U.S. Opiate Prescriptions

An analysis of the U.S. Opioid Epidemic

INTEX Fall 2021 Case

Joshua Keller | YunChen Lee | Donovan Chin | Euigun Jung

Table of Contents

Background Information

Executive Summary

This report provides a full analysis of the opioid crisis in the United States and a web application that allows both drug prescribers (including doctors, physician assistants, nurse practitioners, and anyone else with a license to prescribe) and governmental agencies (U.S. Department of Health and Human Services (HHS) and the Center for Disease Control (CDC)) to observe and analyze data about all drugs prescribed (including opioids) by any prescriber in the States.

The analysis includes SQL analytics, prediction model, recommendation model, and system diagrams based on the prescriptions database which is available to view here. The details of the analysis and data would be also accessed through the web application. This project is designed to analyze and solve the problem of opioid overdose and serve as an integrated application of IS core courses (IS 401 Systems Analysis and Design, IS 402 Database Systems, IS 403 Principle of Business Programming, IS 415 Machine Learning).

Project Overview

The primary focus of the project is to analyze the prescribers, drugs, state, and triple datasets on PostgreSQL and respectively create an optimal prediction and recommendation models for the total number of prescriptions and recommended drugs on Microsoft Azure Machine Learning Studio. All the results would be presented on a web application using Python/Django project.

The following are the main deliverables:

1. Project report: including the description of system diagrams, data analysis, models, and system design
2. Python / Django website: including a landing page, search functionality view (prescribers/drugs), details views (prescribers/drugs), CRUD for prescribers, and model analyses
3. Supporting files: including original datasets and normalized datasets

Business Case

Business Problems/Opportunities

Opioid medications, including oxycodone, hydrocodone, and morphine, are commonly prescribed to treat pain. Opioids gained popularity among doctors in the 1990s for treating patients who had undergone surgery or cancer treatment, but in the last fifteen years, opioids have been prescribed for chronic conditions despite concerns about their safety and effectiveness. Opioid-related deaths have grown in lockstep with the volume of opioids prescribed, coming along with the use of illegal opioids such as heroin. The opioid crisis is devastating communities across America. The latest statistics indicate that the total yearly overdose deaths due to opioids reached a new high of 70,237 in 2017. In October 2017, President Trump declared the opioid crisis a public health emergency. The crisis has reached such a scale that it has become a drag on the economy and a threat to national security.

Several measures have been taken to combat the crisis. However, there is not any accessible database (in the form of an online web app) that allows government officials and prescribers to simply view the volume of prescriptions that are taking place overall. Without any integrated platform, it would be tough for the government to execute corresponding policies and deter the growth of opioid overdose.

Project Scope

The objective of this project is to design a system that solves the problem and allows drug prescribers and governmental agencies to control and keep track of the prescriptions in the United States.

1. **Design features to be incorporated into a single web application that will be used by government officials and prescribers.**
    a. Create a portal that authorized personnel can use to observe and analyze data about all drugs prescribed (including opioids) by any prescriber in the United States
2. **Design a normalized database**
    a. Using this system, data will be stored cleanly and efficiently. Authorized personnel will be able to retrieve information and create reports that will lead to easier planning and finding faster solutions
3. **Design a prediction model**
    a. Create a regression model to predict the total number of prescriptions a prescriber will prescribe using the normalized database
4. **Design a recommender model**
    a. Create a recommender that recommends drugs for a prescriber to prescribe using the normalized database

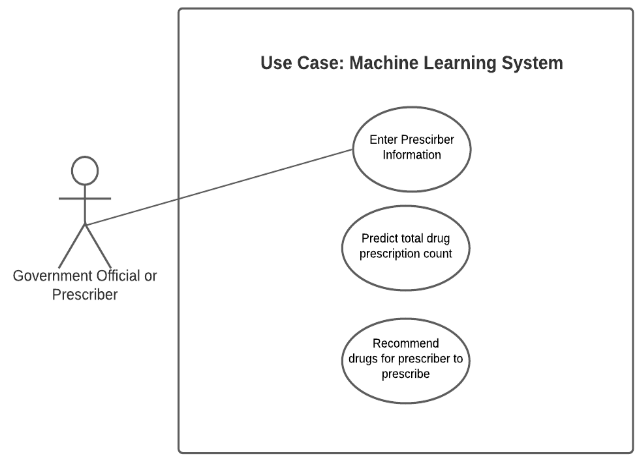
New System Proposal

System Requirements

The system will run on an updating web application. The website will include features for a government official or prescriber, summarizing a searchable list of prescribers and a searchable list of drugs. When either a prescriber or drug is selected, the user should be able to view a “details” page with all available attributes about the object. For example, the government official or prescriber will be able to search for prescribers and look at their information regarding their credentials, state, gender, specialty, and whether they prescribe opioids.
There will also be a CRUD page for prescribers on the website so that users can create, update and delete prescribers from the database as well as update the number of drugs they have prescribed. Yet there is no need to create a CRUD page for drugs. It does not need to track when they were prescribed or updated. The data is a running total.

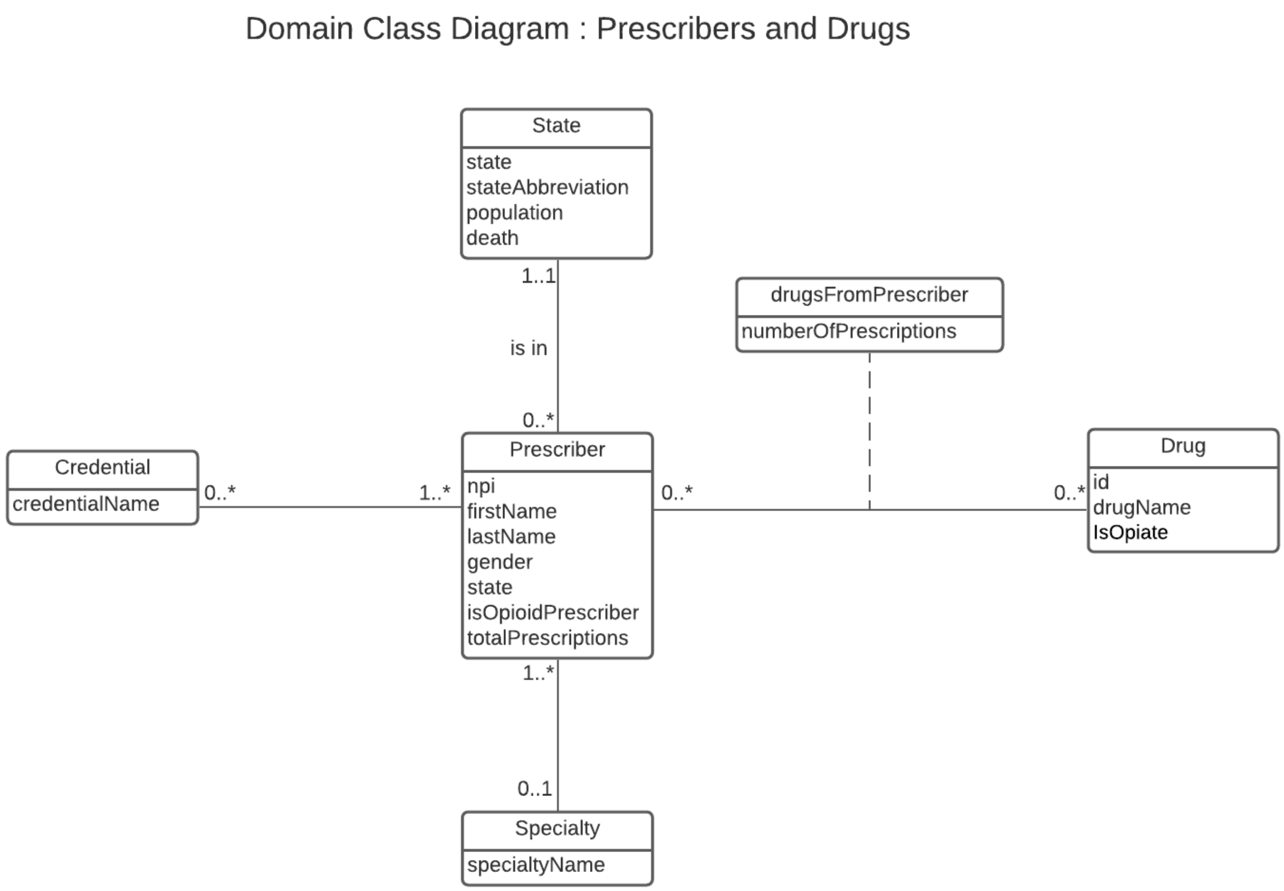
System Diagrams

Use Case Diagrams Proposal



The website system allows users to search and view prescribers and drugs and CRUD prescribers. The Machine Learning System allows users to enter prescriber information and view the predicted total drug prescriptions and recommended drugs for a prescriber.

Domain Class Diagram Proposal



This diagram includes prescriber, state, drug, credential, specialty classes, and associated classes. They have their attributes which were given in the datasets. Each prescriber is in only one state, at most one specialty, and many credentials and drugs prescribed.

Data Analysis

We normalized the tables and created a new database. The new database is not only more efficient, but it also enables us to gather information across tables that we would not be able to do with the original dataset.
We identified the following results:

1. Who is currently prescribing only opioids? Inner join prescribers to drug table

```
Select DISTINCT *
from pd_full_prescriber
WHERE npi not in
(select DISTINCT prescriberid
from pd_triple
inner join pd_drugs on pd_triple.drugname = pd_drugs.drugname
where isopioid = '0')
-- Returns 919 Rows
```

	npi [PK] integer	fname character varying (11)	lname character varying (11)	gender character varying (1)	state character varying (2)	specialtyid integer	isopioidprescriber character varying (5)	totalprescriptions integer
1	1295046324	Parsons	Shawn	M	AZ	13	TRUE	31
2	1770699167	Banks	London	M	MI	24	TRUE	25
3	1396065405	Eva	Mckee	F	OK	11	TRUE	12
4	1174834378	Davenport	Jax	M	MO	5	TRUE	12
5	1083671184	Tate	Louis	M	OH	29	TRUE	26
6	1245551415	Santiago	Raiden	M	GA	4	TRUE	14
7	1821226432	Dahlia	Krueger	F	WI	4	TRUE	19
8	1508880071	Noble	Peyton	M	TX	4	TRUE	22
9	1558678177	Sarai	Reeves	F	NY	3	TRUE	11
10	1801023478	Juarez	Malik	M	ID	1	TRUE	19
11	1700927902	Gisselle	Walters	F	ID	5	TRUE	16
12	1396824595	Reese	Boyer	F	WA	26	TRUE	13
13	1831402841	Arroyo	Lewis	M	WA	32	TRUE	26
14	1194014449	Oneal	Slade	M	IL	4	TRUE	24
15	1033161682	Hood	Cornelius	M	PA	24	TRUE	23
16	1942425632	Tara	Trujillo	F	OR	1	TRUE	13

- Further analysis could be made from here to establish patterns that could lead to understanding why prescribers prescribe different drugs and their amounts.

2. Who is currently prescribing high levels of opioids (compared to other non-opioid drugs)?

```
-- Considering anything above the average as high
Select DISTINCT *
from pd_full_prescriber
WHERE npi in
(select DISTINCT prescriberid
from pd_triple
inner join pd_drugs on pd_triple.drugname = pd_drugs.drugname
where isopioid = '1'
AND qty > (Select AVG(t1.qty) AS AvgNonOpioidPres
FROM
(Select pd_drugs.drugname, pd_drugs.isopioid, pd_triple.qty
from pd_triple
            Inner join pd_full_prescriber ON pd_triple.prescriberid = pd_full_prescriber.NPI
Inner Join pd_drugs ON pd_triple.drugname = pd_drugs.drugname
Where pd_drugs.isopioid = 0) t1));
-- Returns 4352 Rows
```

	npi [PK] integer	fname character varying (11)	lname character varying (11)	gender character varying (1)	state character varying (2)	specialtyid integer	isopioidprescriber character varying (5)	totalprescriptions integer
1	1003009630	Yoder	Juan	M	NY	4	TRUE	232
2	1003016270	Simmons	Jake	M	CT	5	TRUE	2391
3	1003049925	Yaritza	Macdonald	F	ID	3	TRUE	2039
4	1003090986	Cristal	Gardner	F	NH	18	TRUE	3010
5	1003801085	Krystal	Conley	F	MO	11	TRUE	17922
6	1003801366	Cortez	Jamar	M	NY	24	TRUE	2695
7	1003801788	Lamb	Willie	M	TN	11	TRUE	11202
8	1003804519	Jocelyn	Wyatt	F	IL	5	TRUE	2828
9	1003804642	Dominguez	Konner	M	MI	5	TRUE	7285
10	1003806159	Hill	Hassan	M	OH	5	TRUE	10073
11	1003809542	Johns	Zaid	M	NC	5	TRUE	7706
12	1003809765	Noemi	Hoover	F	CA	11	TRUE	2494
13	1003810540	Knox	Quinn	M	AZ	5	TRUE	2948
14	1003813114	Valentine	Ezekiel	M	NC	11	TRUE	6061
15	1003814518	Lozano	Ernesto	M	GA	11	TRUE	4844
16	1003814716	Kaiser	Joel	M	GA	27	TRUE	260

- This analysis highlights prescribers that are potentially prescribing dangerous amounts of opioids.

3. How many opioid drugs have been prescribed?

```
Select SUM(t1.qty) AS TotalOpioidPrescribed
FROM
(Select pd_drugs.drugname, pd_drugs.isopioid, pd_triple.qty
from pd_triple
                    Inner join pd_full_prescriber ON pd_triple.prescriberid = pd_full_prescriber.NPI
Inner join pd_drugs ON pd_triple.drugname = pd_drugs.drugname
WHERE
pd_drugs.drugname LIKE 'ACETAMINOPHEN.CODEINE' OR
pd_drugs.drugname LIKE 'FENTANYL' OR
pd_drugs.drugname LIKE 'HYDROCODONE.ACETAMINOPHEN' OR
pd_drugs.drugname LIKE 'OXYCODONE.ACETAMINOPHEN' OR
pd_drugs.drugname LIKE 'OXYCONTIN' ) t1;
-- RETURNED 1 Row 1,395,571
```

	totalopioidprescribed bigint
1	1395571

- This result tells us how many prescriptions were made out and gives us an idea about the size and age of the database.

4. What opioid drug has been prescribed the most?

```
Select t2.drugname AS MaxOpioid, t2.TotalOpioidPrescribed AS MaxQuantity
FROM
(Select t1.drugname, SUM(t1.qty) AS TotalOpioidPrescribed
FROM
(Select pd_drugs.drugname, pd_drugs.isopioid, pd_triple.qty
from pd_triple
                    Inner join pd_full_prescriber ON pd_triple.prescriberid = pd_full_prescriber.NPI
Inner join pd_drugs ON pd_triple.drugname = pd_drugs.drugname
WHERE pd_drugs.isopioid = 1 ) t1
Group by t1.drugname) t2
Where t2.TotalOpioidPrescribed IN
(Select Max(t2.TotalOpioidPrescribed)
From

(Select t1.drugname, SUM(t1.qty) AS TotalOpioidPrescribed
FROM
(Select pd_drugs.drugname, pd_drugs.isopioid, pd_triple.qty
from pd_triple
                    Inner join pd_full_prescriber ON pd_triple.prescriberid = pd_full_prescriber.NPI
Inner join pd_drugs ON pd_triple.drugname = pd_drugs.drugname
WHERE pd_drugs.isopioid = 1 ) t1
Group by t1.drugname) t2);
-- RETURNED 1 Row HYDROCODONE.ACETAMINOPHEN, 958082
```

	maxopioid character varying (30)	maxquantity bigint
1	HYDROCODONE.ACETAMINOPHEN	958082

- This result highlights the most prescribed opioid and can help data analysts understand what to focus on and figure out why this drug is so popular.

5. How many times has a specific drug been prescribed?

```
Select t1.drugname, SUM(t1.qty) AS TotalOpioidPrescribed
FROM
(Select pd_drugs.drugname, pd_drugs.isopioid, pd_triple.qty
from pd_triple
        Inner join pd_full_prescriber ON pd_triple.prescriberid = pd_full_prescriber.NPI
Inner join pd_drugs ON pd_triple.drugname = pd_drugs.drugname) t1
Group by t1.drugname
-- RETURNED 250 Rows all drugs and total prescribed
```

	drugname character varying (30)	totalopioidprescribed bigint
1	ABILIFY	78929
2	ACETAMINOPHEN.CODEINE	59260
3	ACYCLOVIR	26342
4	ADVAIR.DISKUS	176025
5	AGGRENOX	17711
6	ALENDRONATE.SODIUM	224071
7	ALLOPURINOL	232643
8	ALPRAZOLAM	403203
9	AMIODARONE.HCL	59871
10	AMITRIPTYLINE.HCL	108679
11	AMLODIPINE.BESYLATE	1107865
12	AMLODIPINE.BESYLATE.BE...	71081
13	AMOX.TR.POTASSIUM.CLA...	69593
14	AMOXICILLIN	150149
15	AMPHETAMINE.SALT.COMBO	24644
16	ATENOLOL	394700

- This result is useful because it shows how each drug ranks up against other drugs and further analysis can be done to understand why a certain drug is popular or uncommon.

6. What state has the most opioid-related deaths?

Select *

From pd_statedata

Order by Deaths Desc

LIMIT 1;

-- RETURNED 1 Row California has the highest opioid related deaths

	state character varying (14)	stateabbrev character varying (2)	population integer	deaths integer
1	California	CA	38332521	4521

- The result from this query can lead to further investigation on a particular state for reasons why it has the death toll recorded.

Azure Machine Learning Prediction Model: Total Number of Prescriptions

We used the pd_full_prescriber dataset to create a regression model to predict the total number of prescriptions a prescriber will prescribe. It helps prescribers to determine the number of prescriptions and keep track of the extreme values of total prescriptions.

Data Preparation

TotalPrescriptions in the dataset are positively skewed (Sample Skewness = 4.909007). The skewed label would lead to an inaccurate predicted value in the model. As a result, applying a mathematical transformation to it is needed. A cube root transformation worked better than a square root transformation. It reduced the skewness to 1.343812.

We used the Tukey Box Plot method to check if Cuberoot(TotalPrescriptions) has outliers:

Min = 2.22398, Max = 38.322282, Q1 = 3.556893, Q3 = 10.510194, SD = 5.482738

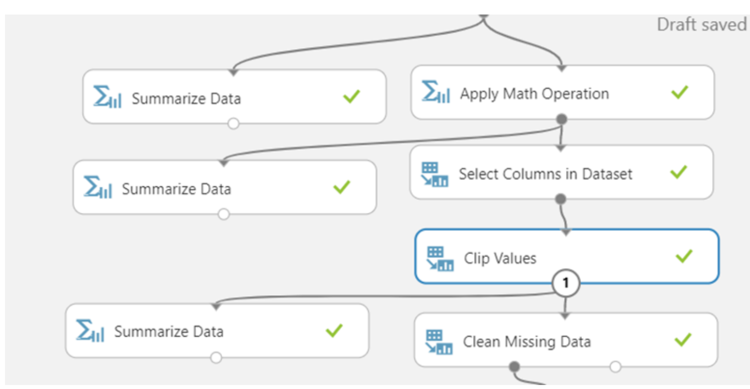
IQR = Q3 – Q1 = 10.510194 - 3.556893 = 6.953301

Upper limit = 10.510194 + (1.5 * 6.953301) = 20.9401455

Lower limit = 3.556893 - (1.5 * 6.953301) = -6.8730585

The maximum is above the upper limit so Cuberoot(TotalPrescriptions) has upper outliers.

We then clipped the upper outliers of Cuberoot(TotalPrescriptions) as missing values and replaced them with probabilistic PCA. Since Microsoft Azure Machine Learning Studio doesn't have the PCA function, we chose to run this process on Microsoft Machine Learning Classic Studio. Other than this process, all models would be run on Azure ML studio.



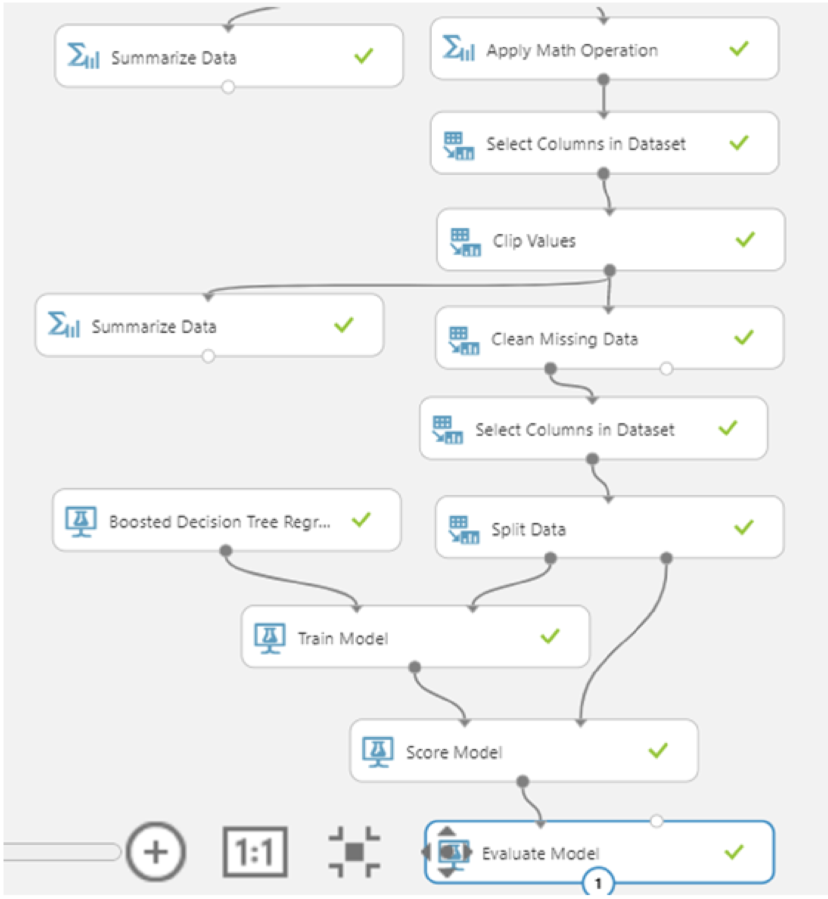
Selecting Algorithms for the Model

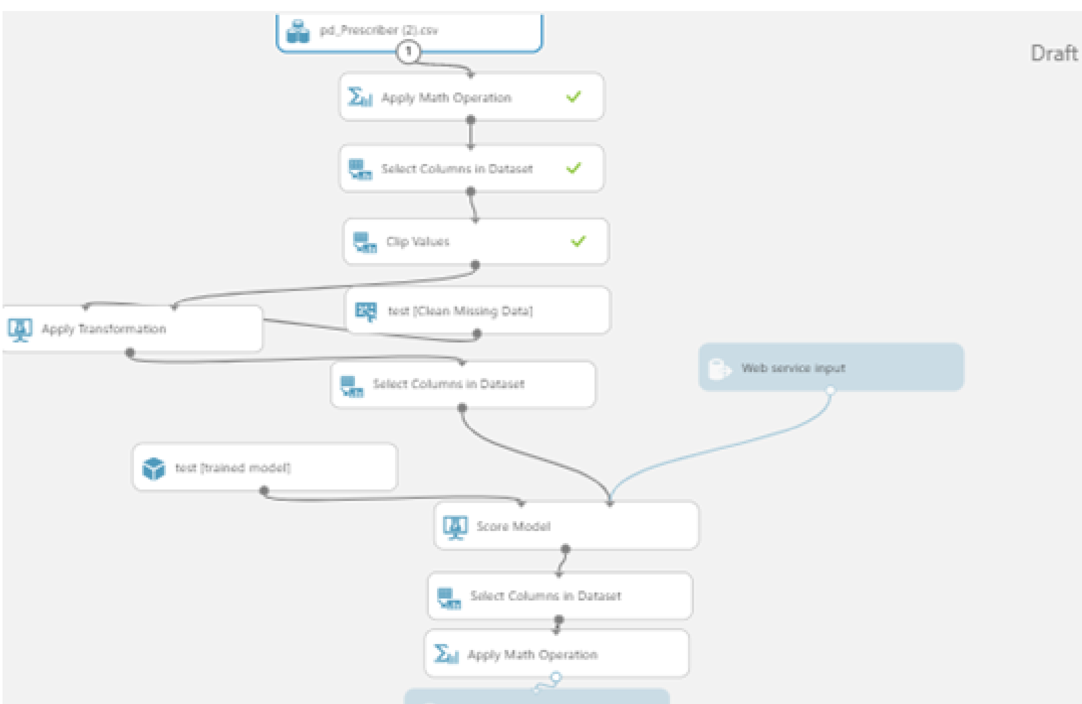
We selected the first 8 columns (NPI, Fname, Lname, Gender, State, SpecialtyID, IsOpioidPrescriber, Cuberoot(TotalPrescriptions)) and split the data into the half with the random seed 12345. We tried Linear Regression, Poisson Regression, Neural Network Regression, Decision Forest Regression, and Boosted Decision Tree Regression and recorded MAE, RMSE, and R Square for each of them. We ended up using Boosted Decision Tree Regression in the final model since it has a higher R square.

Mean Absolute Error	2.867022
Root Mean Squared Error	3.954867
Relative Absolute Error	0.669044
Relative Squared Error	0.535585
Coefficient of Determination	0.464415

Selecting Features for the Model

We included NPI, Fname, Lname, Gender, State, SpecialtyID, and IsOpioidPrescriber in the final model since they are the key columns of the prescriber dataset. The rest of the columns in this table are the possible drugs the prescriber could prescribe. There are 250 additional columns for the 250 drugs. We deleted the drug name columns in the prescribers table and created a drug table since they are not directly relevant to the predicted total prescriptions. As a result, we used NPI, Fname, Lname, Gender, State, SpecialtyID, and IsOpioidPrescriber to predict the total prescriptions.


Summarize Data
Apply Math Operation
Select Columns in Dataset
Clip Values
Summarize Data
Clean Missing Data
Select Columns in Dataset
Boosted Decision Tree Regr...
Split Data
Train Model
Score Model
1:1
Evaluate Model
1


pd_Prescriber (2).csv
1
Draft s
Apply Math Operation
Select Columns in Dataset
Clip Values
test [Clean Missing Data]
Apply Transformation
Web service input
Select Columns in Dataset
test [trained model]
Score Model
Select Columns in Dataset
Apply Math Operation
Web service output

Google Collab link: https://colab.research.google.com/drive/1pDOl3sQAQs5BYp5iAZSINXRiNQZpkSNX

```python
req = urllib.request.Request(url, body, headers)
response = urllib.request.urlopen(req)
result = response.read()
result = json.loads(result)
print("NPI: " + data['Inputs']['input1']['Values'][0][0])
print("Fname: " + data['Inputs']['input1']['Values'][0][1])
print("Lname: " + data['Inputs']['input1']['Values'][0][2])
print("Gender: " + data['Inputs']['input1']['Values'][0][3])
print("State: " + data['Inputs']['input1']['Values'][0][4])
print("SpecialtyID: " + data['Inputs']['input1']['Values'][0][5])
print("IsOpioidPrescriber: " + data['Inputs']['input1']['Values'][0][6])
print("Total Prescription Prediction: " + str(round(float(result['Results']['output1']['value']['Values'][0][0]),0)))
```

```
NPI: 1003009630
Fname: Yoder
Lname: Juan
Gender: M
State: NY
SpecialtyID: 4
IsOpioidPrescriber: TRUE
Total Prescription Prediction: 196.0
```

Azure Machine Learning Recommendation Model: Drugs for a Prescriber

We used the Wide and Deep Recommender, a hybrid (collaborative, content-based) model to accurately recommend the top five drugs for each prescriber. Our goal is to produce a model that has the highest Normalized Discounted Cumulative Gain (NDCG) score between 0 and 1. If the metric is close to 1, it means that the combination of the drug recommendation is perfect. Our team modified the original dataset by converting it to SQL scripts, connecting it to Azure PostgreSQL Flexible Server with pgAdmin4.
* When selecting columns for the Prescribers table, we decided to exclude the “Credentials” column because it has over 700 missings (null) values and multiple credentials for each prescriber, making duplicates for a person. This, in the long run, negatively impacts the whole recommender model.

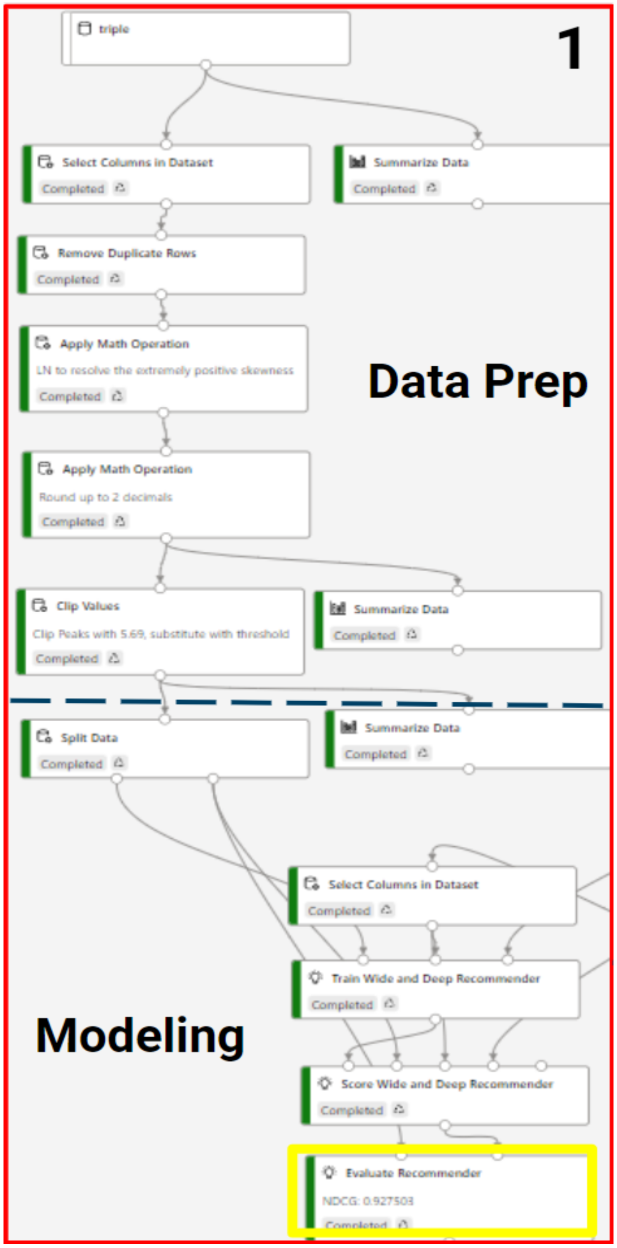
In Azure Machine Learning, we divided into four major steps to build the recommender model:

1. Build a basic Wide and Deep Recommender by using (user-item-rating) triple table, evaluate the model
2. Add User (Prescribers table) features and evaluate the model
3. Add Item (Drugs table) features and evaluate the model
4. Publish the API and deploy the python code in Google Collab

Step 1-3 repeats the following process, and some of the steps can be skipped if there are few column selections or do not apply to the pills:

A. Select columns (Allow duplicates sand preserve column order in selection)
B. Remove Duplicate Rows
C. Data Preparation
   i. Correct Skewness (Apply Math Operation)
   ii. Find outliers (Clip values)
D. Split Data
E. Filter Features (Permutation Feature Importance) with Regression Algorithms
F. Train & Score Wide and Deep Recommender
G. Evaluate Recommender (NDCG score)

1. Basic Recommender Model: User-Item-Rating Triple



A. Select columns (prescriberid, drugname, qty and allow duplicates)

B. Remove Duplicate Rows: select prescriberid, drugname

C. Data Preparation: Select and use appropriate data and cleaning methods to prepare data for modeling

i. Correct Skewness

By summarizing the data, we noticed that the rating(qty) column in the triple table is **extremely positively skewed with a value of 8.739411**.

To improve the skewness of the data, we include the "Apply Math Operation" pill and select Natural Log (LN) to normalize the data distribution. The following is the settings for the pill:

Category: Basic

Basic math function: Ln

Column set: qty

Output mode: Inplace

After running the pill, we see that the skewness for the qty column is in the acceptable range between 0 and 1 (**0.958476**), which is much better than the original data (**8.739411)**.

Rows 3 | Columns 23

	Sample Variance	Sample Standard Deviation	Sample Skewness	Sample Kurtosis	P0.5	P
1	81873190114348270	286134915.93014	-0.026915	-1.193901	1003824731	1
	NaN	NaN	NaN	NaN	NaN	N
656910355	0.806708	0.898169	0.958476	0.485458	2.397895	2

ii. Find Outliers: Empirical Method

Since the triple table is normally distributed, we will use the empirical rule to calculate the outer limits of qty column by using the table below (Numbers are rounded up to two decimals):

Result_dataset | Result_dataset | Result_dataset

Rows 3 | Columns 23

Feature	Count	Unique Value Count	Missing Value Count	Min	Max	Mean	Mean Deviation
prescriberid	625531	25000	0	1003002320	1992998611	1500477307.53109	248124345.040128
drugname	625531	250	0	NaN	NaN	NaN	NaN
qty	625531	428	0	2.4	8.71	3.497199	0.726129

Outlier Upper Limit = mean + (3 * Std. Dev) = 3.5 + (3 * 0.73) = **5.69**

Outlier Lower Limit = mean - (3 * Std. Dev) = 3.5 - (3 * 0.73) = **1.31**

→ The maximum value for the qty column reaches over the outlier upper limit, so we add the "**Clip Value Pill**" to increase the accuracy of the data. This is the setting for the pill:

Set of thresholds: Clippeaks

Upper threshold: Constant

Constant value for upper threshold: 5.69

Substitute value for peaks: Threshold

Columns: qty

Modeling the Recommender

D. Add the “Split Data” pill and configure setting below:

Splitting mode: Split Rows

Fraction of rows in the first output dataset: 0.5

Randomized seed: 12345

Stratified split: False

E. Add “Train Wide and Deep Recommender”: Use the default settings

F. Add “Score Wide and Deep Recommender” (Produce drug recommendations)

Recommender prediction kind: Item Recommendation

Recommended item selection: From Rated Items (for model evaluation)

Minimum size of the recommendation pool for a single user: 1

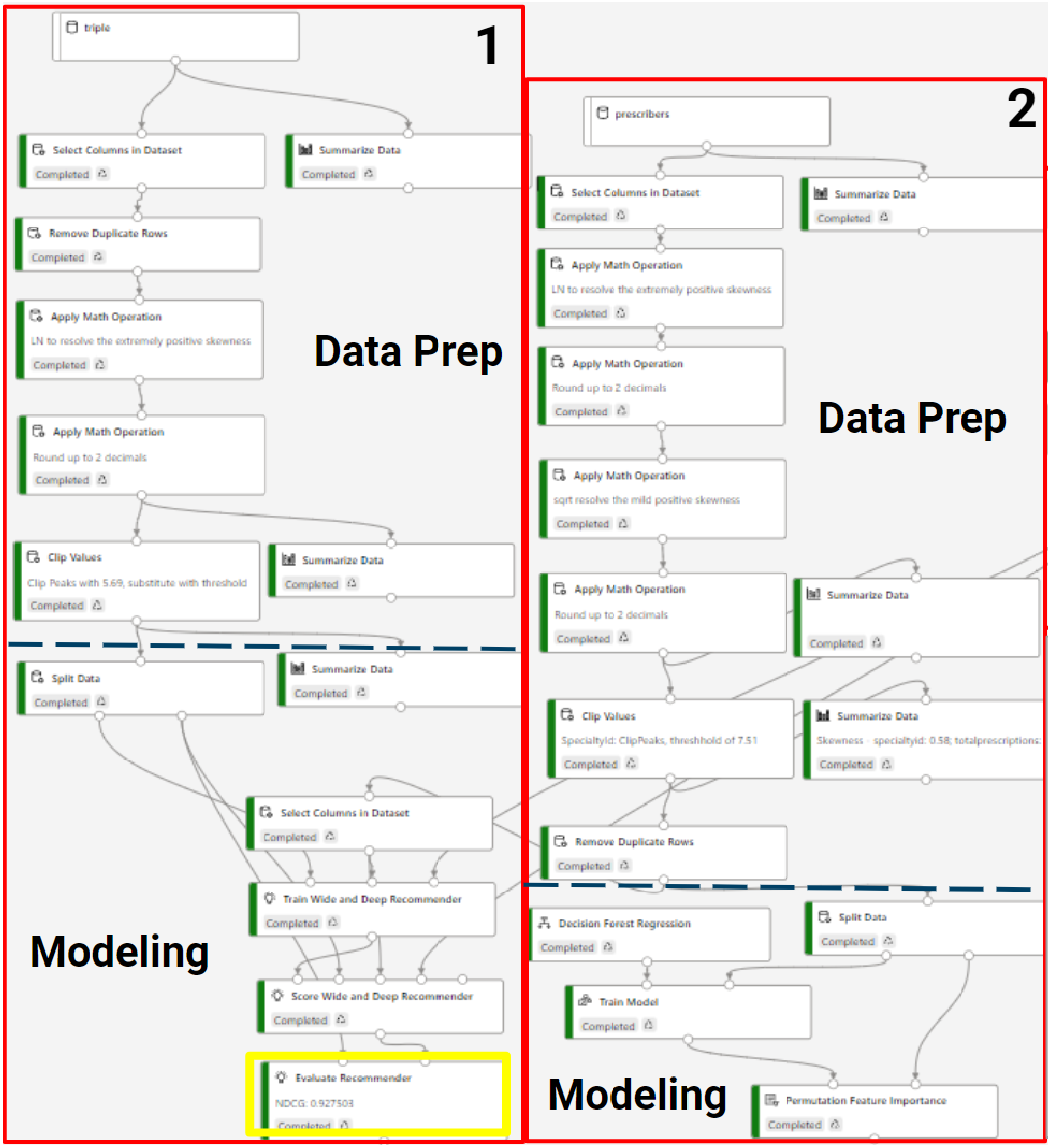
Maximum number of items to recommend to a user: 5

Whether to return the predicted ratings of the items along with the labels: False

G. Add “Evaluation Recommender” (Produce NDCG score)

Add the “Evaluate recommender” pill and run the model. As a result, NDCG score for the model using the triple table is **0.927586**.

2. Adding User Features



We added the user features from the prescribers table to see if this improves the NDCG score for the recommender model. We followed similar steps for the basic recommender (triple).

A. Select Columns: npi,gender,specialtyid,isopioidprescriber,totalprescriptions,state,fname,lname (allow duplicates)
B. Remove Duplicate Rows: select columns prescriberid, drugname
C. Data Preparation

   i. Correct Skewness: Run the “Summarize Data” pill connected to the prescribers table and check the skewness of each column.

	Range	Sample Variance	Sample Standard Deviation	Sample Skewness	Sample Kurtosis	P0.5	P1	P5
320, 1003004771, 75, 1003009630, 70, 1003019019, 72, 1003023193, 94, 1003026055, 07, 1003027038, 27, 1003037979, 32, 1003042805, 24, 1003043480, 28, 1003043993,	989996291	82784677508566940	287723265.497538	-0.007921	-1.199388	1003847309.345	1003995618.12	1053303009
	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	108	213.465488	14.610458	1.952554	4.865902	1	1	1
	1	0.24235	0.492291	-0.355591	-1.873705	0	0	0
	56269	9698098.553009	3114.177027	4.909007	37.71346	11	11	12

Specialtyid: 1.95

TotalPrescriptions: 4.91

Add the “Math Operation Pill” to correct the **skewness** of each data and round up to two decimals. This is the result after running the pills:

Specialtyid, Square root (sqrt): 1.95 → 0.76

TotalPrescriptions, Natural log (ln): 4.91 → 0.26

ii. Find Outliers: Empirical Method

We used the empirical rule for the prescriber table because it is normally distributed after transforming data. Next, we calculated the outlier limits by looking at the table below (Numbers are rounded up to two decimals) and clipped values according to the results:

Feature	Count	Unique Value Count	Missing Value Count	Min	Max	Mean	Mean Deviation
npi	25000	25000	0	1003002320	1992998611	1498162157.92224	249292137.356253
gender	25000	2	0	NaN	NaN	NaN	NaN
specialtyid	25000	109	0	1.0	10.45	3.37363	1.376406
isopioidprescriber	25000	2	0	False	True	0.58752	0.48468
totalprescriptions	25000	667	0	2.4	10.55	5.469617	1.694492
state	25000	57	0	NaN	NaN	NaN	NaN

a) specialtyid

Outlier Upper Limit = mean + (3 * Std. Dev) = 3.37 + (3 * 1.38) = 7.51

Outlier Lower Limit = mean - (3 * Std. Dev) = 3.37 - (3 * 1.38)= -0.77

→ Because the Maximum value passes the upper limit, add the "Clip Values" pill and set "ClipPeaks" to 7.51 and "substitute value for peaks" to threshold.

b) totalprescriptions

Outlier Upper Limit = mean + (3 * Std. Dev) = 5.47 + (3 * 1.70) = 10.57

Outlier Lower Limit = mean - (3 * Std. Dev) = 5.47 - (3 * 1.70) = 0.37

→ We do not clip values because the minimum and the maximum values do not pass the outer limits.

Run the experiment, and check the skewness change for each column after clipping the values:

Specialtyid (Sqrt): 0.76 → 0.58 (Skewness Improved)

TotalPrescriptions (Ln): 0.26 (stays the same because we did not clip values for this column)

D. Split Data: Use the same settings for the basic recommender. Make sure the random seed is “12345”.

E. Filter Features: Permutation Feature Importance

We need to decide which features in the prescribers table help build the accurate recommender model. After splitting the data, add the “Permutation Feature Importance” pill. We will compare three regression algorithms and select the one with the highest coefficient score. In the “Train Model” pill, make sure to include “totalpresctiptions” as a label.

a) Neural Network

Rows 7 Columns 2

Feature	Score
isopioidprescriber	0.216948
specialtyid	0.19248
state	0.026845
gender	0.008694
lname	0.005179
npi	-0.000091
fname	-0.003163

b) Boosted Decision Tree

Rows 7 Columns 2

Feature	Score
specialtyid	0.633395
isopioidprescriber	0.321087
state	0.023317
gender	0.009592
npi	0.003332
lname	0.001307
fname	0.001023

c) Decision Forest

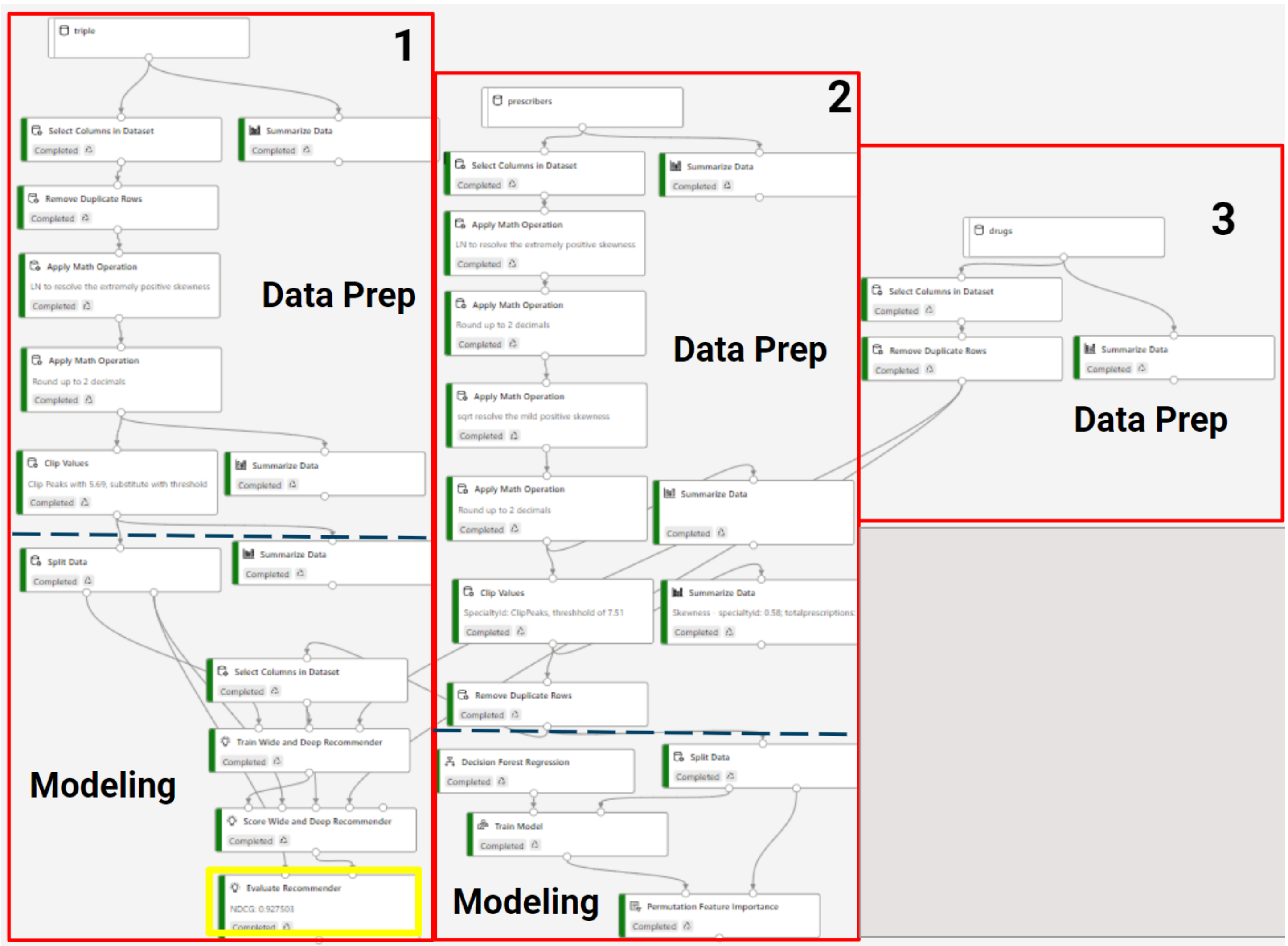
Rows 7 Columns 2

Feature	Score
specialtyid	0.770163
isopioidprescriber	0.43177
state	0.016321
gender	0.011884
lname	-0.000235
npi	-0.00093
fname	-0.003175

We see that npi, lname, and fname rank the lowest among the regressions. However, npi is the userid for the table, so we decided to exclude only fname and lname. Select the following columns for training and scoring the Wide and Deep Recommender: npi, specialtyid, isopioidprescriber, totalprescriptions, gender, and state.

When we trained, scored, and evaluated the recommender with the triple table, the NDCG score showed **0.928127**, which score slightly improved from the first model (0.927586).

3. Adding Item Features (Drugs)



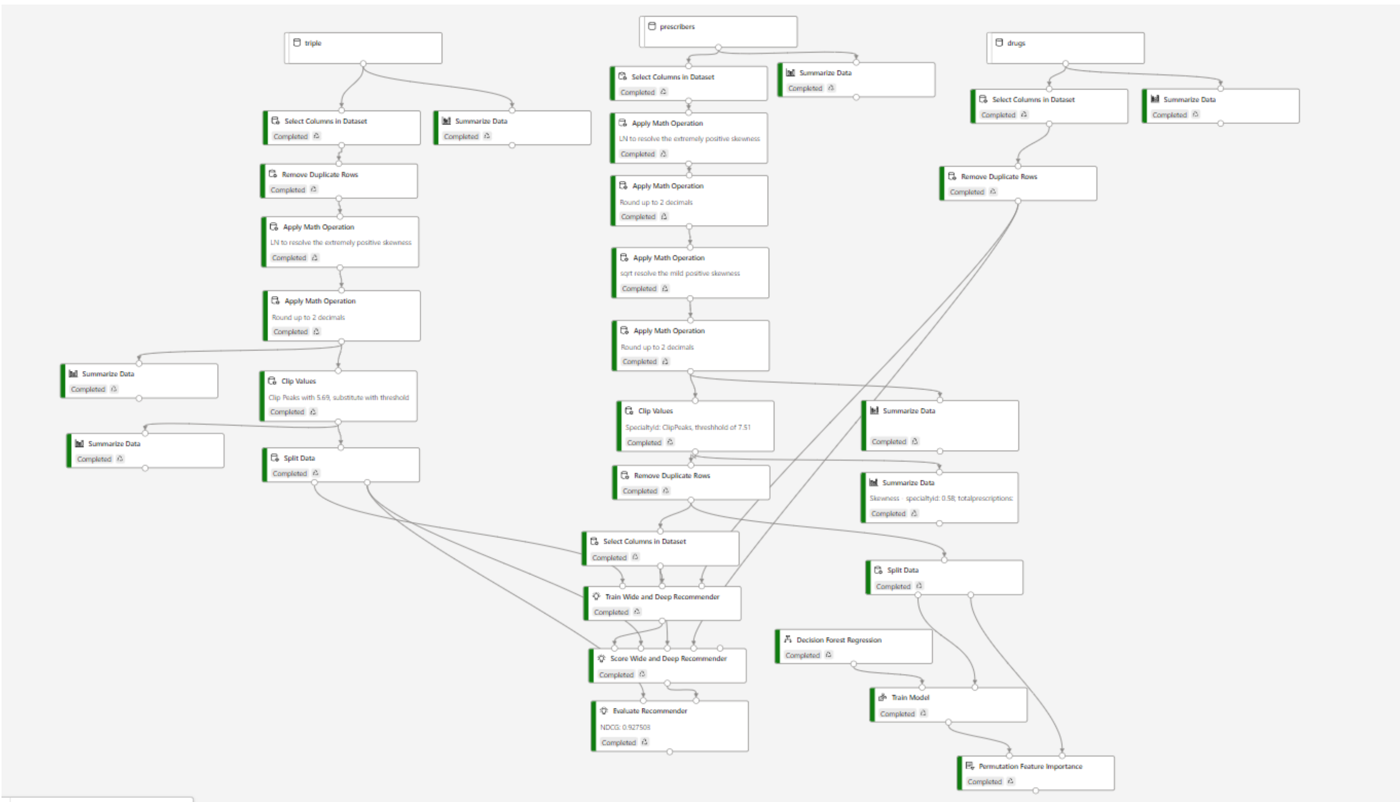
A. Select all columns from the Drugs table: drugid, isopioid, drugname (Allow duplicates and preserve column order in selection)

B. Add the "Remove Duplicate Rows" pill and select drugid and drugname

There is no skewness from this dataset, so we directly connect the "Remove Duplicate Rows" pill to the "Train / Score Wide and Deep Recommender" pill.

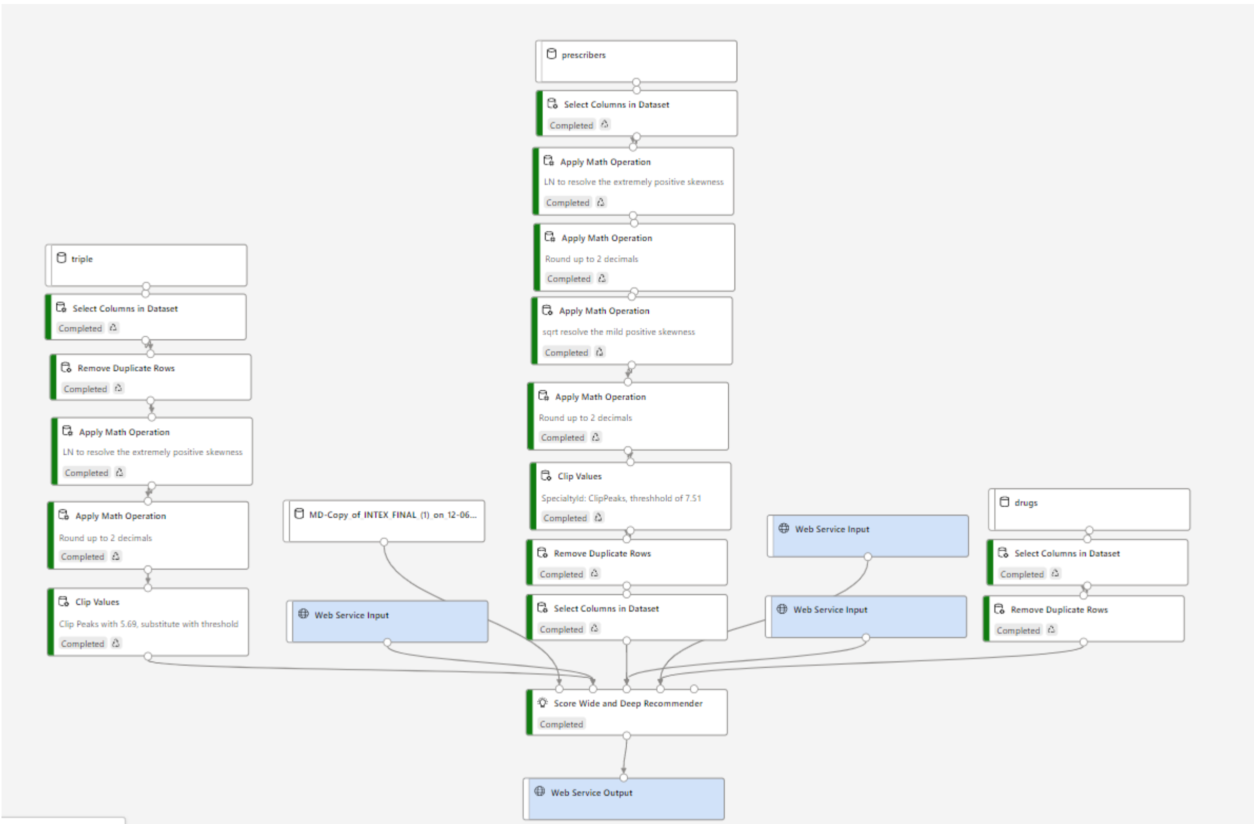
Run the model, and the final NDCG score shows 0.927503, which is 0.000624 less than the previous model. However, this value still gives out a highly reliable NDCG score for the drugs recommender model for prescribers.

A Screenshot of the final model:



4. Deployment

We called a web service from Azure Learning Machine to use this final recommender model.



Rows 25,000 Columns 6

User	Rated Item 1	Rated Item 2	Rated Item 3	Rated Item 4	Rated Item 5
1003002320	HYDROCODONE.ACETAMINOPHEN	AMOXICILLIN	CLINDAMYCIN.HCL		
1003004771	FUROSEMIDE	PROAIR.HFA	PREDNISONE	SPIRIVA	ADVAIR.DISKUS
1003008475	OMEPRAZOLE	LEVOTHYROXINE.SODIUM	FLUTICASONE.PROPIONATE	LISINOPRIL	SIMVASTATIN
1003009630	HYDROCODONE.ACETAMINOPHEN	PREDNISONE	OXYCODONE.ACETAMINOPHEN	AZITHROMYCIN	IBUPROFEN
1003016270	SIMVASTATIN	LEVOTHYROXINE.SODIUM	AMLODIPINE.BESYLATE	LISINOPRIL	ATORVASTATIN.CALCIUM
1003019019	LATANOPROST	LUMIGAN	TRAVATAN.Z	DORZOLAMIDE.TIMOLOL	TIMOLOL.MALEATE
1003022872	AMOXICILLIN				
1003023193	PHENYTOIN.SODIUM.EXTENDED				
1003024894	HYDROCODONE.ACETAMINOPHEN	MELOXICAM	TRAMADOL.HCL	OXYCODONE.ACETAMINOPHEN	IBUPROFEN
1003026055	LEVOTHYROXINE.SODIUM	SIMVASTATIN	ATORVASTATIN.CALCIUM	METFORMIN.HCL	LOSARTAN.POTASSIUM
1003026907	AMOXICILLIN				
1003027038	FLUTICASONE.PROPIONATE	CIPROFLOXACIN.HCL	AMOX.TR.POTASSIUM.CLAVULANATE	MUPIROCIN	
1003034927	LATANOPROST	LUMIGAN	TRAVATAN.Z	DORZOLAMIDE.TIMOLOL	TIMOLOL.MALEATE
1003037979	HYDROCODONE.ACETAMINOPHEN	AMOXICILLIN	CEPHALEXIN	CHLORHEXIDINE.GLUCONATE	

Next, we used Postman and Google Collab to publish API and deploy the final recommender model. Below are the code and results from deploying the recommender:

Google Collab - Json (Python)

```
import requests
import json

url = "http://20fdf162-e061-44cb-9961-9f44f0aee679.eastus2.azurecontainer.io/score"

payload = json.dumps({
  "Inputs": {
    "WebServiceInput1": [
      {
        "npi": 1003002320,
        "gender": "M",
        "specialtyid": 1,
        "isopioidprescriber": True,
        "totalprescriptions": 4.41,
        "state": "MS"
      },
      {
        "npi": 1003004771,
        "gender": "F",
        "specialtyid": 1.42,
        "isopioidprescriber": False,
        "totalprescriptions": 5.86,
        "state": "CO"
      },
      {
        "npi": 1003008475,
        "gender": "F",
        "specialtyid": 1.74,
        "isopioidprescriber": True,
        "totalprescriptions": 4.94,
        "state": "GA"
      }
    ],
    "WebServiceInput2": [
      {
        "prescriberid": 1992883235,
        "drugname": "IPRATROPIUM.BROMIDE",
        "qty": 3.56
      },
      {
        "prescriberid": 1992883235,
        "drugname": "IRBESARTAN",
        "qty": 3.79
      },
      {
        "prescriberid": 1992883235,
        "drugname": "ISOSORBIDE.MONONITRATE.ER",
        "qty": 4.44
```

```
      }
    ],
    "WebServiceInput0": [
      {
        "drugid": 2,
        "isopioid": False,
        "drugname": "ABILIFY"
      },
      {
        "drugid": 3,
        "isopioid": True,
        "drugname": "ACETAMINOPHEN.CODEINE"
      },
      {
        "drugid": 4,
        "isopioid": False,
        "drugname": "ACYCLOVIR"
      }
    ]
  },
  "GlobalParameters": {}
})
headers = {
  'Content-Type': 'application/json',
  'Authorization': 'Bearer 6lA6dTPnojiqNepPumFdgdomffn9fdoI'
}

response = requests.request("POST", url, headers=headers, data=payload)

print(response.text)
```

{"Results": {"WebServiceOutput0": [{"User": "1992883235", "Rated Item 1": "ISOSORBIDE.MONONITRATE.ER", "Rated Item 2": "IPRATROPIUM.BROMIDE", "Rated Item 3": "IRBESARTAN", "Rated Item 4": null, "Rated Item 5": null}]}}

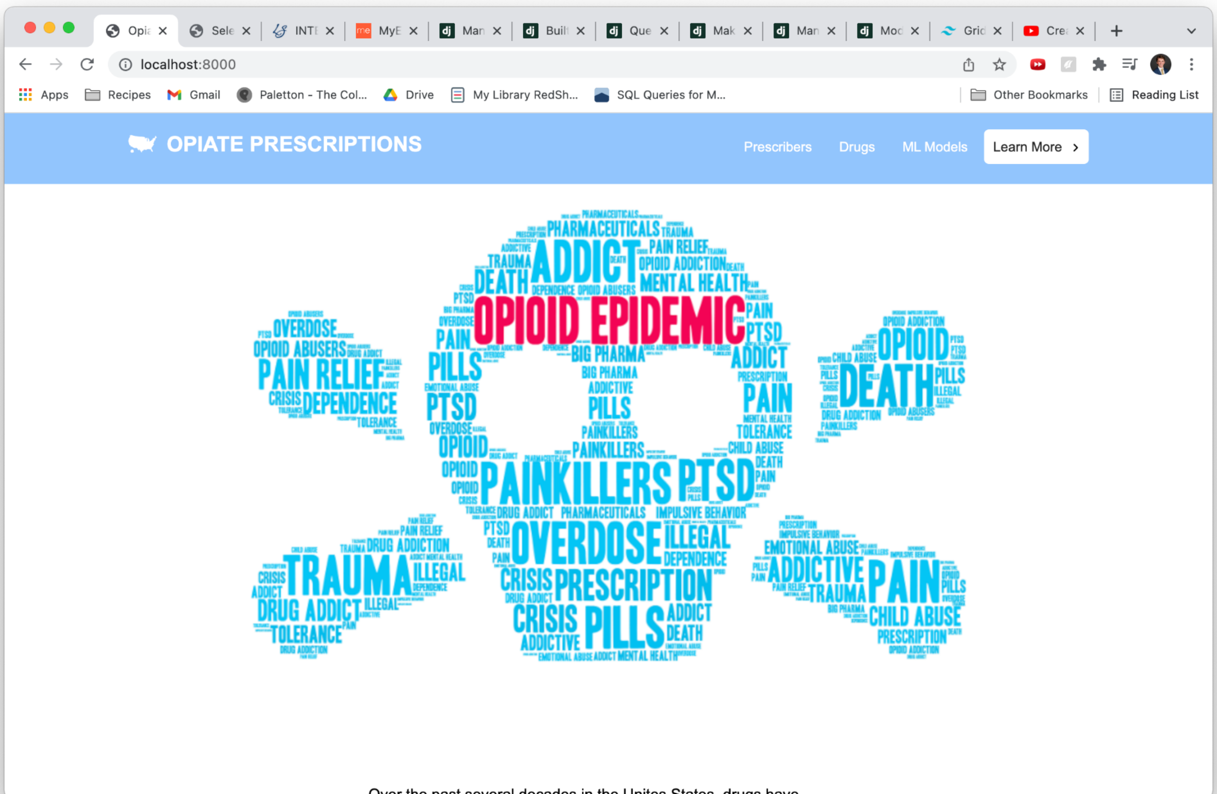
```
jsonData = json.loads(response.text)


for a in jsonData["Results"]["WebServiceOutput0"]:
    print("Recommended Drugs for Prescriber", a["User"])
    print("Recommendation 1:", a["Rated Item 1"])
    print("Recommendation 2:", a["Rated Item 2"])
    print("Recommendation 3:", a["Rated Item 3"])
    print("Recommendation 4:", a["Rated Item 4"])
    print("Recommendation 5:", a["Rated Item 5"])
    print("")
```

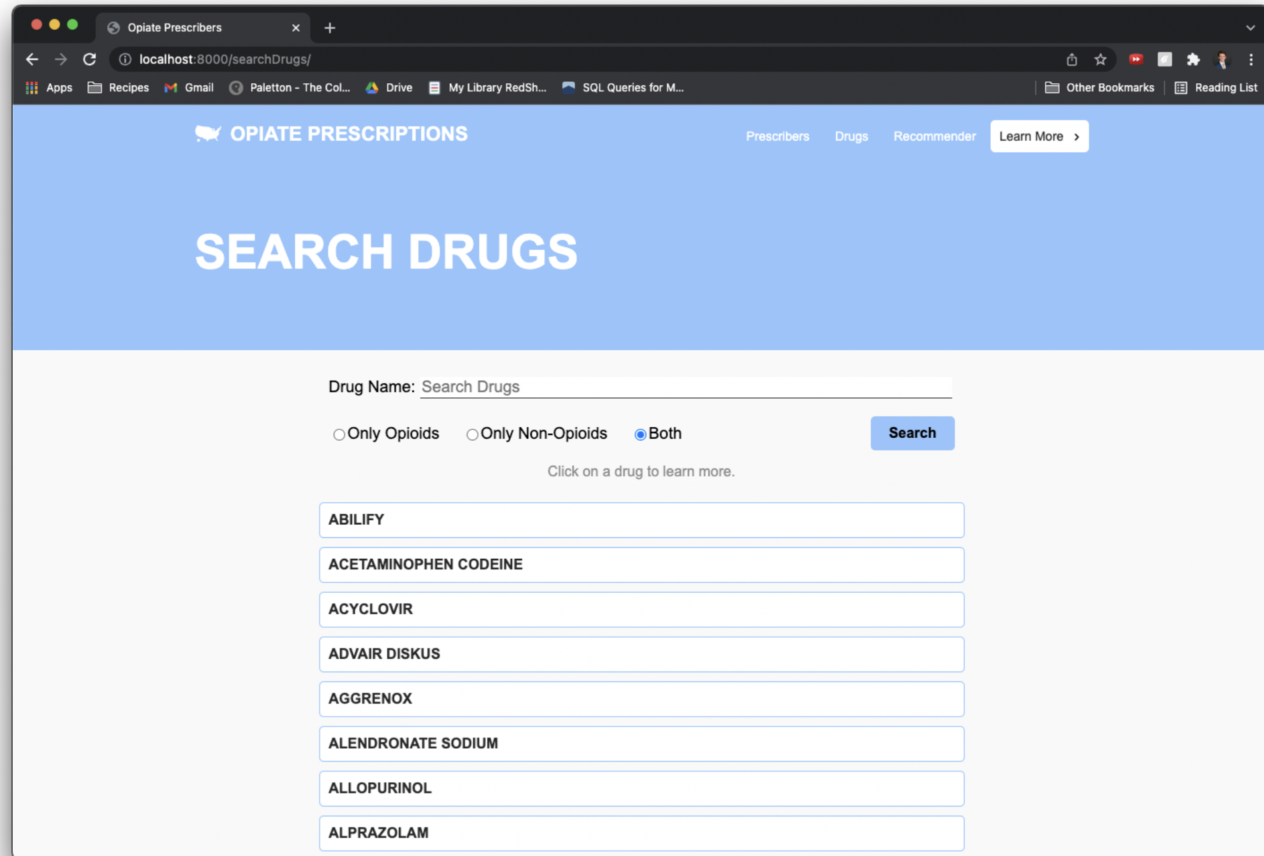
```
Recommended Drugs for Prescriber 1992883235
Recommendation 1: ISOSORBIDE.MONONITRATE.ER
Recommendation 2: IPRATROPIUM.BROMIDE
Recommendation 3: IRBESARTAN
Recommendation 4: None
Recommendation 5: None
```

System Design / Interface

Landing Page

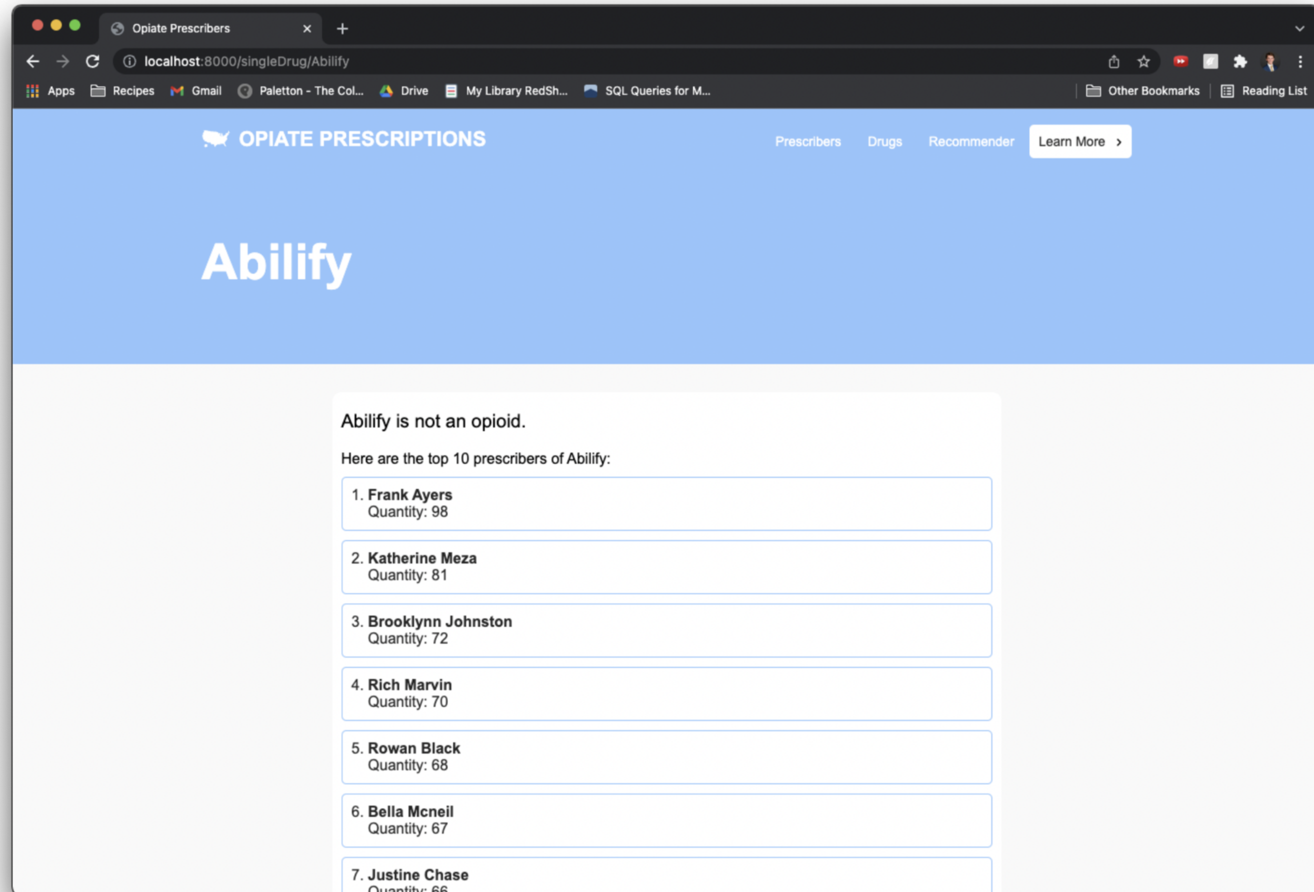


Drug Search



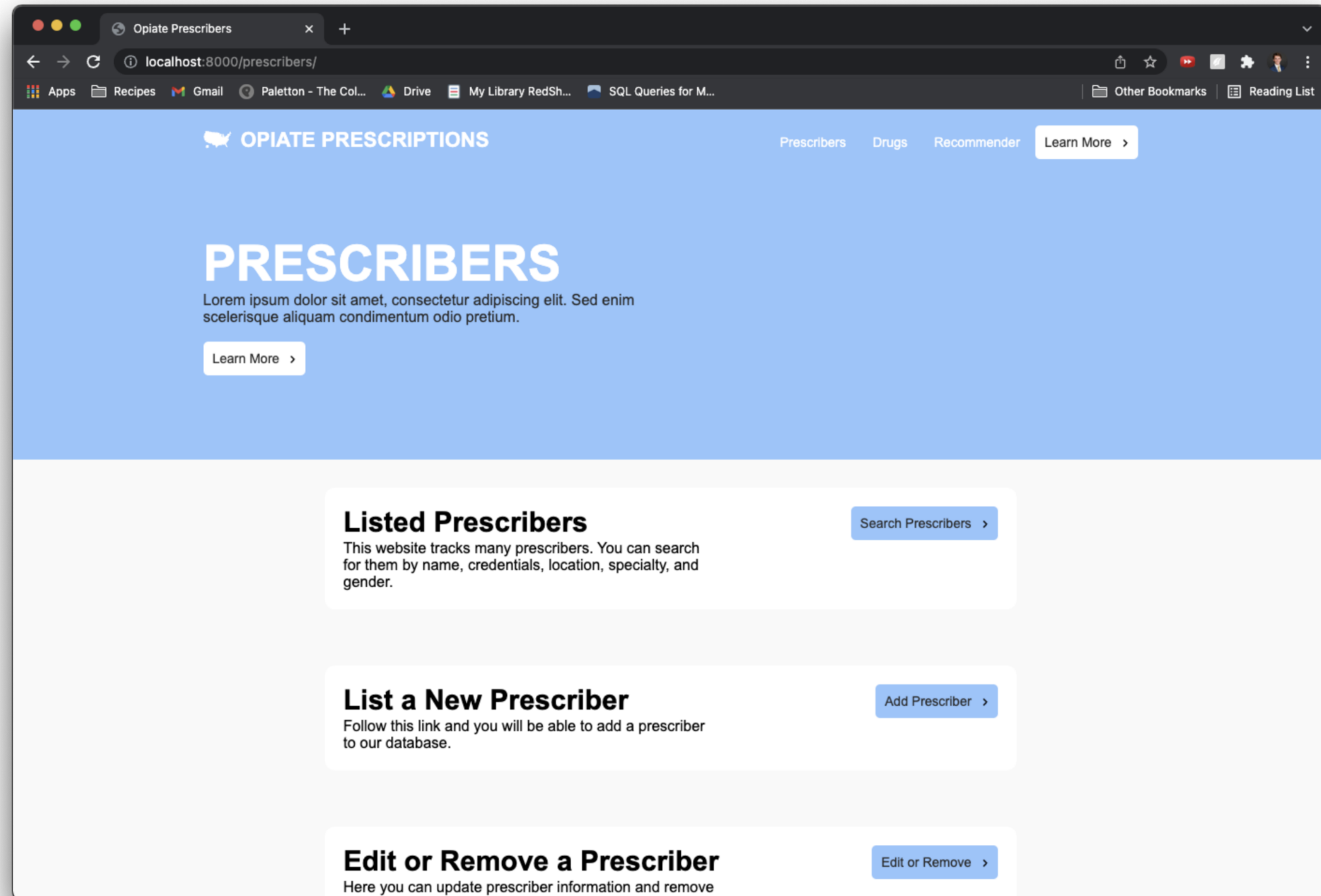
- This page allows you to search for drugs by name or by whether or not it is an opioid.

Drug Page



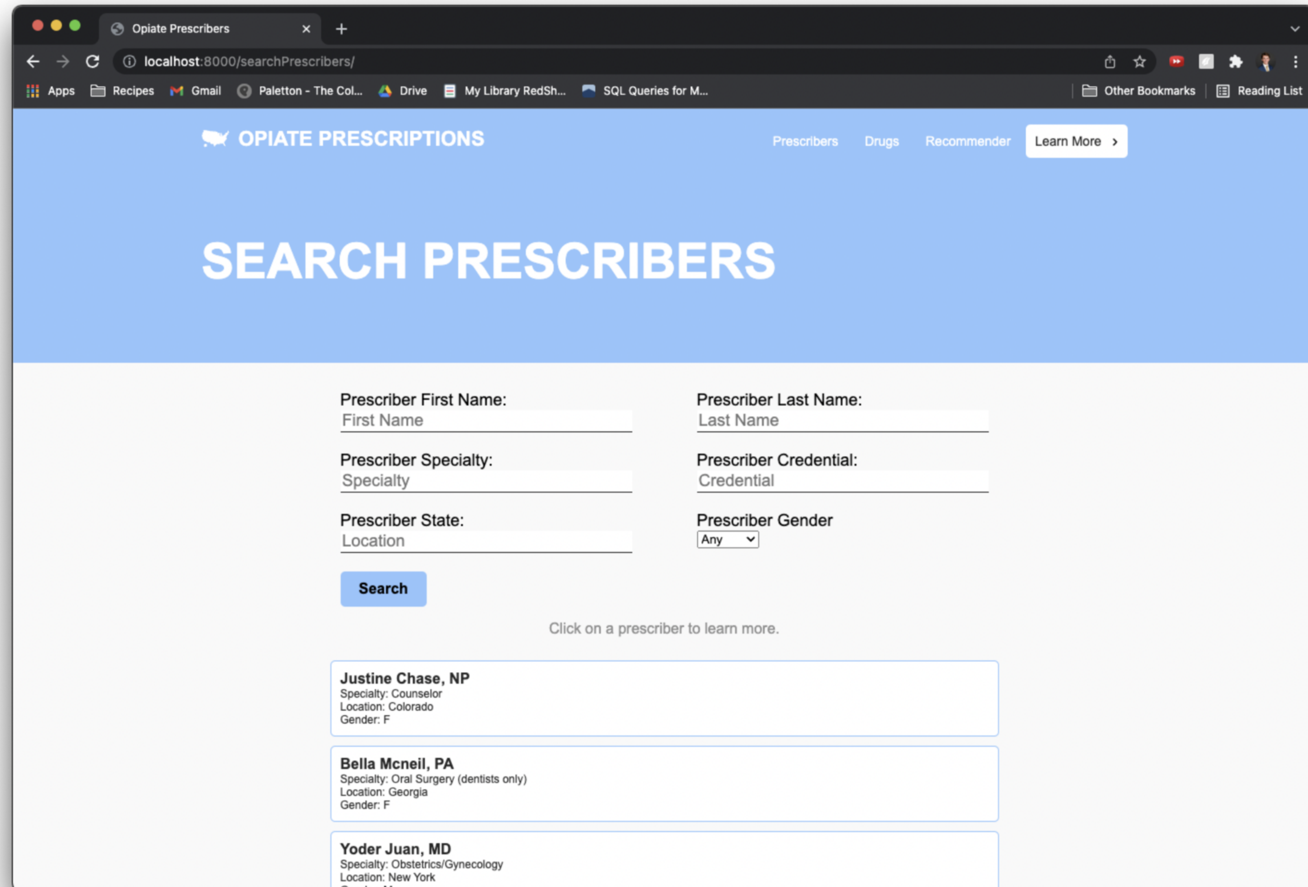
- Every drug has its page. This page displays the name of the drug and details of whether the drug is an opioid or not with the top 10 prescribers of that drug with its given quantity.

Prescribers CRUD Page



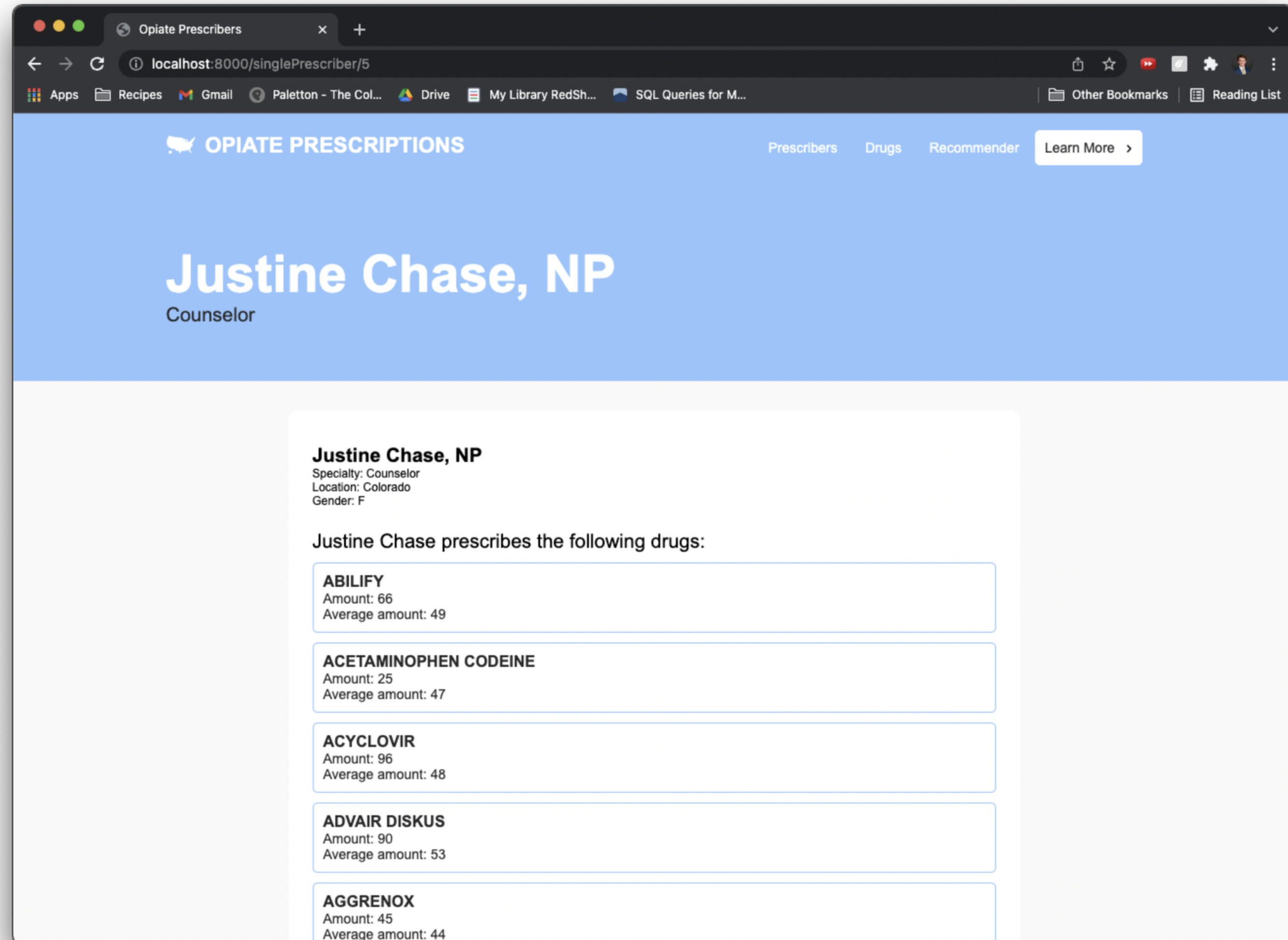
- This page has the functionality to search for prescribers, add prescribers, and edit current prescribers.

Prescriber Search



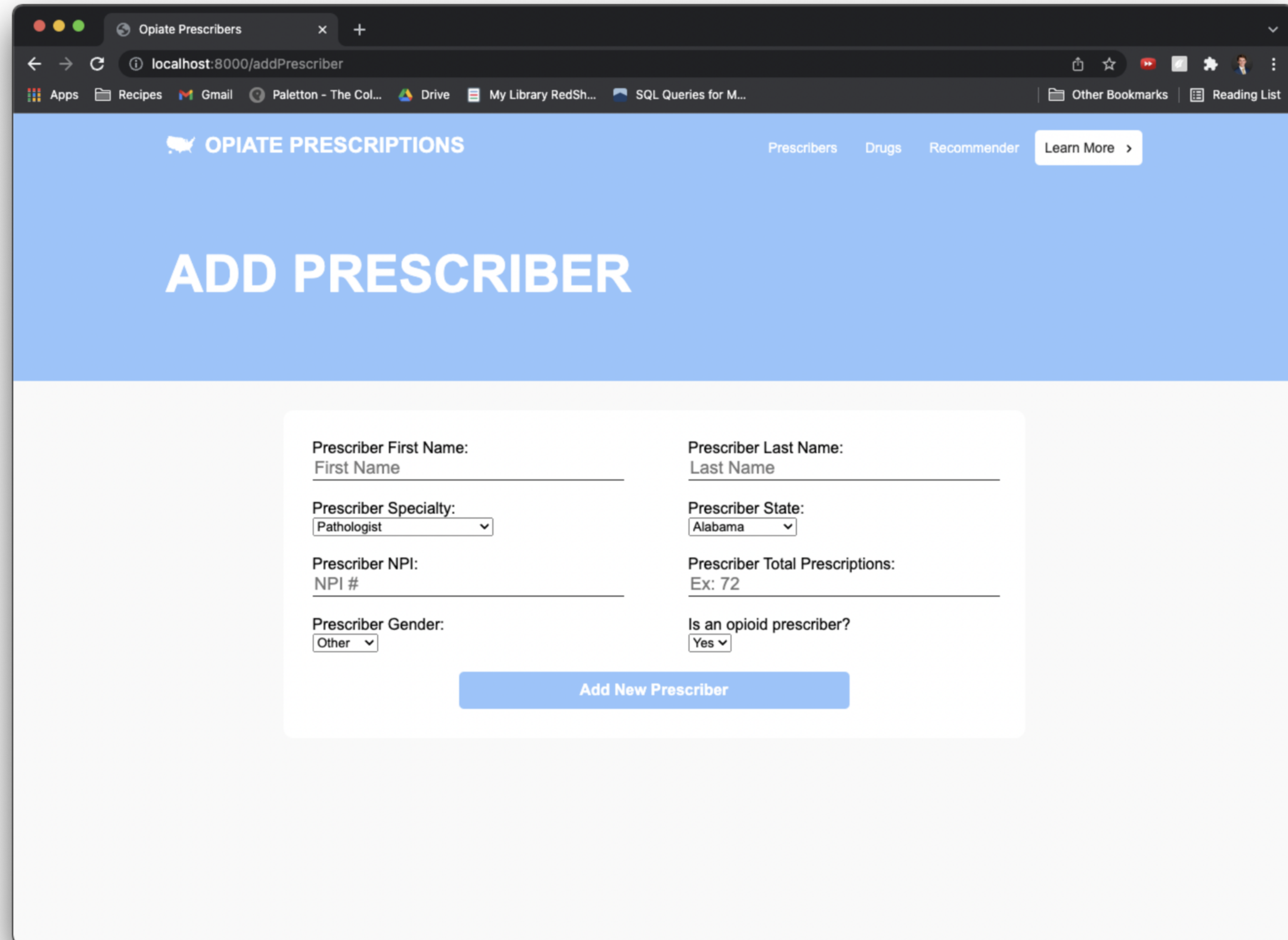
- This page allows you to search for any prescriber in the database. The results are narrowed down by inputting a prescriber's first name, last name, specialty, credentials, state, and gender.

Prescriber Page



- Page lists a specific prescriber and their description (i.e. name, gender, credentials, location, specialty). This also lists all the drugs they prescribed and their corresponding quantity.

Add Prescriber Page



- This page allows authorized personnel to add a prescriber to the database. The user would input all the required information and after they click the add button the prescriber is submitted into the database.

Appendices

Normalized Database

Triple table:

ID	PrescriberID	DrugName	Qty
624075	1992883235	IPRATROPIUM.BROMIDE	35
624076	1992883235	IRBESARTAN	44
624077	1992883235	ISOSORBIDE.MONONITRATE.ER	84
624078	1992883235	JANUMET	22
624079	1992883235	JANUVIA	115
624080	1992883235	KETOCONAZOLE	41
624081	1992883235	KLOR.CON.M20	21
624082	1992883235	LABETALOL.HCL	21
624083	1992883235	LACTULOSE	88
624084	1992883235	LAMOTRIGINE	26
624085	1992883235	LANSOPRAZOLE	16
624086	1992883235	LANTUS	50
624087	1992883235	LANTUS.SOLOSTAR	49
624088	1992883235	LATANOPROST	24
624089	1992883235	LEVEMIR.FLEXPEN	24
624090	1992883235	LEVETIRACETAM	94
624091	1992883235	LEVOFLOXACIN	58
624092	1992883235	LEVOTHYROXINE.SODIUM	425

pd_triple

Prescriber table:

	NPI	Fname	Lname	Gender	State	SpecialtyI[	IsOpioidPr	TotalPrescriptions
1	NPI	Fname	Lname	Gender	State	SpecialtyI[	IsOpioidPr	TotalPrescriptions
2	1E+09	Jameson	Jones	M	MS	1	TRUE	82
3	1E+09	Justine	Chase	F	CO	2	FALSE	349
4	1E+09	Bella	Mcneil	F	GA	3	TRUE	139
5	1E+09	Yoder	Juan	M	NY	4	TRUE	232
6	1E+09	Simmons	Jake	M	CT	5	TRUE	2391
7	1E+09	Lowe	Winston	M	MO	6	FALSE	381
8	1E+09	Rich	Marvin	M	CA	1	FALSE	14
9	1E+09	Jeremy	Jax	M	TX	7	FALSE	13
10	1E+09	Santana	Royce	M	OH	8	TRUE	151
11	1E+09	Ashlynn	Hoffman	F	NC	9	FALSE	1300
12	1E+09	Black	Rowan	M	OR	1	FALSE	18
13	1E+09	Michael	Rodney	M	TX	10	FALSE	133
14	1E+09	Mcintosh	Donte	M	IL	6	FALSE	621
15	1E+09	Wong	Eden	M	PA	1	TRUE	69
16	1E+09	Ayers	Frank	M	OR	11	TRUE	359
17	1E+09	Mayra	Wiley	F	GA	5	TRUE	1950
18	1E+09	Brooklynn	Johnston	F	TX	5	TRUE	991
19	1E+09	Carly	Rowe	F	FL	3	TRUE	12

pd_Prescriber (2)

Specialty table:

Specialty	SpecialtyID
Dentist	1
Pulmonar	2
Nurse Pra	3
Emergenc	4
Family Pra	5
Ophthalm	6
Counselor	7
Orthopedi	8
Endocrino	9
Otolaryng	10
Internal M	11
Psychiatry	12
Physician /	13
Dermatolc	14
Specialist	15
Neurology	16
Critical Ca	17
Infectious	18

pd_Specialty

Credentials table:

Credential	Credentials
1	DMD
2	PA
3	MD
4	NP
5	DDS
6	LPC
7	MED
8	DO
9	PHD
10	ARNP
11	FNP
12	C
13	RPA
14	MPH
15	BC
16	APN
17	BSN
18	MSN

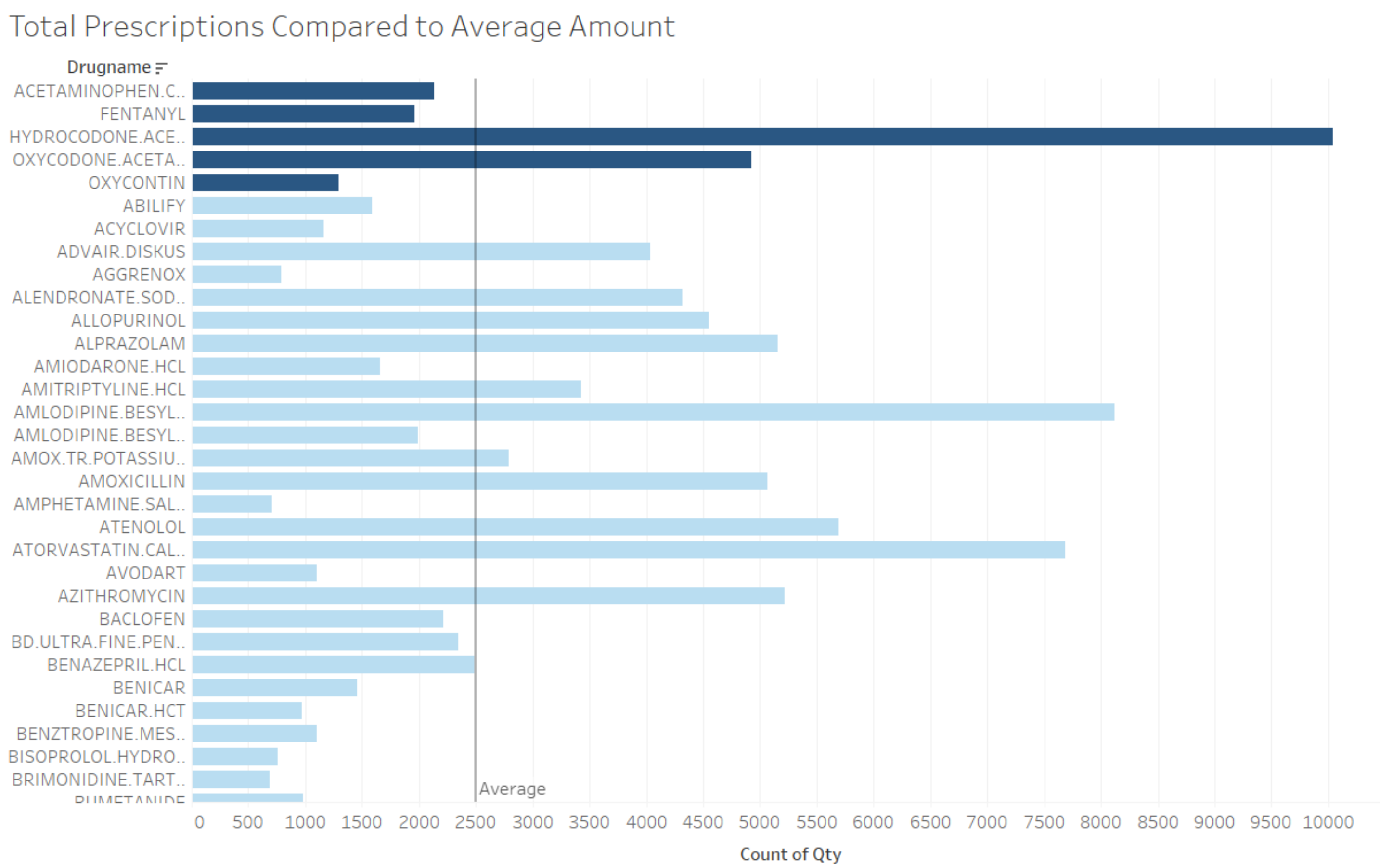
pd_Credentials

Prescriber-Credential table:

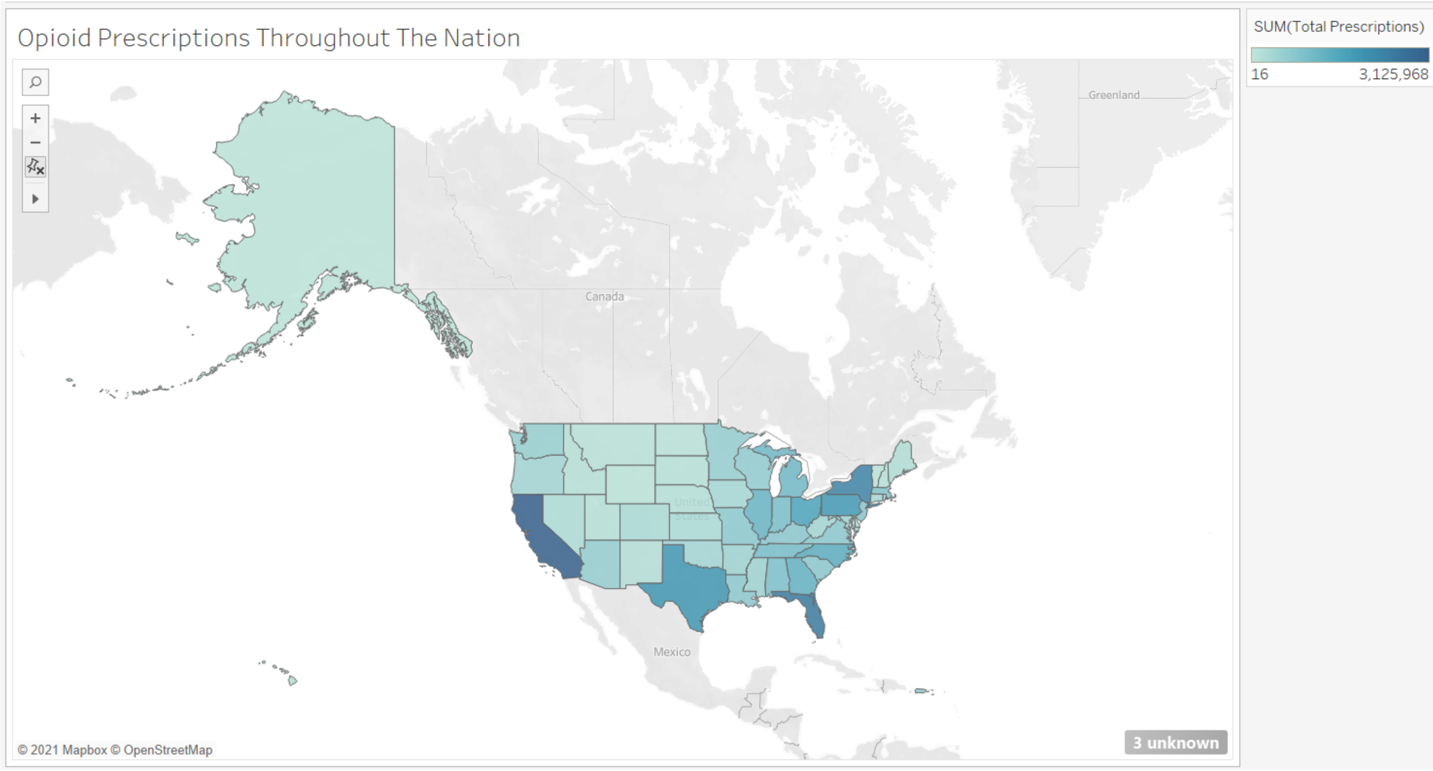
NPI	CredentialID
1003002320	1
1003002320	2
1003004771	3
1003008475	4
1003009630	3
1003016270	3
1003019019	3
1003022872	5
1003023193	6
1003023193	7
1003024894	8
1003026055	3
1003026055	9
1003027038	8
1003034927	3
1003037979	1
1003038332	3
1003042805	3

pd_Prescriber_Credential

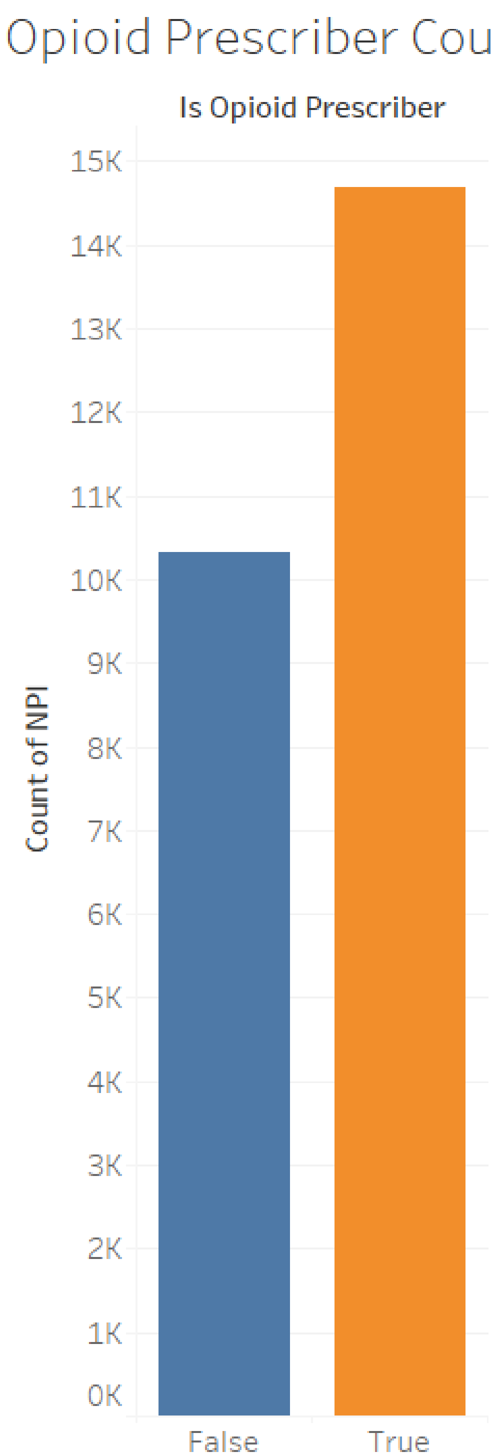
Tableau Chart and Tables



https://public.tableau.com/shared/W3SGG9Z4S?:display_count=n&:origin=viz_share_link



https://public.tableau.com/views/OpioidPrescriptionThroughoutNation/OpioidPrescriptionsThroughoutTheNation?:language=en-US&publish=yes&:display_count=n&:origin=viz_share_link



https://public.tableau.com/views/IntexOpioid/OpioidPrescriberCount?:language=en-US&publish=yes&:display_count=n&:origin=viz_share_link