value = te.person_id
WHERE te.adm_dt IS NOT NULL
AND te.disch_dt IS NOT NULL;

```

### 5.Mapping CONDITION\_OCCURENCE table Part 1.sql

```

WITH [Condition_Selection_1] AS

```

```

(SELECT VO.person_id as person_id,
      VO.visit_start_date as condition_start_date,
      VO.visit_start_datetime as condition_start_datetime,
      VO.visit_end_date as condition_end_date,
      VO.visit_end_datetime as condition_end_datetime,
      44786627 as condition_type_concept_id, -- standardized
concept_id for primary condition
      VO.visit_occurrence_id as visit_occurrence_id,
      PD1.icd_clean as condition_source_value,
      MAX(C.concept_id) as condition_source_concept_id -- we
select max because we group by all other columns
-- this is
to take account for the ICD codes that are in multiple dictionaries

      -- for instance, ICD9 and ICD9CM

FROM Sapphire_KTPH.dbo.t_primary_diagnosis PD1
      INNER JOIN OHDSI_KTPH.dbo.visit_id_source_mapping VISD -- join
with VISD to link the PK of PD2 with the PK of VO
      ON PD1.encounter_id = VISD.encounter_id
      COLLATE SQL_Latin1_General_CP1_CI_AS
      AND PD1.pat_type_cd = VISD.pat_type_cd
      COLLATE SQL_Latin1_General_CP1_CI_AS
      INNER JOIN OHDSI_KTPH.dbo.visit_occurrence VO -- join with VO
to select the visit_id from OHDSI_KTPH.dbo.visit_id and not from
Sapphire_KTPH.dbo.t_encounter
      ON VISD.visit_occurrence_id = VO.visit_occurrence_id
      LEFT OUTER JOIN OHDSI_KTPH.dbo.concept C
      ON PD1.icd_clean = C.concept_code
      COLLATE SQL_Latin1_General_CP1_CI_AS
      AND (C.vocabulary_id LIKE '%ICD%') -- any ICD dictionary (ICD9,
ICD10, ICD9CM, ICD10CM)
      AND C.domain_id = 'Condition' -- condition on domain
      --AND C.invalid_reason NOT IN ('D','U') -- ICD code has no been
Deleted ('D') or Updated ('U')
GROUP BY PD1.icd_clean,
      VO.person_id,
      VO.visit_start_date,
      VO.visit_start_datetime,
      VO.visit_end_date,
      VO.visit_end_datetime,
      VO.visit_occurrence_id,
      PD1.icd_clean

```

```

    ),

[Condition_Selection_11] AS
    (SELECT *
      FROM Condition_Selection_1 CS1
      WHERE CS1.condition_source_concept_id IS NOT NULL),

[Condition_Selection_111] AS
    (SELECT CS11.person_id,
           CR.concept_id_2 as condition_concept_id, -- first cohort
of mapped ICD codes
           CS11.condition_start_date,
           CS11.condition_start_datetime,
           CS11.condition_end_date,
           CS11.condition_end_datetime,
           CS11.condition_type_concept_id,
           CS11.visit_occurrence_id,
           CS11.condition_source_value,
           CS11.condition_source_concept_id
      FROM [Condition_Selection_11] CS11
      LEFT OUTER JOIN OHDSI_KTPH.dbo.CONCEPT_RELATIONSHIP CR
      ON CS11.condition_source_concept_id = CR.concept_id_1
      AND CR.relationship_id = 'Maps to'),

[Condition_Selection_12] AS
    (SELECT person_id,
           condition_start_date,
           condition_start_datetime,
           condition_end_date,
           condition_end_datetime,
           condition_type_concept_id,
           visit_occurrence_id,
           condition_source_value
           -- condition_source_concept_id
      FROM Condition_Selection_1 CS1
      WHERE CS1.condition_source_concept_id IS NULL
      UNION
      SELECT person_id,
           condition_start_date,
           condition_start_datetime,
           condition_end_date,

```

```

        condition_end_datetime,
        condition_type_concept_id,
        visit_occurrence_id,
        condition_source_value
FROM Condition_Selection_111 CS111
WHERE CS111.condition_concept_id IS NULL),

[Condition_Selection_121] AS
(SELECT CS12.person_id,
        -- CR.concept_id_2 as condition_concept_id,
        CS12.condition_start_date,
        CS12.condition_start_datetime,
        CS12.condition_end_date,
        CS12.condition_end_datetime,
        CS12.condition_type_concept_id,
        CS12.visit_occurrence_id,
        CS12.condition_source_value,
        MAX(C.concept_id) as condition_source_concept_id
FROM [Condition_Selection_12] CS12
LEFT OUTER JOIN OHDSI_KTPH.dbo.concept C
ON
    -- ICD code is of the format 'T.D.34' then it becomes of the
format 'D34'
    -- else if it is of the format 'T.D.3412' then it becomes of
the format 'D34.12'
    -- else it stays the same
    (CASE
        WHEN CS12.condition_source_value LIKE 'T._._'
        THEN SUBSTRING (CS12.condition_source_value, 3, 1)
        + SUBSTRING (CS12.condition_source_value, 5, LEN
(CS12.condition_source_value) - 4)
        WHEN CS12.condition_source_value LIKE 'T._._%_%'
-- has at least 3 characters after the second decimal point
        THEN SUBSTRING (CS12.condition_source_value, 3,
1)
        + SUBSTRING (CS12.condition_source_value, 5, 2)
        + '.'
        + SUBSTRING (CS12.condition_source_value, 7, LEN
(CS12.condition_source_value) - 6)
        ELSE LEFT(CS12.condition_source_value,
LEN(CS12.condition_source_value) - 1)
        END = C.concept_code
        COLLATE SQL_Latin1_General_CP1_CI_AS)

```

```

        AND (C.vocabulary_id LIKE 'ICD%')
        AND C.domain_id = 'Condition'
        --AND C.invalid_reason NOT IN ('D','U')
GROUP BY CS12.person_id,
        CS12.condition_start_date,
        CS12.condition_start_datetime,
        CS12.condition_end_date,
        CS12.condition_end_datetime,
        CS12.condition_type_concept_id,
        CS12.visit_occurrence_id,
        CS12.condition_source_value
),

[Condition_Selection_1211] AS
(SELECT CS121.person_id,
        CR.concept_id_2 as condition_concept_id,
        CS121.condition_start_date,
        CS121.condition_start_datetime,
        CS121.condition_end_date,
        CS121.condition_end_datetime,
        CS121.condition_type_concept_id,
        CS121.visit_occurrence_id,
        CS121.condition_source_value,
        CS121.condition_source_concept_id
FROM [Condition_Selection_121] CS121
     LEFT OUTER JOIN OHDSI_KTPH.dbo.CONCEPT_RELATIONSHIP CR
     ON  CS121.condition_source_concept_id = CR.concept_id_1
     AND CR.relationship_id = 'Maps to'
WHERE CS121.condition_source_concept_id IS NOT NULL),

[Condition_Selection_122] AS
(SELECT person_id,
        condition_start_date,
        condition_start_datetime,
        condition_end_date,
        condition_end_datetime,
        condition_type_concept_id,
        visit_occurrence_id,
        condition_source_value
        -- condition_source_concept_id
FROM Condition_Selection_121 CS121
WHERE CS121.condition_source_concept_id IS NULL
UNION

```



```

SELECT person_id,
       condition_start_date,
       condition_start_datetime,
       condition_end_date,
       condition_end_datetime,
       condition_type_concept_id,
       visit_occurrence_id,
       condition_source_value
FROM Condition_Selection_1211 CS1211
WHERE CS1211.condition_concept_id IS NULL),

```

[Condition\_Selection\_1221] AS

```

    (SELECT CS122.person_id,
           CS122.condition_start_date,
           CS122.condition_start_datetime,
           CS122.condition_end_date,
           CS122.condition_end_datetime,
           CS122.condition_type_concept_id,
           CS122.visit_occurrence_id,
           CS122.condition_source_value,
           MAX(C.concept_id) as condition_source_concept_id
    FROM Condition_Selection_122 CS122
    LEFT OUTER JOIN OHDSI_KTPH.dbo.concept C
    ON
        -- ICD code is of the format 'T.D.34' then it becomes of the
format 'D34'
        -- else if it is of the format 'T.D.3412' then it becomes of
the format 'D34.1'
        -- else if it is of any other format, say 'E.1234', it becomes
'E12.3'
        (CASE
            WHEN CS122.condition_source_value LIKE 'T._._'
            THEN SUBSTRING (CS122.condition_source_value, 3,
1)
                + SUBSTRING (CS122.condition_source_value, 5, LEN
(CS122.condition_source_value) - 5) -- 5 not 4!
            WHEN CS122.condition_source_value LIKE 'T._._%._'
            THEN SUBSTRING (CS122.condition_source_value, 3,
1)
                + SUBSTRING (CS122.condition_source_value, 5, 2)
                + '.'
                + SUBSTRING (CS122.condition_source_value, 7, LEN
(CS122.condition_source_value) - 7) -- 7 not 6!

```

```

        ELSE LEFT(CS122.condition_source_value,
LEN(CS122.condition_source_value) - 2)
        END = C.concept_code
        COLLATE SQL_Latin1_General_CP1_CI_AS)
    GROUP BY CS122.person_id,
            CS122.condition_start_date,
            CS122.condition_start_datetime,
            CS122.condition_end_date,
            CS122.condition_end_datetime,
            CS122.condition_type_concept_id,
            CS122.visit_occurrence_id,
            CS122.condition_source_value
    ),

[Condition_Selection_12211] AS
    (SELECT CS1221.person_id,
            CR.concept_id_2 as condition_concept_id,
            CS1221.condition_start_date,
            CS1221.condition_start_datetime,
            CS1221.condition_end_date,
            CS1221.condition_end_datetime,
            CS1221.condition_type_concept_id,
            CS1221.visit_occurrence_id,
            CS1221.condition_source_value,
            CS1221.condition_source_concept_id
    FROM [Condition_Selection_1221] CS1221
        LEFT OUTER JOIN OHDSI_KTPH.dbo.CONCEPT_RELATIONSHIP CR
    ON CS1221.condition_source_concept_id = CR.concept_id_1
        AND CR.relationship_id = 'Maps to'
    WHERE CS1221.condition_source_concept_id IS NOT NULL),

[Condition_Selection_1222] AS
    (SELECT person_id,
            condition_start_date,
            condition_start_datetime,
            condition_end_date,
            condition_end_datetime,
            condition_type_concept_id,
            visit_occurrence_id,
            condition_source_value
            -- condition_source_concept_id
    FROM Condition_Selection_1221 CS1221
    WHERE CS1221.condition_source_concept_id IS NULL

```

```

UNION
SELECT person_id,
       condition_start_date,
       condition_start_datetime,
       condition_end_date,
       condition_end_datetime,
       condition_type_concept_id,
       visit_occurrence_id,
       condition_source_value
FROM Condition_Selection_111 CS12211
WHERE CS12211.condition_concept_id IS NULL),

[Condition_Selection_12221] AS
  (SELECT CS1222.person_id,
         CS1222.condition_start_date,
         CS1222.condition_start_datetime,
         CS1222.condition_end_date,
         CS1222.condition_end_datetime,
         CS1222.condition_type_concept_id,
         CS1222.visit_occurrence_id,
         CS1222.condition_source_value,
         MAX(C.concept_id) as condition_source_concept_id
  FROM Condition_Selection_1222 CS1222
  LEFT OUTER JOIN OHDSI_KTPH.dbo.concept C
  ON
    -- ICD code is of the format 'T.D.34' then it becomes of the
format 'D34'
    -- else if it is of the format 'T.D.3412' then it becomes of
the format 'D34'
    -- else if it is of any other format, say 'E12.34' or
'F45.656', it becomes 'E12' or 'F45.'
    (CASE
      WHEN CS1222.condition_source_value LIKE 'T._._'
      THEN SUBSTRING (CS1222.condition_source_value, 3,
1)
      + SUBSTRING (CS1222.condition_source_value, 5, LEN
(CS1222.condition_source_value) - 5) -- 6 not 4!
      WHEN CS1222.condition_source_value LIKE
'T._._%_%'
      THEN SUBSTRING (CS1222.condition_source_value, 3,
1)
      + SUBSTRING (CS1222.condition_source_value, 5, 2)
    )
    --- map to highest level

```

```

        ELSE (CASE
                WHEN
LEN(CS1222.condition_source_value) > 3
                THEN
LEFT(CS1222.condition_source_value, LEN(CS1222.condition_source_value) - 3)
                ELSE
LEFT(CS1222.condition_source_value, LEN(CS1222.condition_source_value) - 2)
                END)
        END = C.concept_code
        COLLATE SQL_Latin1_General_CP1_CI_AS)
GROUP BY CS1222.person_id,
        CS1222.condition_start_date,
        CS1222.condition_start_datetime,
        CS1222.condition_end_date,
        CS1222.condition_end_datetime,
        CS1222.condition_type_concept_id,
        CS1222.visit_occurrence_id,
        CS1222.condition_source_value
),

[Condition_Selection_122211] AS
(SELECT CS12221.person_id,
        CR.concept_id_2 as condition_concept_id,
        CS12221.condition_start_date,
        CS12221.condition_start_datetime,
        CS12221.condition_end_date,
        CS12221.condition_end_datetime,
        CS12221.condition_type_concept_id,
        CS12221.visit_occurrence_id,
        CS12221.condition_source_value,
        CS12221.condition_source_concept_id
FROM [Condition_Selection_12221] CS12221
LEFT OUTER JOIN OHDSI_KTPH.dbo.CONCEPT_RELATIONSHIP CR
ON CS12221.condition_source_concept_id = CR.concept_id_1
AND CR.relationship_id = 'Maps to'),

[Condition_Selection_12222] AS
(SELECT person_id,
        condition_start_date,
        condition_start_datetime,
        condition_end_date,
        condition_end_datetime,

```

```

        condition_type_concept_id,
        visit_occurrence_id,
        condition_source_value
        -- condition_source_concept_id
FROM Condition_Selection_12221 CS12221
WHERE CS12221.condition_source_concept_id IS NULL
UNION
SELECT person_id,
        condition_start_date,
        condition_start_datetime,
        condition_end_date,
        condition_end_datetime,
        condition_type_concept_id,
        visit_occurrence_id,
        condition_source_value
FROM Condition_Selection_111 CS122211
WHERE CS122211.condition_concept_id IS NULL),

[Condition_Selection_122221] AS
    (SELECT CS12222.person_id,
        CS12222.condition_start_date,
        CS12222.condition_start_datetime,
        CS12222.condition_end_date,
        CS12222.condition_end_datetime,
        CS12222.condition_type_concept_id,
        CS12222.visit_occurrence_id,
        CS12222.condition_source_value,
        MAX(C.concept_id) as condition_source_concept_id
    FROM Condition_Selection_12222 CS12222
    LEFT OUTER JOIN OHDSI_KTPH.dbo.concept C
    ON
        -- if the ICD code is of the format 'E.12345' then it becomes
'E12.345'
        -- else it stays the same
        (CASE
        WHEN LEN (condition_source_value) > 6
        THEN LEFT(condition_source_value, 1)
            + SUBSTRING (condition_source_value,3,2)
            + '.'
            + SUBSTRING (condition_source_value, 5, LEN
(condition_source_value) - 6)
        ELSE condition_source_value
        END

```

```

        )= C.concept_code
        COLLATE SQL_Latin1_General_CP1_CI_AS
    GROUP BY CS12222.person_id,
            CS12222.condition_start_date,
            CS12222.condition_start_datetime,
            CS12222.condition_end_date,
            CS12222.condition_end_datetime,
            CS12222.condition_type_concept_id,
            CS12222.visit_occurrence_id,
            CS12222.condition_source_value
    ),

[Condition_Selection_1222211] AS
    (SELECT CS122221.person_id,
            CR.concept_id_2 as condition_concept_id,
            CS122221.condition_start_date,
            CS122221.condition_start_datetime,
            CS122221.condition_end_date,
            CS122221.condition_end_datetime,
            CS122221.condition_type_concept_id,
            CS122221.visit_occurrence_id,
            CS122221.condition_source_value,
            CS122221.condition_source_concept_id
    FROM [Condition_Selection_122221] CS122221
        LEFT OUTER JOIN OHDSI_KTPH.dbo.CONCEPT_RELATIONSHIP CR
        ON CS122221.condition_source_concept_id = CR.concept_id_1
        AND CR.relationship_id = 'Maps to')

INSERT INTO OHDSI_KTPH.dbo.condition_occurrence(person_id,

condition_concept_id,

condition_start_date,

condition_start_datetime,

condition_end_date,

condition_end_datetime,

condition_type_concept_id,

```

```
visit_occurrence_id,
```

```
condition_source_value,
```

```
condition_source_concept_id)
```

```
SELECT * FROM Condition_Selection_111
WHERE condition_concept_id IS NOT NULL -- because sometimes a concept_id
does not have a mapping on the CONCEPT RELATIONSHIP TABLE
UNION ALL
SELECT * FROM Condition_Selection_1211
WHERE condition_concept_id IS NOT NULL -- because sometimes a concept_id
does not have a mapping on the CONCEPT RELATIONSHIP TABLE
UNION ALL
SELECT * FROM Condition_Selection_12211
WHERE condition_concept_id IS NOT NULL -- because sometimes a concept_id
does not have a mapping on the CONCEPT RELATIONSHIP TABLE
UNION ALL
SELECT * FROM Condition_Selection_122211
WHERE condition_concept_id IS NOT NULL -- because sometimes a concept_id
does not have a mapping on the CONCEPT RELATIONSHIP TABLE
UNION ALL
SELECT * FROM Condition_Selection_1222211
WHERE condition_concept_id IS NOT NULL -- because sometimes a concept_id
does not have a mapping on the CONCEPT RELATIONSHIP TABLE
```

## Mapping DRUG\_EXPOSURE table.sql

```
WITH Drug_Selection_11 AS
  (SELECT V0.person_id as person_id,
         CONVERT (date, IM0.drug_order_start_dt) as
drug_exposure_start_date,
         CONVERT (time, IM0.drug_order_start_dt) as
drug_exposure_start_time,
         CONVERT (date, IM0.drug_order_end_dt) as
drug_exposure_end_date,
         CONVERT (time, IM0.drug_order_end_dt) as
drug_exposure_end_time,
         581373 as drug_type_concept_id, -- Physician administered
drug
```

```

        VO.visit_occurrence_id as visit_occurrence_id,
        IMO.drug_inpat_order_ingred_clean as drug_source_value,
-- source value
        MIN(C.concept_id) as drug_source_concept_id, -- we search
min because we group by all other columns

        -- this is because some drugs have the same name one are
registered more than once
        route_adm_cd as route_source_value,
        strength as dose_unit_source_value

FROM Sapphire_KTPH.dbo.t_inpatient_med_order IMO
LEFT OUTER JOIN OHDSI_KTPH.dbo.visit_id_source_mapping VISD --
join with VISD to link the PK of IMO with the PK of VO
ON IMO.encounter_id = VISD.encounter_id
COLLATE SQL_Latin1_General_CP1_CI_AS
AND IMO.pat_type_cd = VISD.pat_type_cd
COLLATE SQL_Latin1_General_CP1_CI_AS
LEFT OUTER JOIN OHDSI_KTPH.dbo.visit_occurrence VO -- join with
VO to select the vist_id from OHDSI_KTPH.dbo.visit_id and not from
Sapphire_KTPH.dbo.t_encoutner
ON VISD.visit_occurrence_id = VO.visit_occurrence_id
LEFT OUTER JOIN OHDSI_KTPH.dbo.concept C
ON IMO.drug_inpat_order_ingred_clean = C.concept_name
COLLATE SQL_Latin1_General_CP1_CI_AS
/* ON CAST (IMO.drug_inpat_order_clean_cd as VARCHAR(100)) =
C.concept_code */
AND C.domain_id = 'Drug' -- condition on domain
-- AND C.invalid_reason NOT IN ('U', 'D') --AND
C.invalid_reason NOT IN ('D','U') -- ICD code has no been Deleted ('D') or
Updated ('U')
WHERE C.concept_id IS NOT NULL
GROUP BY VO.person_id,
        CONVERT (date, IMO.drug_order_start_dt),
        CONVERT (time, IMO.drug_order_start_dt),
        CONVERT (date, IMO.drug_order_end_dt),
        CONVERT (time, IMO.drug_order_end_dt),
        VO.visit_occurrence_id,
        IMO.drug_inpat_order_ingred_clean,
        route_adm_cd,
        strength),

```

Drug\_Selection\_12 AS



```

    (SELECT person_id,
           CR.concept_id_2 as drug_concept_id,
           drug_exposure_start_date,
           drug_exposure_start_time,
           drug_exposure_end_date,
           drug_exposure_end_time,
           581373 as drug_type_concept_id, -- Physician administered
drug
           visit_occurrence_id,
           drug_source_value,
           drug_source_concept_id,
           route_source_value,
           dose_unit_source_value

    FROM Drug_Selection_11 DS11
         LEFT OUTER JOIN OHDSI_KTPH.dbo.CONCEPT_RELATIONSHIP CR
         ON DS11.drug_source_concept_id = CR.concept_id_1
         AND CR.relationship_id = 'Maps to'
    WHERE DS11.drug_source_concept_id IS NOT NULL),

-- Drug_Selection_12 contains 90% of the mapped data
-- the other 10% is all the compositions of different drugs
-- Dr.Sung mapped all these drugs manually on the table
OHDSI_KTPH.dbo.manually_mapped_drugs MMD
-- in Drug_Selection_2 we match the compositions to their manually mapped
standardized codes

Drug_Selection_2 AS
    (SELECT V0.person_id as person_id,
           CR.concept_id_2 as drug_concept_id,
           CONVERT (date, IMO.drug_order_start_dt) as
drug_exposure_start_date,
           CONVERT (time, IMO.drug_order_start_dt) as
drug_exposure_start_time,
           CONVERT (date, IMO.drug_order_end_dt) as
drug_exposure_end_date,
           CONVERT (time, IMO.drug_order_end_dt) as
drug_exposure_end_time,
           581373 as drug_type_concept_id, -- Physician administered
drug
           V0.visit_occurrence_id as visit_occurrence_id,
           IMO.drug_inpat_order_ingred_clean as drug_source_value,

```

```

        CR.concept_id_2 as drug_source_concept_id,
        route_adm_cd as route_source_value,
        strength as dose_unit_source_value

FROM Sapphire_KTPH.dbo.t_inpatient_med_order IMO
  LEFT OUTER JOIN OHDSI_KTPH.dbo.visit_id_source_mapping VISD
ON IMO.encounter_id = VISD.encounter_id
  COLLATE SQL_Latin1_General_CP1_CI_AS
  AND IMO.pat_type_cd = VISD.pat_type_cd
  COLLATE SQL_Latin1_General_CP1_CI_AS
  LEFT OUTER JOIN OHDSI_KTPH.dbo.visit_occurrence VO
ON VISD.visit_occurrence_id = VO.visit_occurrence_id
  LEFT OUTER JOIN OHDSI_KTPH.dbo.manually_mapped_drugs MMD
ON IMO.drug_inpat_order_ingred_clean = MMD.clean_drug_name
  COLLATE SQL_Latin1_General_CP1_CI_AS
  AND IMO.drug_inpat_order_unclean = MMD.[unclean_drug_name ]
  COLLATE SQL_Latin1_General_CP1_CI_AS
  LEFT OUTER JOIN OHDSI_KTPH.dbo.CONCEPT_RELATIONSHIP CR
ON MMD.[OMOP Standard Code] = CR.concept_id_1
  AND CR.relationship_id = 'Maps to'

WHERE IMO.drug_inpat_order_ingred_clean NOT IN (SELECT
DS12.drug_source_value

FROM Drug_Selection_12 DS12)
  AND MMD.[OMOP Standard Code] IS NOT NULL)

INSERT INTO OHDSI_KTPH.dbo.drug_exposure (person_id,

drug_concept_id,

drug_exposure_start_date,

drug_exposure_start_datetime,

drug_exposure_end_date,

drug_exposure_end_datetime,

drug_type_concept_id,

visit_occurrence_id,

```

```

drug_source_value,

drug_source_concept_id,

route_source_value,

dose_unit_source_value)

SELECT * FROM Drug_Selection_12 DS12
    WHERE DS12.person_id IS NOT NULL
    AND DS12.visit_occurrence_id IS NOT NULL
    AND DS12.drug_concept_id IS NOT NULL
    AND DS12.drug_exposure_start_date IS NOT NULL
    AND DS12.drug_exposure_end_date IS NOT NULL
    AND DS12.drug_exposure_start_time IS NOT NULL
    AND DS12.drug_exposure_end_time IS NOT NULL

UNION ALL

SELECT *
FROM Drug_Selection_2 DS2
    WHERE DS2.person_id IS NOT NULL
    AND DS2.visit_occurrence_id IS NOT NULL
    AND DS2.drug_concept_id IS NOT NULL
    AND DS2.drug_exposure_start_date IS NOT NULL
    AND DS2.drug_exposure_end_date IS NOT NULL
    AND DS2.drug_exposure_start_time IS NOT NULL
    AND DS2.drug_exposure_end_time IS NOT NULL

```

## Mapping OBSERVATION\_PERIOD.sql

```

-- the observation period is the difference in time
-- between the first visit_occurrence of a certain patient
-- and his/her last visit_occurrence.
INSERT INTO OHDSI_KTPH.dbo.observation_period
SELECT VO.person_id AS person_id,
    MIN(VO.visit_start_date) as observation_period_start_date,
    MAX(VO.visit_end_date) as observation_period_end_date,
    38000280 as period_type_concept_id --Observation recorded from EHR
FROM OHDSI_KTPH.dbo.visit_occurrence VO

```

```
GROUP BY VO.person_id
```

## 6.Mapping CONDITION\_ERA.sql

```
WITH cteConditionTarget
    (CONDITION_OCCURRENCE_ID,
     PERSON_ID,
     CONDITION_CONCEPT_ID,
     CONDITION_TYPE_CONCEPT_ID,
     CONDITION_START_DATE,
     CONDITION_END_DATE) AS

    (SELECT co.CONDITION_OCCURRENCE_ID,
            co.PERSON_ID,
            co.CONDITION_CONCEPT_ID,
            co.CONDITION_TYPE_CONCEPT_ID,
            co.CONDITION_START_DATE,
            COALESCE(co.CONDITION_END_DATE, DATEADD(day,1,CONDITION_START_DATE))
    AS CONDITION_END_DATE
    FROM OHDSI_KTPH.dbo.CONDITION_OCCURRENCE co),

cteEndDates (PERSON_ID,
              CONDITION_CONCEPT_ID,
              END_DATE) AS -- the magic
    (SELECT PERSON_ID,
            CONDITION_CONCEPT_ID,
            DATEADD (day,-30,EVENT_DATE) AS END_DATE -- unpad the end
    date
    FROM
    (
        SELECT PERSON_ID,
            CONDITION_CONCEPT_ID,
            EVENT_DATE,
            EVENT_TYPE,
            MAX(START_ORDINAL) OVER
                (PARTITION BY PERSON_ID, CONDITION_CONCEPT_ID
                ORDER BY EVENT_DATE, EVENT_TYPE
                ROWS UNBOUNDED PRECEDING) as START_ORDINAL,
        -- this pulls the current START down from the prior rows so that the NULLs
        from the END DATES will contain a value we can compare with
```

```

        ROW_NUMBER() OVER (PARTITION BY PERSON_ID,
CONDITION_CONCEPT_ID
                                ORDER BY EVENT_DATE,
EVENT_TYPE) AS OVERALL_ORD -- this re-numbers the inner UNION so all rows
are numbered ordered by the event date
        FROM
        (
        -- select the start dates, assigning a row number to each
        SELECT PERSON_ID,
                CONDITION_CONCEPT_ID,
                CONDITION_START_DATE AS EVENT_DATE,
                -1 as EVENT_TYPE,
                ROW_NUMBER() OVER (PARTITION BY PERSON_ID,
CONDITION_CONCEPT_ID
                                ORDER BY
CONDITION_START_DATE) as START_ORDINAL
        FROM cteConditionTarget

        UNION ALL

        -- pad the end dates by 30 to allow a grace period for
        overlapping ranges.
        SELECT PERSON_ID,
                CONDITION_CONCEPT_ID,
                DATEADD(day,30,CONDITION_END_DATE),
                1 as EVENT_TYPE,
                NULL
        FROM cteConditionTarget
        ) RAWDATA
    ) E

    WHERE (2 * E.START_ORDINAL) - E.OVERALL_ORD = 0
),

cteConditionEnds (PERSON_ID,
                CONDITION_CONCEPT_ID,
                CONDITION_TYPE_CONCEPT_ID,
                CONDITION_START_DATE, ERA_END_DATE) as
(select c.PERSON_ID,
        c.CONDITION_CONCEPT_ID,
        c.CONDITION_TYPE_CONCEPT_ID,

```

```

        c.CONDITION_START_DATE,
        MIN(e.END_DATE) as ERA_END_DATE
FROM cteConditionTarget c
JOIN cteEndDates e
ON c.PERSON_ID = e.PERSON_ID
   and c.CONDITION_CONCEPT_ID = e.CONDITION_CONCEPT_ID
   and e.END_DATE >= c.CONDITION_START_DATE
GROUP BY
c.PERSON_ID,
c.CONDITION_CONCEPT_ID,
c.CONDITION_TYPE_CONCEPT_ID,
c.CONDITION_START_DATE )

INSERT INTO OHDSI_KTPH.dbo.condition_era (person_id,
condition_concept_id,
condition_era_start_date,
condition_era_end_date,
condition_occurrence_count)
SELECT person_id,
        CONDITION_CONCEPT_ID,
        min(CONDITION_START_DATE) as CONDITION_ERA_START_DATE,
        ERA_END_DATE as CONDITION_ERA_END_DATE,
        COUNT(*) as CONDITION_OCCURRENCE_COUNT
FROM cteConditionEnds
GROUP BY person_id, CONDITION_CONCEPT_ID, CONDITION_TYPE_CONCEPT_ID,
ERA_END_DATE
ORDER BY person_id, CONDITION_CONCEPT_ID

```

## References

- ➔ Christian Reich, Patrick Ryan, Rimma Belenkaya, Karthik Natarajan and Clair Blacketer. (November 2017). OMOP Common Data Model Specifications
- ➔ SAPHIRE Schema - Data Model v5
- ➔ Qianli Ma, Erica Voss, Chris Knoll, Ajit Londhe. (March 2017). Common Data Model (CDM v5.0) ETL Mapping Specification for Optum Extended SES & Extended DOD (Optum Clinformatics DataMart v7.0)
- ➔ SAPHIRE Data Schema Description (as of 13 Feb 2015)