CSCI 59: Database Project

PHARMACY CARE MANAGEMENT FHIR DATABASE

ERIN REBHOLZ, SPRING 2024, CSCI-59

RECORDED PRESENTATION:

https://www.youtube.com/watch?v=HoeGPjimgDs

Background Information

Database Purpose:

Pharmacies are increasingly interested in getting broader access to patient data that can better support pharmacy patient care.

With the rise of Health Information Networks(HINs) and Quality Health Information Networks (QHIN)s and adoption of Fast Healthcare Interoperability Resource (FHIR) data exchange standards, **patient clinical data is becoming more broadly available for systematic use within the pharmacy practice**.

Dataset Background:

This project will **leverage a synthetic patient data set, SyntheaMass to parse pharmacy relevant patient data into a standard form** that could be **consumed by a pharmacy application.**

This project will help me personally **become more informed about FIHR based data exchanges** and to **assess the ease of getting access to clinical data on behalf of pharmacy customers.**

References:

Synthea Mass Data Set: Downloads | Synthea (mitre.org); https://github.com/syntheticmass

FIHR Background: Note: FHIR standard that aligns with the date (2021) of the SyntheaMass data set: https://hl7.org/fhir/R4/index.html

Background Information

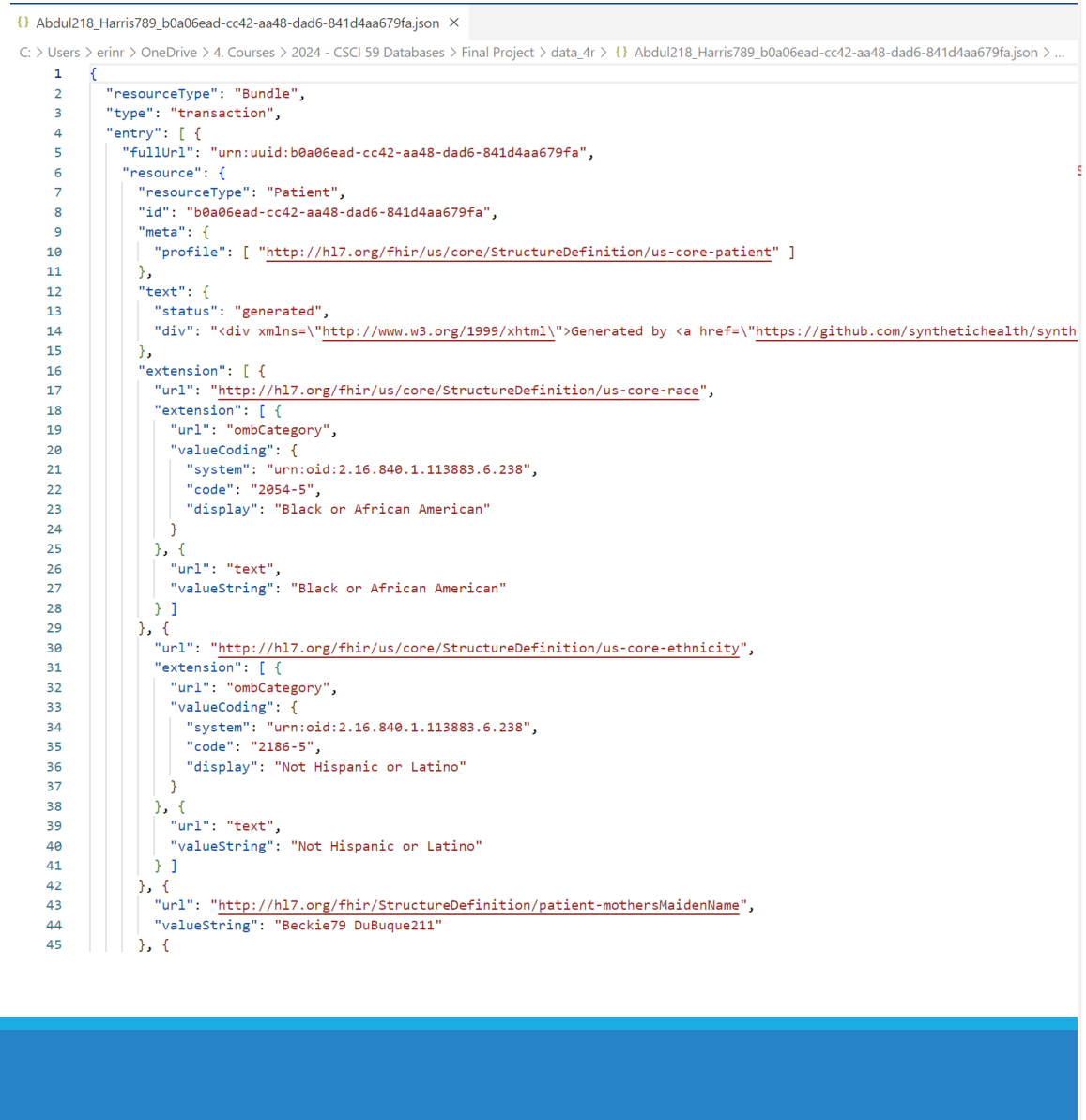
Choice of Database: MySQL

The SyntheaMass **FHIR based formats can be easily parsed into a SQL, normalized format,** based on ‘resoureType’ field to form tables.

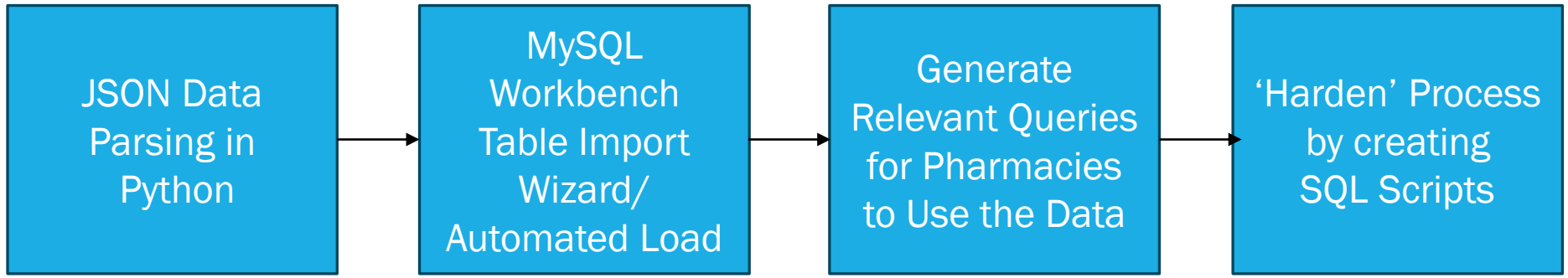
Pharmacies predominantly use SQL data systems to run practices through pharmacy management systems.

A **SQL format will integrate better with existing systems**, including business intelligence tools that are used for patient care assessment.

FHIR Patient File Example (JSON Format):



Project Arc:



Key Outputs:

.csv files of relevant data tables and fields

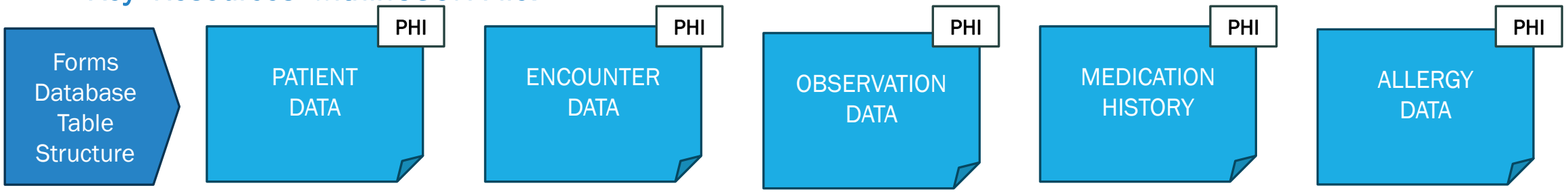
Patient record for Insert Into sql statements

Entity Relationship Diagram of Design

Diabetic Patient Profile

The Synthea data set includes a ‘coherent’ view of a patient over their lifetime of care and includes information about patient encounters, biometrics, payments, and more

Key ‘Resources’ within JSON File:



Key Linking IDs:

PATIENT DATA	ENCOUNTER DATA	OBSERVATION DATA	MEDICATION HISTORY	ALLERGY DATA
patient ID	patient_id	patient_id	patient_id	patient_id
		observation_id	rx_id	allergy_id
	encounter_id	encounter_id	encounter_id	
	provider_id (NPI)	provider_id (NPI)	provider_id (NPI)	

Relevance:
Tracks Patient Level data elements

Used to link to pharmacy management system source

Relevance:
Tracks provider level encounter data with the patient, which generates observations and medications

Relevance:
Biometric/lab values that are important to tracking patient progress on drug or pharmacy care programs

Initial set of metrics might focus on patient height, weight, blood pressure, A1-C, blood glucose readings

Relevance:
Medication history to track potential medication interactions within pharmacy

Relevance:
May have implications for medication allergies

Initial FHIR Data Parsing was done using Python to create .csv files that could be imported into the SQL database as test data

Patient Table – JSON to .csv Code Example

Patient Data

```
[6]: def patient_data_parser(df_full):
    #Create patient data data frame
    patient_data = df_full[df_full['resource_resourceType'] == 'Patient'].reset_index()
    pat_cols = ['resource_resourceType', 'id', 'resource_name', 'resource_telecom',
           'resource_gender', 'resource_birthDate', 'resource_deceasedDateTime',
           'resource_address','resource_subject_reference']
    patient_df = patient_data.loc[:,pat_cols]

    #Normalize to deceased date (strip time)
    patient_df['resource_deceasedDateTime_norm'] = ['1000-01-01' if type(ii)!=(str) else ii[:10] for ii in  patient_df['resource_deceasedDateTime']]
    #Parse Resource Address into Sub-fields
    patient_df['resource_address_line'] = [pd.json_normalize(x).loc[0,'line'][0] for x in patient_df['resource_address']]
    patient_df['resource_address_city'] = [pd.json_normalize(x).loc[0,'city'] for x in patient_df['resource_address']]
    patient_df['resource_address_state'] = [pd.json_normalize(x).loc[0,'state'] for x in patient_df['resource_address']]
    ### Note that not all entries had 'postal code' so we need to assign "" in the second line
    patient_df['resource_address_postalCode'] = [pd.json_normalize(x).iloc[0,4] for x in patient_df['resource_address']]
    patient_df['resource_address_postalCode'] = ["" if x == 'US' else x for x in patient_df['resource_address_postalCode']]
    patient_df['resource_address_country'] = [pd.json_normalize(x).loc[0,'country'] for x in patient_df['resource_address']]

    #Parse Resource Name into sub-fields
    patient_df['resource_name_given'] = [pd.json_normalize(x).iloc[0,1] for x in patient_df['resource_name']]
    patient_df['resource_name_family'] = [pd.json_normalize(x).iloc[0,2][0] for x in patient_df['resource_name']] #Enclosed in br
    #Extract phone info from telecom
    patient_df['resource_telecom_phone'] = [pd.json_normalize(x).iloc[0,1] for x in patient_df['resource_telecom']]


    #Reorder columns and get rid of unnecessary columns
    cols = [ 'id', 'resource_name_given','resource_name_family',
           'resource_gender', 'resource_birthDate', 'resource_deceasedDateTime',
            'resource_address_line', 'resource_address_city',
           'resource_address_state', 'resource_address_postalCode',
           'resource_address_country',  'resource_telecom_phone']

    patient_df = patient_df.loc[:,[ 'id', 'resou
           'resource_gender', 'resource_birthDat
            'resource_address_line', 'resource_a
           'resource_address_state', 'resource_a
           'resource_address_country',  'resourc

    #rename columns to friendly names
    patient_df.columns = [ 'patient_id', 'given_
           'gender', 'birth_date', 'deceased_dat
            'street_address', 'city',
           'state', 'postal_code',
           'country',  'phone']

    return patient_df
```

Iterate Formatting And Normalization To More Easily Import to SQL

555 Patient Data Samples

```
[7]: patient_df = patient_data_parser(df_full)

[8]: #check Encounter_ID is unique
     len(patient_df), len(patient_df.patient_id.unique())

[8]: (555, 555)

[9]: patient_df.to_csv('tables/patient_table.csv', index = False)

[10]: patient_df.head()
```

[10]:

	patient_id	given_name	family_name	gender	birth_date	deceased_date	street_address	city	state	postal_code	country	phone
0	b0a06ead-cc42-aa48-dad6-841d4aa679fa	Harris789	Abdul218	male	1952-12-05	2002-02-01	572 Keebler Esplanade	Boston	MA	02134	US	555-443-1041
1	ccfc4db2-2026-7adb-3db0-33f3828140bb	Schmidt332	Abe604	male	1976-06-06	1000-01-01	768 Mertz Camp	Peabody	MA	01940	US	555-345-5342
2	31a2e8ec-69fc-8a71-3ab6-[illegible]	DuBuque211	Adelaida985	female	1917-05-15	2017-02-18	848 Casper Way	Quincy	MA	02169	US	555-925-4660
3	71a8b156-760b-df6b-859e-eefc7932a526	Hintz995	Adriana394	female	2014-11-15	1000-01-01	455 Bayer Rapid Suite 75	Boston	MA	02120	US	555-237-5240
4	76b289fd-e825-734c-8446-316f59643593	Gerhold939	Agueda283	female	1985-03-03	1000-01-01	733 Bednar Grove Apt 36	Boston	MA	02115	US	555-831-3391

.csv output example

Encounter 27,812 records

```
def encounter_data_parser(df_full):

    enc_data = df_full[df_full['resource_resourceType'] == 'Encounter'].reset_index()

    #Select columns for exploration
    enc_cols = ['id','resource_class_code','resource_type','resource_subject_reference',
                'resource_participant','resource_period_start','resource_period_end','resource_location']

    enc_df = enc_data.loc[:, enc_cols]

    #normalize date times to dates
    enc_df['start_date'] = [ii if type(ii)!=(str) else ii[:10] for ii in  enc_df.resource_period_start]
    enc_df['end_date'] = [ii if type(ii)!=(str) else ii[:10] for ii in  enc_df.resource_period_end]

    #Get info
    enc_df['snomed_code'] = [pd.json_normalize(pd.json_normalize(x).loc[0,'coding'][0])['code'][0] for x in enc_df['resource_type']]
    enc_df['snomed_text'] = [pd.json_normalize(pd.json_normalize(x).loc[0,'coding'][0])['display'][0] for x in enc_df['resource_type']]

    #Provider info
    enc_df['provider_NPI'] = [pd.json_normalize(x)['individual.reference'][0][-10:]  for x in enc_df.resource_participant]
    enc_df['provider_name'] = [pd.json_normalize(x)['individual.display'][0]  for x in enc_df.resource_participant]

    #Facility info
    enc_df['facility_name'] = [pd.json_normalize(x)['location.display'][0]  for x in enc_df.resource_location]

    #Patient id
    enc_df['patient_id'] = [ii[9:] for ii in enc_df.resource_subject_reference]

    #Drop and remove column names
    enc_df = enc_df.loc[:,enc_df_cols]

    #Rename columns to friendly names
    enc_df.columns = ['encounter_id', 'patient_id', 'start_date', 'end_date',
                      'snomed_code', 'snomed_text', 'provider_NPI', 'provider_name',
                      'facility_name','facility_code']
    return enc_df
```

Observations 131,703 records

```
def obs_data_parser(df_full):

    #Create observation data data frame
    obs_data = df_full[df_full['resource_resourceType'] == 'Observation'].reset_index()

    #Select columns for exploration
    obs_cols = [1,2,7,18,23,29,32,35,36,47,65,72,73,75]
    obs_df = obs_data.iloc[:, obs_cols]

    #Normalize date
    obs_df['effective_date'] = [ii if type(ii)!=(str) else ii[:10] for ii in  obs_df.resource_effectiveDateTime]

    #Truncate Encounter and Patient Reference Keys to Link Back to Other Table's Formats
    obs_df['encounter_id'] = [ii[9:] for ii in obs_df.resource_encounter_reference]
    obs_df['patient_id'] = [ii[9:] for ii in obs_df.resource_subject_reference]

    #Observation code (strip unneeded data and normalize json dictionary stuff
    obs_df['obs_code'] = [pd.json_normalize(pd.json_normalize(x).iloc[0,0]).iloc[0,1] for x in obs_df['resource_category']]


    #Reshuffle columns and remove not needed columns)
    obs_cols = ['id', 'encounter_id', 'patient_id', 'effective_date',
           'obs_code','resource_code_text',
           'resource_valueQuantity_value', 'resource_valueQuantity_unit',
           'resource_valueQuantity_code']

    obs_df = obs_df.loc[:,obs_cols]

    #rename columns to friendly names
    obs_df.columns = ['obs_id', 'encounter_id', 'patient_id', 'effective_date', 'obs_code',
                      'code_text', 'quantity_value',
                      'quantity_unit', 'quantity_code']

    return obs_df
```

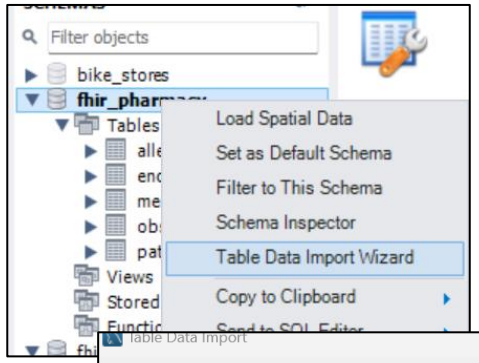
Med History 24,256 records

```
def medication_data_parser(df_full):
    med_data = df_full[df_full['resource_resourceType'] == 'MedicationRequest'].reset_index()

    #Select columns for exploration
    med_cols_raw = ['id','resource_status','resource_subject_reference','resource_encounter_reference',
                    'resource_intent','resource_medicationCodeableConcept_text', 'resource_authoredOn',
                    'resource_requester_reference','resource_prescription_reference','patientid',
                    'resource_medicationCodeableConcept_codingcodes','resource_dosageInstruction']

    med_df = med_data.loc[:, med_cols_raw]

    #normalize Date times to dates
    med_df['request_date'] = [ii[:10] for ii in  med_df.resource_authoredOn]

    #Truncate Encounter and Patient Reference Keys to Link Back to Other Table's Formats
    med_df['encounter_id'] = [ii[10:] for ii in med_df.resource_encounter_reference]
    med_df['patient_id'] = [ii[10:] for ii in med_df.resource_subject_reference]

    #rx Name
    med_df['resource_code'] = med_df.resource_medicationCodeableConcept_codingcodes.astype('str').str.strip('[').str.strip(']').str.strip("'")
    med_df['resource_text'] =  med_df.resource_medicationCodeableConcept_text.replace(',',' ')

    #MPI
    med_df['provider_id'] = [x[-10:] for x in med_df.resource_requester_reference]


    ###Dosing Instructions
    text = []
    snomed_code = []
    snomed_text = []
    repeat_freq = []
    repeat_period = []
    repeat_period_unit = []
    as_needed = []

    for ii in range(len(med_df.resource_dosageInstruction)):
        try:
            helper = med_df.resource_dosageInstruction[ii][0]['text']
            text.append(helper)
        except:
            text.append("")

        try:
            helper = med_df.resource_dosageInstruction[ii][0]['timing']
            helper1 = helper['repeat']['frequency']
```

Allergy 499 records

```
def allergy_data_parser(df_full):

    #Create allergy data frame
    al_data = df_full[df_full['resource_resourceType'] == 'AllergyIntolerance'].reset_index()

    #Select columns for exploration
    al_cols_raw = ['resource_resourceType', 'id', 'request_url', 'resource_type',
           'resource_category', 'resource_patient_reference',
           'resource_criticality', 'resource_code_text', 'resource_recordedDate',
           'resource_reaction']

    al_df = al_data.loc[:, al_cols_raw]

    #normalize Date times to dates
    al_df['recorded_date'] = [ii[:10] for ii in  al_df.resource_recordedDate]

    #Truncate Patient Reference Keys to Link Back to Other Table's Formats
    al_df['patient_id'] = [ii[9:] for ii in al_df.resource_patient_reference]

    al_df['resource_category'] = al_df.resource_category.astype('str').str.strip('[').str.strip(']').str.strip("'")

    ## Get allergy symptoms from nested JSON column
    text = []

    for ii in range(len(al_df.resource_reaction)):
        try:
            helper = ""
            for i in range(len(al_df.iloc[ii,0])):
                helper = helper + (f"{i+1}: {al_df.iloc[ii,9][i]['manifestation'][0]['coding'][0]['display']} ")
            text.append(helper)
        except:
            text.append("")

    al_df['reaction'] = text

    #Reorder columns and omit unnecessary values:
    al_cols = ['id', 'patient_id', 'recorded_date','resource_category', 'resource_code_text', 'reaction', 'resource_criticality']

    al_df = al_df.loc[:,al_cols]

    al_df.columns = ['allergy_id', 'patient_id', 'recorded_date','allergy_category', 'allergy_details', 'reaction', 'allergy_criticality']

    return al_df
```

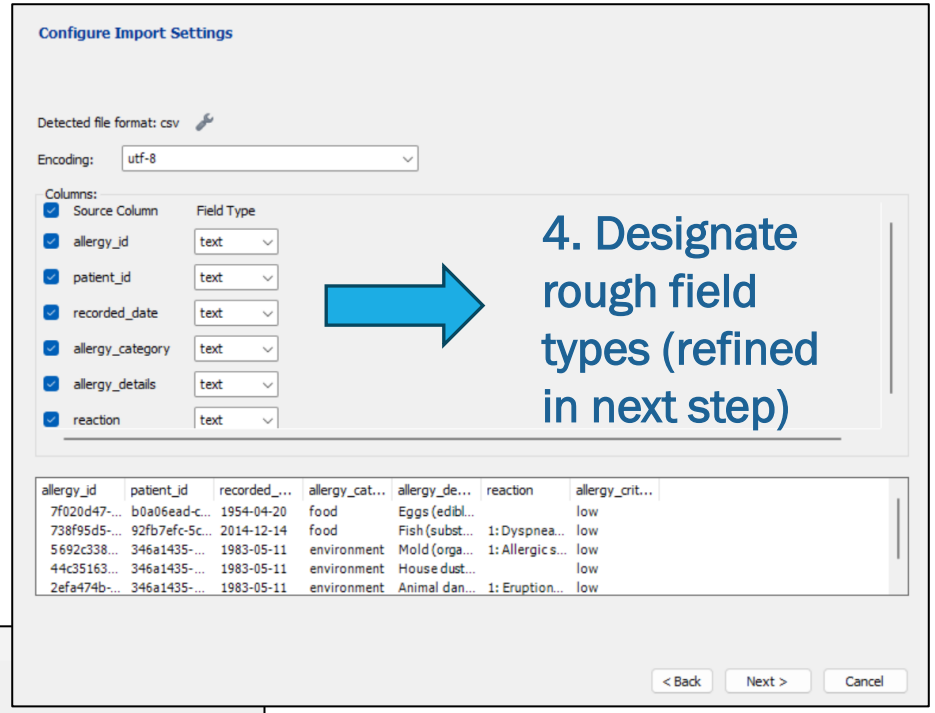
The table import wizard functionality was used to jump start table creation.



Right click on schema and select wizard

Select file to use as table basis

Name new table



4. Designate rough field types (refined in next step)

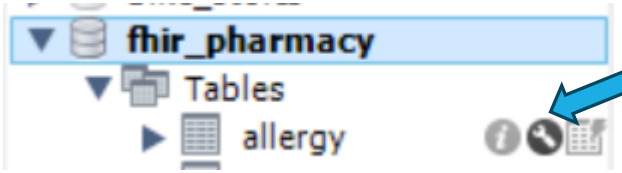
Table fields are catalogued and can be used for automating creation process:

Table: allergy

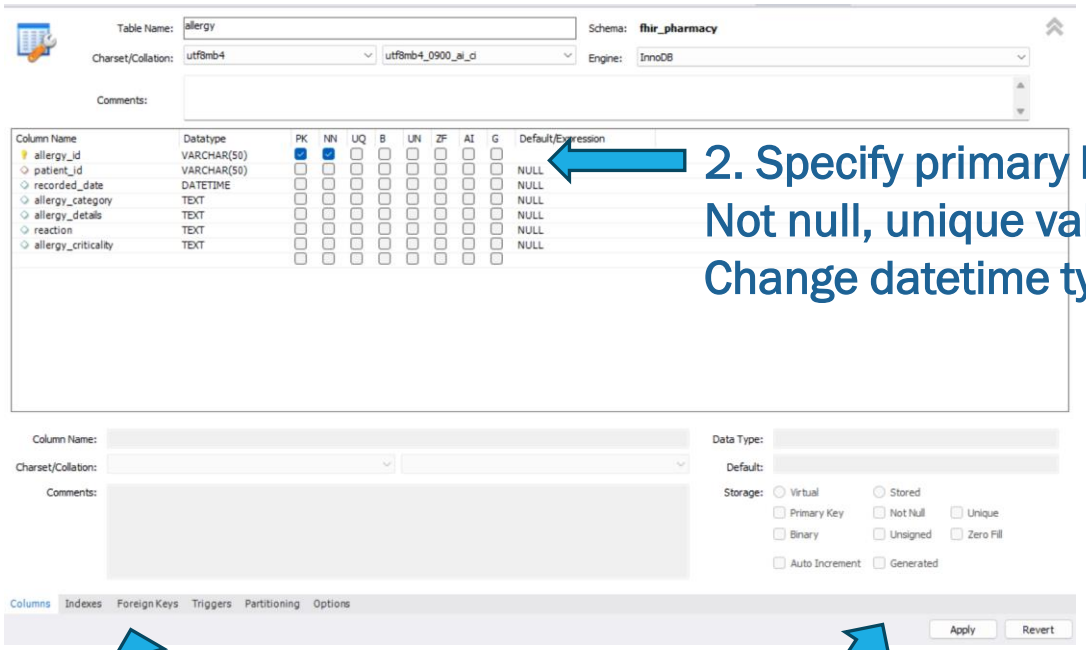
Columns:

allergy_id	varchar(50) PK
patient_id	varchar(50)
recorded_date	datetime
allergy_category	text
allergy_details	text
reaction	text
allergy_criticality	text

Adjustments to variable types and designation of foreign keys and indexes on those foreign keys can be made using table properties functionality.



1. Click on wrench…

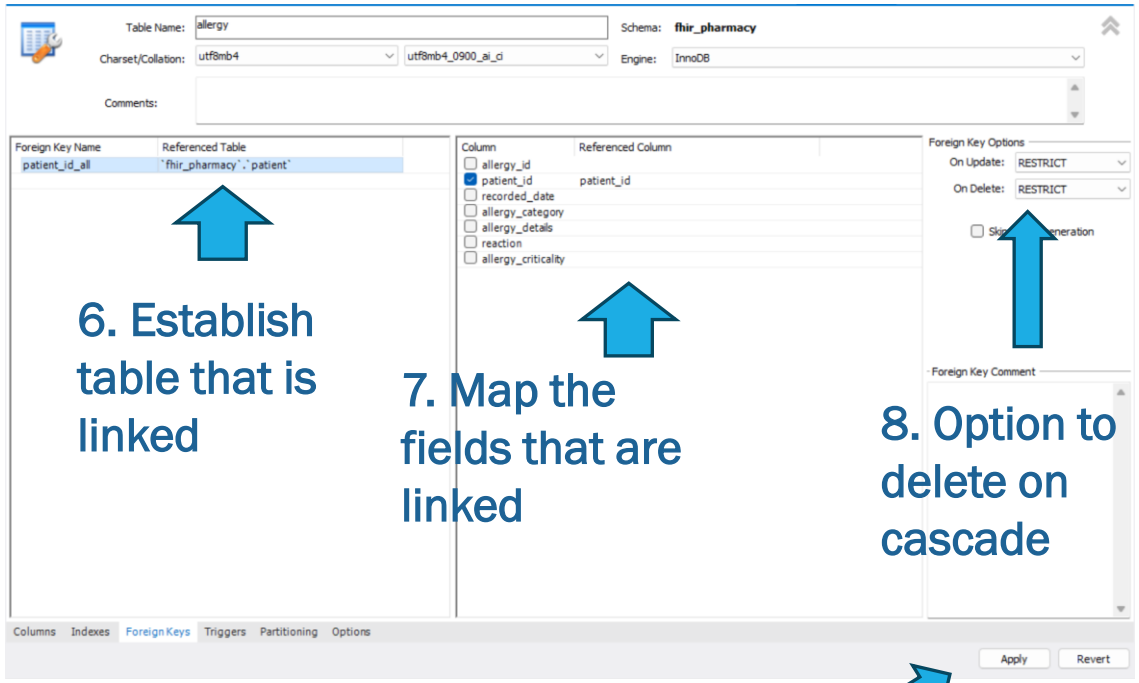


2. Specify primary key
Not null, unique values.
Change datetime types.

4. Designate foreign keys

3. Applies changes. (generates sql code)

5. Foreign key relationship designation..



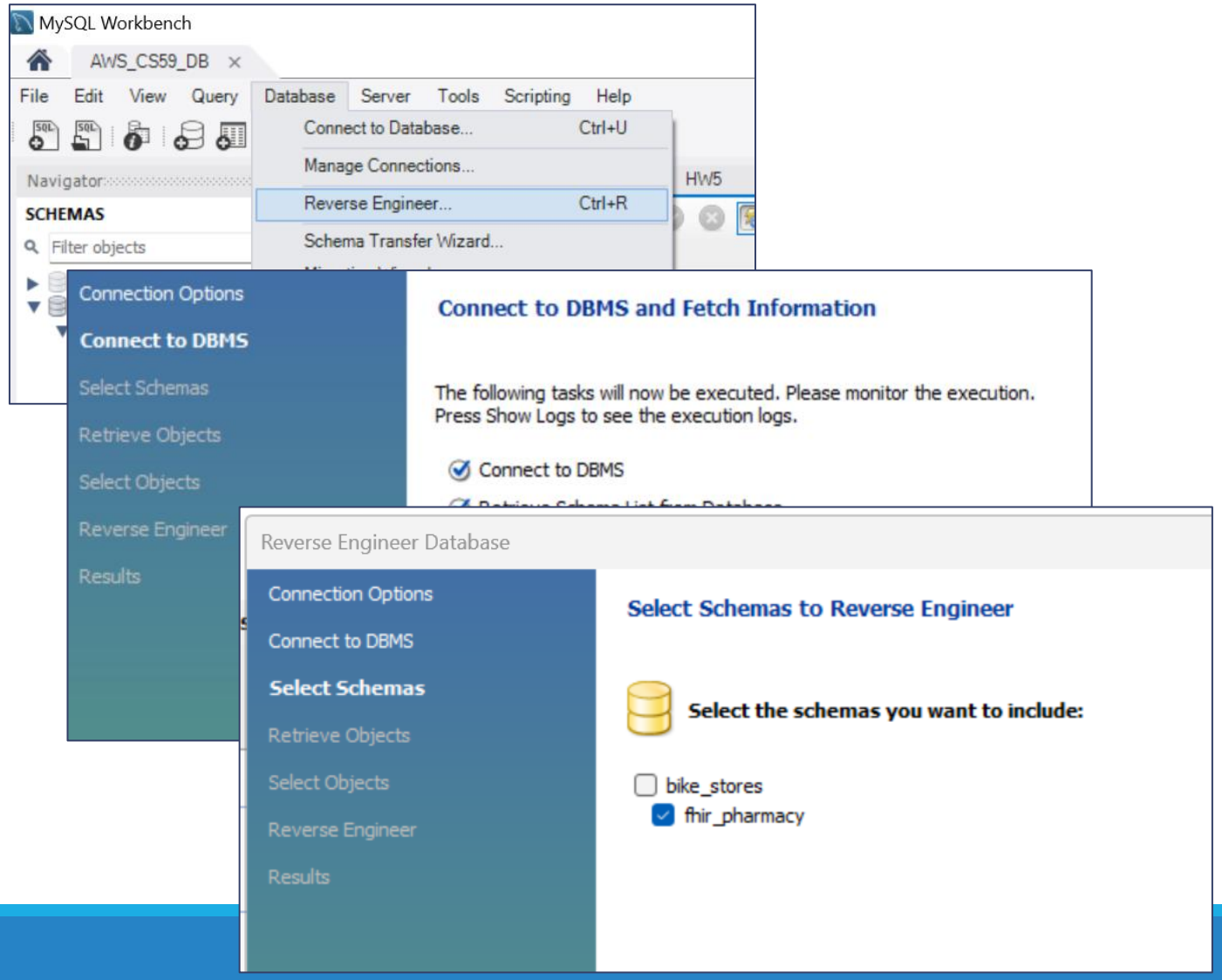
6. Establish table that is linked

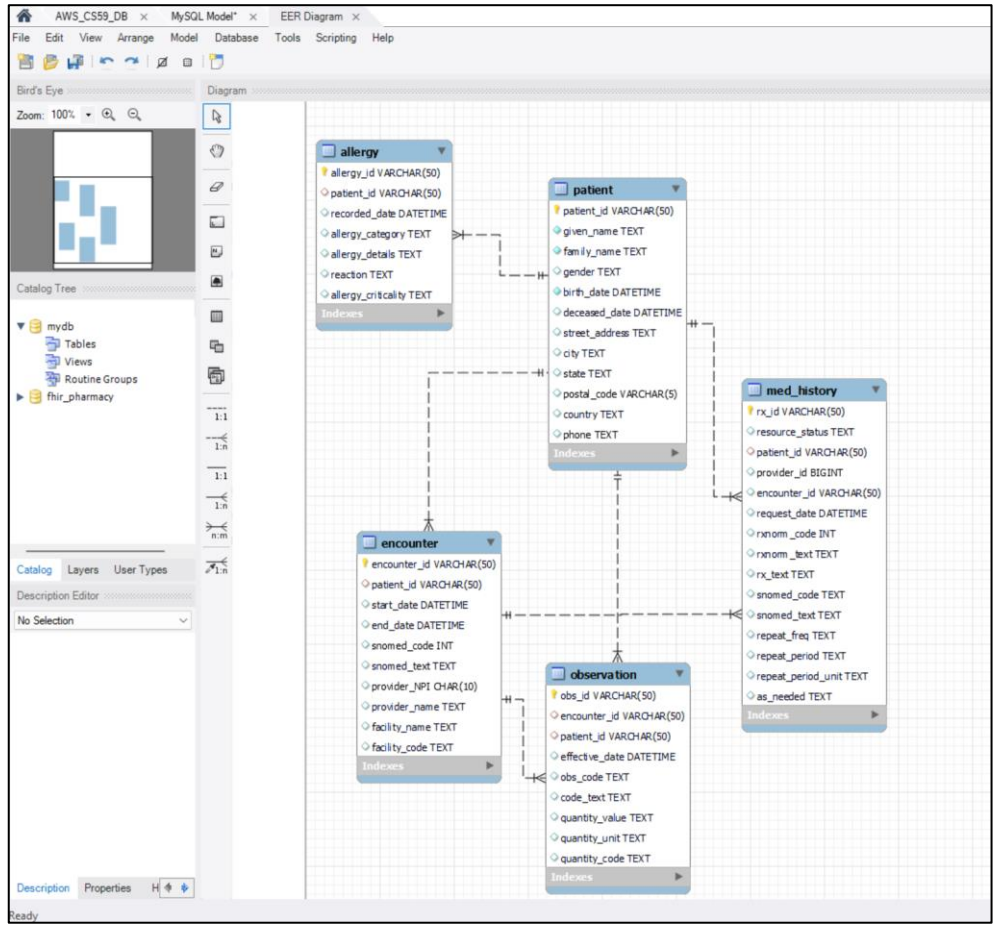
7. Map the fields that are linked

8. Option to delete on cascade

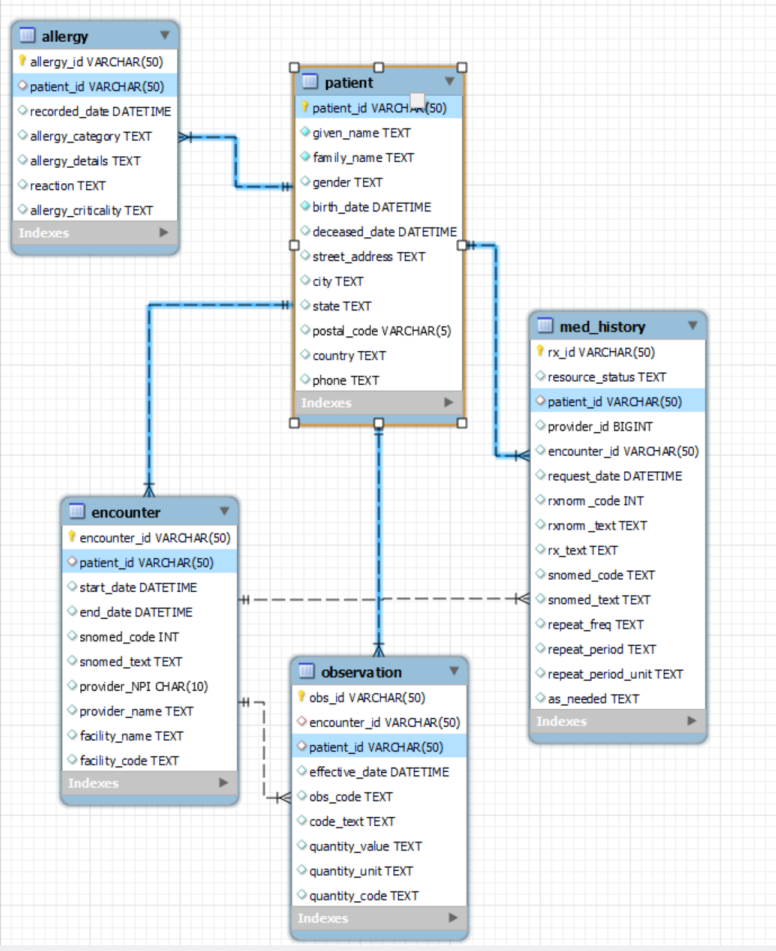
9. Applies changes.

The ‘Reverse Engineer’ functionality in MySQL Workbench was helpful in helping to generate the final entity relationship diagram



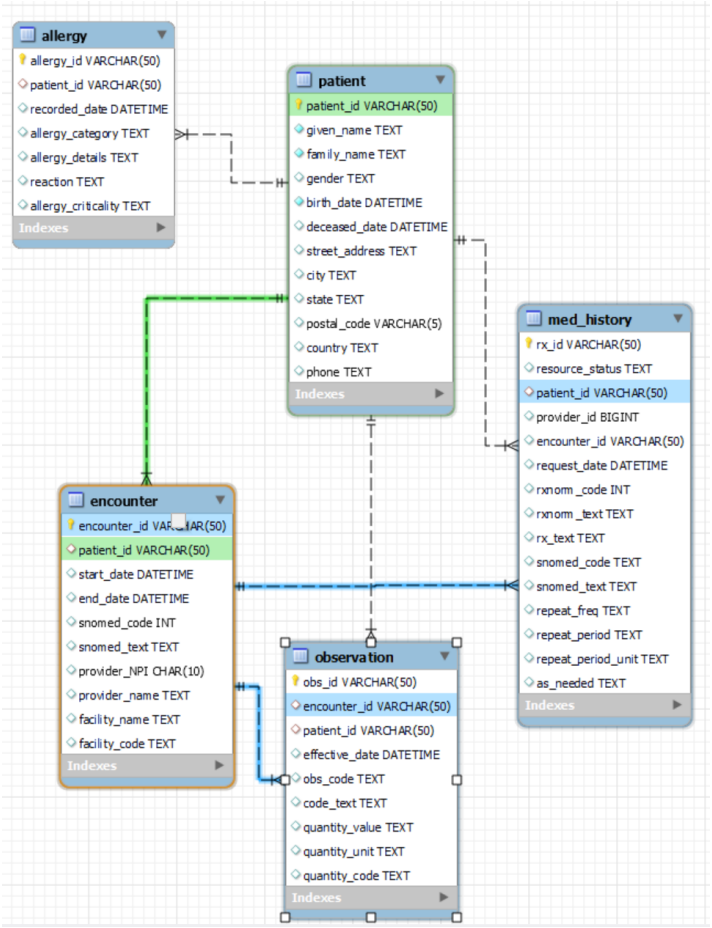


The patient table is the main table, which links to the other four tables. The encounter table is linked to medication and observations derived from that encounter:



Patient ID has a foreign key relationship with all four satellite tables.

A single patient can be linked to multiple records in the satellite tables.



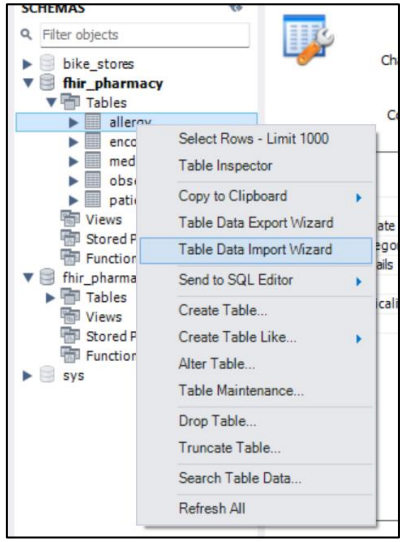
Encounter ID also has a foreign key relationship with observation and med_history tables.

A single encounter can be linked to multiple observation or medication records.

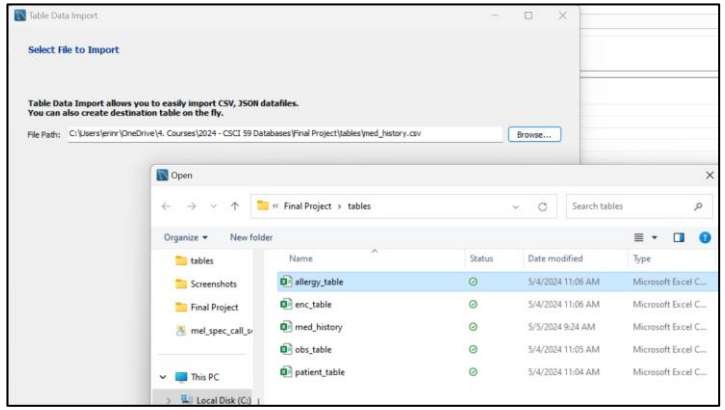
Helpful Hint: Start with tables that have dependencies first.

The Table Import Wizard in MySQL Workbench can also import data. Larger tables took multiple hours to load…

1. Right click on table…

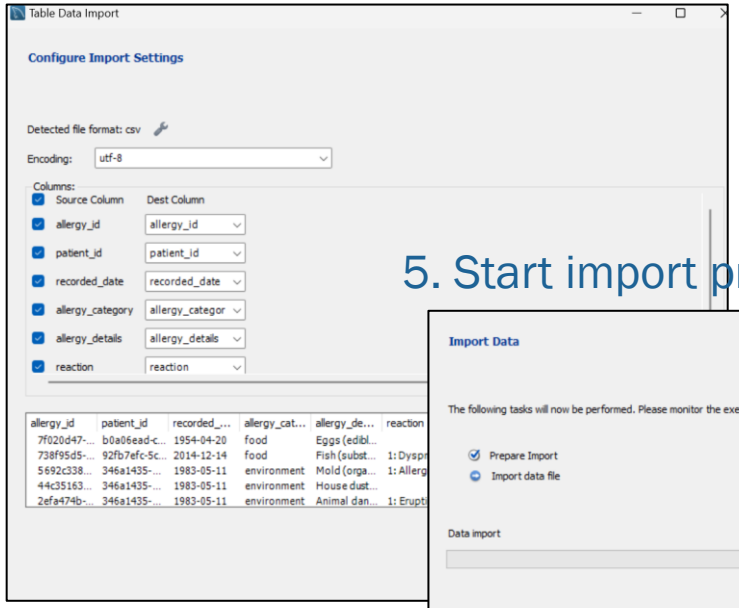


2. Select file to import…

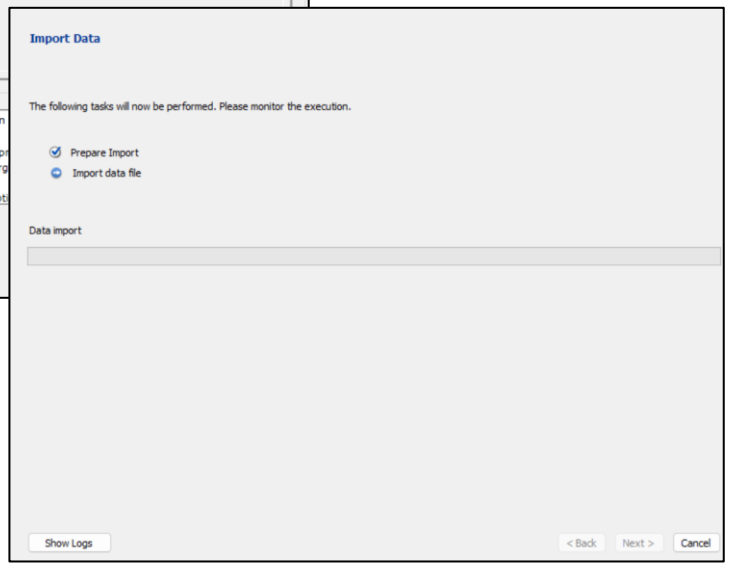


3. Select destination table…

4. Check mapping to existing fields..



5. Start import process…



With the new table structure, pharmacists can easily access data about their patient, which could be useful for prescription verification and patient coaching programs.

Sample Queries
Patient Profile

Active Medication Profile

```
select p.given_name, p.family_name,m.request_date, m.rxnorm_text,
m.repeat_freq, m.repeat_period, m.repeat_period_unit  from patient as p
inner join med_history as m using (patient_id)
where (p.patient_id = '9e84e569-7adc-ff42-ccdb-9fe9c23842a6' and m.resource_status = 'active');
```

given_name	family_name	request_date	rxnorm_text	repeat_freq	repeat_period	repeat_period_unit
Hilll811	Armando772	2021-03-07 00:00:00	24 HR Metformin hydrochloride 500 MG Extende...			
Hilll811	Armando772	2021-03-07 00:00:00	Hydrochlorothiazide 25 MG Oral Tablet	1	1.0	d
Hilll811	Armando772	2021-03-07 00:00:00	Warfarin Sodium 5 MG Oral Tablet			
Hilll811	Armando772	2021-03-07 00:00:00	Digoxin 0.125 MG Oral Tablet			
Hilll811	Armando772	1993-06-13 00:00:00	ferrous sulfate 325 MG Oral Tablet			
Hilll811	Armando772	2021-03-07 00:00:00	Verapamil Hydrochloride 40 MG			
Hilll811	Armando772	2021-03-07 00:00:00	amLODIPine 2.5 MG Oral Tablet	1	1.0	d
Hilll811	Armando772	2021-03-07 00:00:00	insulin human, isophane 70 UNT/ML / Regular In...			

Most Recent Patient Observations:

Sample Queries
Patient Profile

```
#  Observations list from 2021  (Latest Data date is mid 2021)

select p.given_name, p.family_name, o.effective_date ,o.obs_code, o.code_text, o.quantity_value,
o.quantity_unit, o.quantity_code  from patient as p
inner join observation as o using (patient_id)
where (p.patient_id = '9e84e569-7adc-ff42-ccdb-9fe9c23842a6' and o.effective_date > '2021-01-01');
```

given_name	family_name	effective_date	obs_code	code_text	quantity_value	quantity_unit	quantity_code
Hilll811	Armando772	2021-03-07 00:00:00	laboratory	Triglycerides	124.95	mg/dL	mg/dL
Hilll811	Armando772	2021-03-07 00:00:00	laboratory	Microalbumin Creatinine Ratio	19.13	mg/g	mg/g
Hilll811	Armando772	2021-03-07 00:00:00	vital-signs	Blood Pressure			
Hilll811	Armando772	2021-03-07 00:00:00	survey	Generalized anxiety disorder 7 item (GAD-7) tot...	4.0	{score}	{score}
Hilll811	Armando772	2021-03-07 00:00:00	survey	Patient Health Questionnaire 2 item (PHQ-2) tot...	1.0	{score}	{score}
Hilll811	Armando772	2021-03-07 00:00:00	laboratory	Chloride	101.75	mmol/L	mmol/L
Hilll811	Armando772	2021-03-07 00:00:00	laboratory	High Density Lipoprotein Cholesterol	65.37	mg/dL	mg/dL
Hilll811	Armando772	2021-03-07 00:00:00	laboratory	Calcium	9.17	mg/dL	mg/dL
Hilll811	Armando772	2021-03-07 00:00:00	laboratory	Total Cholesterol	186.13	mg/dL	mg/dL
Hilll811	Armando772	2021-03-07 00:00:00	vital-signs	Body Weight	88.7	kg	kg
Hilll811	Armando772	2021-03-07 00:00:00	vital-signs	Body Mass Index	27.68	kg/m2	kg/m2
Hilll811	Armando772	2021-03-07 00:00:00	survey	Protocol for Responding to and Assessing Patie...			
Hilll811	Armando772	2021-03-07 00:00:00	laboratory	Estimated Glomerular Filtration Rate	52.58	mL/min/{1.7...	mL/min/{1.73...
Hilll811	Armando772	2021-03-07 00:00:00	laboratory	Potassium	4.78	mmol/L	mmol/L
Hilll811	Armando772	2021-03-07 00:00:00	survey	Tobacco smoking status NHIS			
Hilll811	Armando772	2021-03-07 00:00:00	laboratory	Glucose	90.16	mg/dL	mg/dL
Hilll811	Armando772	2021-03-07 00:00:00	survey	Fall risk total [Morse Fall Scale]	28.0	{#}	{#}

List of Patient Allergies:

```
# Allergies list (DIfferent Patient, who has Allergies)
select p.given_name, p.family_name, a.recorded_date,
a.allergy_category, a.allergy_details, a.reaction, a.allergy_criticality
from patient as p
inner join allergy as a using (patient_id)
where (p.patient_id = 'be82309d-1a8f-df82-4cd6-5f03e1060e8e');
```

given_name	family_name	recorded_date	allergy_category	allergy_details	reaction	allergy_criticality
Pouros728	Felton646	1973-10-16 00:00:00	environment	Animal dander (substance)	1: Eruption of skin (disorder) 2: Wheal (finding) ...	low
Pouros728	Felton646	1973-10-16 00:00:00	environment	Mold (organism)		low
Pouros728	Felton646	1973-10-16 00:00:00	environment	Grass pollen (substance)		low
Pouros728	Felton646	1973-10-16 00:00:00	environment	House dust mite (organism)		low
Pouros728	Felton646	1973-10-16 00:00:00	environment	Tree pollen (substance)		low

Sample Queries
Patient Profile

List of Patient Recent Provider Encounters:

```
# List of Providers and Relevant Encounter Information for a Given Patient
select m.request_date, m.rxnorm_text, e.provider_NPI, e.provider_name, e.snomed_text,e.facility_name, e.facility_code
from med_history as m
inner join encounter as e using (encounter_id)
where (m.patient_id = '9e84e569-7adc-ff42-ccdb-9fe9c23842a6' and m.resource_status = 'active');
```

request_date	rxnorm_text	provider_NPI	provider_name	snomed_text	facility_name	facility
2021-03-07 00:00:00	24 HR Metformin hydrochloride 500 MG Extende...	9999999729	Dr. Damaris45 Borer986	Encounter for check up (procedure)	HALLMARK HEALTH SYSTEM	AMB
2021-03-07 00:00:00	Hydrochlorothiazide 25 MG Oral Tablet	9999999729	Dr. Damaris45 Borer986	Encounter for check up (procedure)	HALLMARK HEALTH SYSTEM	AMB
2021-03-07 00:00:00	Warfarin Sodium 5 MG Oral Tablet	9999999729	Dr. Damaris45 Borer986	Encounter for check up (procedure)	HALLMARK HEALTH SYSTEM	AMB
2021-03-07 00:00:00	Digoxin 0.125 MG Oral Tablet	9999999729	Dr. Damaris45 Borer986	Encounter for check up (procedure)	HALLMARK HEALTH SYSTEM	AMB
1993-06-13 00:00:00	ferrous sulfate 325 MG Oral Tablet	9999999729	Dr. Damaris45 Borer986	Encounter for problem	HALLMARK HEALTH SYSTEM	AMB
2021-03-07 00:00:00	Verapamil Hydrochloride 40 MG	9999999729	Dr. Damaris45 Borer986	Encounter for check up (procedure)	HALLMARK HEALTH SYSTEM	AMB
2021-03-07 00:00:00	amLODIPine 2.5 MG Oral Tablet	9999999729	Dr. Damaris45 Borer986	Encounter for check up (procedure)	HALLMARK HEALTH SYSTEM	AMB
2021-03-07 00:00:00	insulin human, isophane 70 UNT/ML / Regular In...	9999999729	Dr. Damaris45 Borer986	Encounter for check up (procedure)	HALLMARK HEALTH SYSTEM	AMB

List of Patients with Diabetic A1-c Levels:

```
#List of diabetes patients and recent (after 2020) A1c levels above diabetes threshold of 6.5
select p.given_name, p.family_name, o.effective_date ,o.obs_code, o.code_text, o.quantity_value,
o.quantity_unit, o.quantity_code  from patient as p
inner join observation as o using (patient_id)
where (o.code_text like '%A1%' and o.effective_date > '2020-01-01' and quantity_value > 6.5)
order by o.effective_date DESC;
```

given_name	family_name	effective_date	obs_code	code_text	quantity_value	quantity_unit	quantity_code
Bosco882	Loretta235	2021-11-15 00:00:00	laboratory	Hemoglobin A1c/Hemoglobin.total in Blood	7.46	%	%
Rosenbaum794	Lorinda137	2021-09-10 00:00:00	laboratory	Hemoglobin A1c/Hemoglobin.total in Blood	7.49	%	%
Osinski784	Beth967	2021-05-27 00:00:00	laboratory	Hemoglobin A1c/Hemoglobin.total in Blood	6.86	%	%
O'Connell601	Matthew562	2021-03-06 00:00:00	laboratory	Hemoglobin A1c/Hemoglobin.total in Blood	6.95	%	%
Bosco882	Loretta235	2020-11-09 00:00:00	laboratory	Hemoglobin A1c/Hemoglobin.total in Blood	7.46	%	%
Rosenbaum794	Lorinda137	2020-09-04 00:00:00	laboratory	Hemoglobin A1c/Hemoglobin.total in Blood	7.35	%	%
Osinski784	Beth967	2020-05-21 00:00:00	laboratory	Hemoglobin A1c/Hemoglobin.total in Blood	6.86	%	%
O'Connell601	Matthew562	2020-02-29 00:00:00	laboratory	Hemoglobin A1c/Hemoglobin.total in Blood	6.75	%	%
Tillman293	Benny518	2020-02-14 00:00:00	laboratory	Hemoglobin A1c/Hemoglobin.total in Blood	6.6	%	%
Tillman293	Benny518	2020-01-10 00:00:00	laboratory	Hemoglobin A1c/Hemoglobin.total in Blood	6.6	%	%

Sample Queries
Patient Profile

List of Patients with Kidney Failure:

```
##  List of patients with recent creatinin clearance values in kidney failure range
##  Dosing needs to be adjusted for medicines

select p.given_name, p.family_name, o.effective_date ,o.obs_code, o.code_text, o.quantity_value,
o.quantity_unit, o.quantity_code  from patient as p
inner join observation as o using (patient_id)
where (o.code_text like '%creat%' and o.effective_date > '2021-01-01' and quantity_value > 90)
order by p.given_name, o.effective_date DESC;
```

given_name	family_name	effective_date	obs_code	code_text	quantity_value	quantity_unit	quantity_code
Gerhold939	Jospeh459	2021-11-04 00:00:00	laboratory	Microalbumin Creatinine Ratio	252.59	mg/g	mg/g
Gerhold939	Jospeh459	2021-09-02 00:00:00	laboratory	Microalbumin Creatinine Ratio	262.36	mg/g	mg/g
Gerhold939	Jospeh459	2021-07-29 00:00:00	laboratory	Microalbumin Creatinine Ratio	242.98	mg/g	mg/g
Gerhold939	Jospeh459	2021-04-29 00:00:00	laboratory	Microalbumin Creatinine Ratio	261.24	mg/g	mg/g
Gerhold939	Jospeh459	2021-04-08 00:00:00	laboratory	Microalbumin Creatinine Ratio	279.56	mg/g	mg/g
Gerhold939	Jospeh459	2021-03-18 00:00:00	laboratory	Microalbumin Creatinine Ratio	227.49	mg/g	mg/g
Gerhold939	Jospeh459	2021-01-28 00:00:00	laboratory	Microalbumin Creatinine Ratio	134.39	mg/g	mg/g
Pouros728	Felton646	2021-10-16 00:00:00	laboratory	Microalbumin Creatinine Ratio	244.82	mg/g	mg/g
Pouros728	Felton646	2021-06-26 00:00:00	laboratory	Microalbumin Creatinine Ratio	168.53	mg/g	mg/g
Weber641	Sung603	2021-09-05 00:00:00	laboratory	Microalbumin Creatinine Ratio	209.15	mg/g	mg/g
Weber641	Sung603	2021-08-01 00:00:00	laboratory	Microalbumin Creatinine Ratio	146.06	mg/g	mg/g
Weber641	Sung603	2021-07-11 00:00:00	laboratory	Microalbumin Creatinine Ratio	200.92	mg/g	mg/g
Weber641	Sung603	2021-04-04 00:00:00	laboratory	Microalbumin Creatinine Ratio	181.85	mg/g	mg/g
Weber641	Sung603	2021-03-14 00:00:00	laboratory	Microalbumin Creatinine Ratio	265.02	mg/g	mg/g
Weber641	Sung603	2021-02-14 00:00:00	laboratory	Microalbumin Creatinine Ratio	239.83	mg/g	mg/g
Weber641	Sung603	2021-01-31 00:00:00	laboratory	Microalbumin Creatinine Ratio	116.21	mg/g	mg/g
Weber641	Sung603	2021-01-03 00:00:00	laboratory	Microalbumin Creatinine Ratio	229.64	mg/g	mg/g

Sample Queries
Patient Profile

With design finalized, SQL code was created to automate the table structure and insert patient data in the future

Create Tables: fhir_create_tables

```sql
CREATE SCHEMA fhir_pharmacy_wip;

-- Select the schema
USE fhir_pharmacy_wip;

-- create tables
CREATE TABLE patient (
    patient_id char(36) PRIMARY KEY NOT NULL UNIQUE,
    given_name varchar(25),
    family_name varchar(25),
    gender varchar(10),
    birth_date datetime,
    deceased_date datetime,
    street_address varchar(64),
    city varchar(25) ,
    state char(2),
    postal_code varchar(5),
    country varchar(25),
    phone varchar(12)
);

CREATE TABLE encounter (
    encounter_id char(36) PRIMARY KEY NOT NULL UNIQUE,
    patient_id char(36),
    start_date datetime,
    end_date datetime,
    snomed_code int,
    snomed_text varchar(512),
    provider_NPI char(10),
    provider_name varchar(50) ,
    facility_name varchar(128),
    facility_code varchar(10),
    FOREIGN KEY (patient_id) REFERENCES patient (patient_id) ON DELETE CASCADE ON UPDATE CASCADE
);
```

```sql
CREATE TABLE med_history (
    rx_id char(36) PRIMARY KEY NOT NULL UNIQUE,
    resource_status varchar(10),
    patient_id char(36),
    provider_id char(10),
    encounter_id char(36),
    request_date datetime,
    rxnorm_code char(10),
    rxnorm_text varchar(256),
    rx_text varchar(256),
    snomed_code text,
    snomed_text varchar(128),
    repeat_freq varchar(12),
    repeat_period varchar(12),
    repeat_period_unit varchar(12),
    as_needed varchar(128),
    FOREIGN KEY (patient_id) REFERENCES patient (patient_id) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (encounter_id) REFERENCES encounter (encounter_id) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

```sql
CREATE TABLE observation (
    obs_id char(36) PRIMARY KEY NOT NULL UNIQUE,
    encounter_id char(36),
    patient_id char(36) NOT NULL,
    effective_date datetime,
    obs_code text,
    code_text text,
    quantity_value text,
    quantity_unit text,
    quantity_code text,
    FOREIGN KEY (patient_id) REFERENCES patient (patient_id) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (encounter_id) REFERENCES encounter (encounter_id) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

```sql
CREATE TABLE allergy (
    allergy_id char(36) PRIMARY KEY NOT NULL UNIQUE,
    patient_id char(36) NOT NULL,
    recorded_date datetime,
    allergy_category varchar(15),
    allergy_details varchar(36),
    reaction varchar(256),
    allergy_criticality varchar(15),
    FOREIGN KEY (patient_id) REFERENCES patient (patient_id) ON DELETE CASCADE ON UPDATE CASCADE
);
```

Helpful Hint: Start with tables that have dependencies first.

Patient → Encounter → Others

With design finalized, SQL code was created to automate the table structure and insert patient data in the future

Load Data: Fhir_patient_load:

```
-- load data for a patient

use fhir_pharmacy_wip;

-- patient data

INSERT INTO patient(patient_id, given_name, family_name, gender, birth_date, deceased_date,
street_address,city,state,postal_code, country, phone)
VALUES("9e84e569-7adc-ff42-ccdb-9fe9c23842a6" ,"Hilll811" ,"Armando772" ,"male" ,
"1945-11-04" ,"1000-01-01" ,"397 Rodriguez Promenade Suite 93" ,"Lynn" ,"MA" ,"01902" ,"US" ,"555-296-4764");

-- encounter data

INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code,
snomed_text, provider_NPI, provider_name,facility_name, facility_code)
VALUES ("f42b5a45-0e30-c707-4618-14ba9e107390" ,"9e84e569-7adc-ff42-ccdb-9fe9c23842a6" ,"1963-12-29" ,"1963-12-29" ,"162673000" ,
"General examination of patient (procedure)" ,"9999954489" ,"Dr. Marian936 Wiza601" ,"PCP112182" ,"AMB");

-- med history data

INSERT INTO med_history(rx_id, resource_status,patient_id, provider_id, encounter_id, request_date, rxnorm_code,
rxnorm_text, rx_text, snomed_code, snomed_text,repeat_freq,repeat_period,repeat_period_unit, as_needed)
VALUES ("0bdf98f7-a0aa-d264-be31-5bbc87ec946d" ,"stopped" ,"9e84e569-7adc-ff42-ccdb-9fe9c23842a6" ,"9999954489" ,
"f42b5a45-0e30-c707-4618-14ba9e107390" ,"1963-12-29" ,"310798" ,"Hydrochlorothiazide 25 MG Oral Tablet" ,
"" ,"" ,"" ,"1" ,"1.0" ,"d" ,"False");

-- observation data (requires related encounter and patient, due to foreign key)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI,
provider_name,facility_name, facility_code) VALUES ("1aebaaae-f3a0-0b9e-936c-9d6202941131" ,
"9e84e569-7adc-ff42-ccdb-9fe9c23842a6" ,"2012-01-22" ,"2012-01-22" ,"185349003" ,"Encounter for check up (procedure)" ,"9999999729" ,"
```

Purpose: I created this SQL query to load a few records as a more digestible way to view the Insert Into statements

With design finalized, SQL code was created to automate the table structure and insert patient data in the future

FHIR_single_patient_load:

```
use fhir_pharmacy_wip;

INSERT INTO patient(patient_id, given_name, family_name, gender, birth_date, deceased_date,street_address,city,state,postal_code, country, phone) VALUES
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
INSERT INTO encounter(encounter_id,patient_id, start_date, end_date, snomed_code, snomed_text, provider_NPI, provider_name,facility_name, facility_code)
```

Purpose: This query loads diabetic patient data.

This is 16K+ patient records.
The .txt data was generated using Python code and outputted to a text file

Key Findings:

Automated MySQL workbench table import useful to get a feel for columns and data types

Helpful to pre-process the data in python to streamline loading process and firm up data formats

Note: Loading process was extremely slow given record counts

SQL queries helpful to automatically generate the environment and load data in the future

Future Enhancements:

Transition data to Snowflake for better performance and for use with pharmacy end users
(Major Pharmacy Systems vendors both use Snowflake)

Python queries can be used as part of the SQL Extract/Load/Transform process and automate data load to database

Additional logic required to understand which data is new versus will need to be updated

Attachments

1. Python Code:
   - JSON_Patient_Data_Parser_submitted.pdf file

2. SQL Code:
   - fhir_create_tables: SQL code to create database and table structure
   - Fhir_patient_load: Loads select data from two patients to feed queries
   - FHIR_single_patient_load: Loads a single patient record (thousands of records)
   - pharmacy_queries: Several queries based on the database

3. Link to youtube presentation:
   - https://www.youtube.com/watch?v=HoeGPjimgDs