Data Analysis – D211

Gooden, Nina S. Gooden [ID #: 009823504]

Dr. David Gagner

Data Analysis – D211

## **Interactive Data Dashboard**

* I will be using two data sets: WGU’s provided *Medical* data and *Diabetic Patients' Re-admission Prediction*. (TAYAL, 2020)
* 4 data representations: Included 8.
* 2 different controls: Included 4.
* 2 different KPIs: Identified 2 KPIs on dashboard: Duration of initial stays, Readmission rate.

1. Data sets attached to submission.
2. Dashboard Installation: Installing the dashboard is unnecessary, as it is available to view via Tableau Public. Simply click [this link](https://public.tableau.com/app/profile/nina8405/viz/ADVANCEDDATAACQUISITION_v3/DiabeticData?publish=yes) and you will be able to view the dashboard.
3. The dashboard consists of 8 visualizations and 4 user controls. Starting from the upper lefthand corner:
   1. Total Diabetic Patients: The total number of patients in the dataset.
      1. Affected by the Gender and Age toggle and will adjust to user selections.
   2. ReAdmis: The overall number of patients that were (Yes) and were not (No) readmitted according to the data.
      1. Affected by ReAdmis, Gender, and Age selections.
   3. Days Bars: Show the percentage of days spent on the initial visit.
      1. Affected by ReAdmis, Gender, and Age selections.
   4. Demographics: Shows the breakdown of Race and Gender.
      1. Affected by ReAdmis and Gender selections.
   5. Age Bar: Shows the distribution of patients within each age group.
      1. Affected by ReAdmis, Gender, and Age selections.
      2. Also reflected in Min-Max legend.
   6. Area: Shows breakdown of patient data per area type.
      1. Affected by ReAdmis, Gender, Age, and Area selections.
      2. Also reflected in Pie chart.
4. Creating the datasets with SQL was accomplished with the following code, as well as the Postgres Import/Export Functions. My cleaned data is also attached to this submission.

#First, I created the diabetic\_data table so that I could import it into Postgres

-- Table: public.diabetic\_data

-- DROP TABLE IF EXISTS public.diabetic\_data;

CREATE TABLE IF NOT EXISTS public.diabetic\_data

(

race "char" DEFAULT 'U'::"char",

gender "char" DEFAULT 'U'::"char",

age text COLLATE pg\_catalog."default",

time\_in\_hospital numeric,

medical\_specialty text COLLATE pg\_catalog."default",

num\_lab\_procedures integer,

num\_procedures integer,

num\_medications integer,

number\_outpatient integer,

number\_emergency integer,

number\_inpatient integer,

number\_diagnoses integer,

max\_glu\_serum text COLLATE pg\_catalog."default",

"A1Cresult" text COLLATE pg\_catalog."default",

metformin text COLLATE pg\_catalog."default",

repaglinide text COLLATE pg\_catalog."default",

nateglinide text COLLATE pg\_catalog."default",

chlorpropamide text COLLATE pg\_catalog."default",

glimepiride text COLLATE pg\_catalog."default",

acetohexamide text COLLATE pg\_catalog."default",

glipizide text COLLATE pg\_catalog."default",

glyburide text COLLATE pg\_catalog."default",

tolbutamide text COLLATE pg\_catalog."default",

pioglitazone text COLLATE pg\_catalog."default",

rosiglitazone text COLLATE pg\_catalog."default",

acarbose text COLLATE pg\_catalog."default",

miglitol text COLLATE pg\_catalog."default",

troglitazone text COLLATE pg\_catalog."default",

tolazamide text COLLATE pg\_catalog."default",

examide text COLLATE pg\_catalog."default",

citoglipton text COLLATE pg\_catalog."default",

insulin text COLLATE pg\_catalog."default",

"glyburide-metformin" text COLLATE pg\_catalog."default",

"glipizide-metformin" text COLLATE pg\_catalog."default",

"glimepiride-pioglitazone" text COLLATE pg\_catalog."default",

"metformin-rosiglitazone" text COLLATE pg\_catalog."default",

"metformin-pioglitazone" text COLLATE pg\_catalog."default",

change text COLLATE pg\_catalog."default",

"diabetesMed" text COLLATE pg\_catalog."default",

readmitted text COLLATE pg\_catalog."default",

encounter\_id text COLLATE pg\_catalog."default",

patient\_nbr text COLLATE pg\_catalog."default",

weight text COLLATE pg\_catalog."default",

admission\_type\_id text COLLATE pg\_catalog."default",

discharge\_disposition\_id text COLLATE pg\_catalog."default",

admission\_source\_id text COLLATE pg\_catalog."default",

payer\_code text COLLATE pg\_catalog."default",

diag\_1 text COLLATE pg\_catalog."default",

diag\_2 text COLLATE pg\_catalog."default",

diag\_3 text COLLATE pg\_catalog."default"

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.diabetic\_data

OWNER to postgres;

#Next, I did the same for the medical data—I needed to use my own installed version of Postgres because my computer would not run the Launched version

-- Table: public.medical\_data

-- DROP TABLE IF EXISTS public.medical\_data;

CREATE TABLE IF NOT EXISTS public.medical\_data

(

caseorder integer NOT NULL,

customer\_id text COLLATE pg\_catalog."default",

interaction text COLLATE pg\_catalog."default",

uid text COLLATE pg\_catalog."default",

city text COLLATE pg\_catalog."default",

state text COLLATE pg\_catalog."default",

county text COLLATE pg\_catalog."default",

zip integer,

lat numeric,

lng numeric,

population integer,

area text COLLATE pg\_catalog."default",

timezone text COLLATE pg\_catalog."default",

job text COLLATE pg\_catalog."default",

children integer,

age integer,

income numeric,

marital text COLLATE pg\_catalog."default",

gender text COLLATE pg\_catalog."default",

readmis text COLLATE pg\_catalog."default",

vitd\_levels numeric,

doc\_visits integer,

full\_meals\_eaten integer,

vitd\_supp integer,

soft\_drink text COLLATE pg\_catalog."default",

initial\_admin text COLLATE pg\_catalog."default",

highblood text COLLATE pg\_catalog."default",

stroke text COLLATE pg\_catalog."default",

complication\_risk text COLLATE pg\_catalog."default",

overweight text COLLATE pg\_catalog."default",

arthritis text COLLATE pg\_catalog."default",

diabetes text COLLATE pg\_catalog."default",

hyperlipidemia text COLLATE pg\_catalog."default",

backpain text COLLATE pg\_catalog."default",

anxiety text COLLATE pg\_catalog."default",

allergic\_rhinitis text COLLATE pg\_catalog."default",

reflux\_esophagitis text COLLATE pg\_catalog."default",

asthma text COLLATE pg\_catalog."default",

services text COLLATE pg\_catalog."default",

initial\_days numeric,

totalcharge numeric,

additional\_charges numeric,

item1 integer,

item2 integer,

item3 integer,

item4 integer,

item5 integer,

item6 integer,

item7 integer,

item8 integer,

CONSTRAINT medic\_data\_pkey PRIMARY KEY (caseorder)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.medical\_data

OWNER to postgres;

#I began cleaning the data up in a separate table—this is the table I would use for my final export. The diabetic data needed to be renamed and given default, values for the NULLs.

CREATE TABLE diabetic\_cln

AS (SELECT encounter\_id, age, gender, time\_in\_hospital, readmitted, race

FROM diabetic\_data);

ALTER TABLE diabetic\_cln

RENAME COLUMN time\_in\_hospital TO Initial\_days;

ALTER TABLE diabetic\_cln

RENAME COLUMN readmitted TO readmis;

ALTER TABLE diabetic\_cln

ADD COLUMN area text DEFAULT 'Unknown';

ALTER TABLE diabetic\_cln

ADD COLUMN state text DEFAULT 'Unknown';

UPDATE diabetic\_cln

SET gender = 'Female'

WHERE gender LIKE 'F'

UPDATE diabetic\_cln

SET gender = 'Male'

WHERE gender LIKE 'M'

UPDATE diabetic\_cln

SET gender = 'Nonbinary'

WHERE gender LIKE 'U'

UPDATE diabetic\_cln

SET readmis = 'No'

WHERE readmis LIKE 'NO'

UPDATE diabetic\_cln

SET readmis = 'Yes'

WHERE readmis LIKE '%30'

#I only wanted the medical data that was TRUE for diabetes, so I filtered that out and also put it in a secondary table.

CREATE TABLE medical\_cln

AS (SELECT customer\_id, age, gender, initial\_days, readmis, area, state, diabetes

FROM medical\_data

WHERE diabetes LIKE 'Yes'

)

#Then I cleaned up the data so it would match the diabetic information.

ALTER TABLE medical\_cln

DROP COLUMN diabetes;

ALTER TABLE medical\_cln

ADD COLUMN race text DEFAULT 'Unknown';

CREATE TABLE combined\_cln

AS (SELECT D.encounter\_id, m.customer\_id

FROM diabetic\_cln as D

FULL OUTER JOIN medical\_cln as M

ON D.encounter\_id = m.customer\_id

)

#I also had to create the age groups and convert the medical age to text.

SELECT age,

CASE

WHEN age >= 0 and age <= 10 THEN '[0-10)'

WHEN age > 10 and age <= 20 then '[10-20)'

WHEN age > 20 and age <= 30 then '[20-30)'

WHEN age > 30 and age <= 40 then '[30-40)'

WHEN age > 40 and age <= 50 then '[40-50)'

WHEN age > 50 and age <= 60 then '[50-60)'

WHEN age > 60 and age <= 70 then '[60-70)'

WHEN age > 70 and age <= 80 then '[70-80)'

WHEN age > 80 and age <= 90 then '[80-90)'

WHEN age > 90 and age <= 100 then '[90-100)'

END age\_group

#I created one last column for the JOINED data

-- Table: public.combined\_cln

-- DROP TABLE IF EXISTS public.combined\_cln;

CREATE TABLE IF NOT EXISTS public.combined\_cln

(

encounter\_id text COLLATE pg\_catalog."default",

readmis text COLLATE pg\_catalog."default",

age text COLLATE pg\_catalog."default",

area text COLLATE pg\_catalog."default",

gender text COLLATE pg\_catalog."default",

initial\_days integer,

race text COLLATE pg\_catalog."default",

state text COLLATE pg\_catalog."default"

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.combined\_cln

OWNER to postgres;

FROM medical\_cln

ORDER BY customer\_id;

#And pulled all the information I needed into that clean, empty table. I attempted to join the data but ultimately abandoned the effort to do so in Tableau instead.

SELECT M.area, M.age, M.gender, M.initial\_days, M.readmis, M.area, M.state, M.race

FROM medical\_cln AS M

UNION ALL

SELECT D.area, D.age, D.gender, D.initial\_days, D.readmis, D.area, D.state, D.race

FROM diabetic\_cln AS D

UNION ALL;

UPDATE combined\_cln C

SET gender = m.gender,

C.initial\_days = m.initial\_days

FROM medical\_cln m

WHERE C.customer\_id = m.customer\_id

## **B. Storytelling**

[XX- SLM1 | D211](https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=518f8ad6-f2a8-4bc1-9580-af650146b076)

## **C. Reflection Paper:**

Nina S. Gooden

Professor David Gagner

Advanced Data Acquisition - D211

December 7, 2022

Advanced Munging with SQL

**(1)** Exploring the relationship between rising hospitalization costs and the cost of living with diabetes is something that has always been important and fascinating to me. According to the CDC, 1 in 10 Americans have the disease and it is the seventh leading cause of death in the United States. In addition, the disease costs #327 billion in medical costs and lost work/wages a year (The Facts, Stats, and Impacts of Diabetes, 2022). Some of those costs and burdens fall onto the medical centers that provide care for these individuals. As such, running analyses on the cost of these efforts is relevant to real-world needs.

**(2)** For my evaluation I utilized PostgreSQL and Tableau as these are two tools I am familiar with from my professional work. Working with SQL is easier on Tableau than working manually with just the code because Tableau allows you to create joins with the interface, rather than needing to get the code to function correctly. I did both in my work and found that the Tableau interface was user friendly and easy to manage.

**(3)** Cleaning and preparing the data for analysis in PostgreSQL was a straightforward process. I knew what I wanted the data to look like when I was finished, so I quickly created a table outline. Next, I had to import all of my data manually, as I could not get the school-provi9ded system to work. This was done by creating the tables and then using the import wizard to load in each of the data sets. After that, I simply needed to clean up the data so they would work together. I changed all Null values to “Unknown,” created dummy columns for what each set was missing, and then made visualization changes so that the information was presented in the same way (changed column names, converted the data to the same groups, changed Boolean answers to match). I then exported the data into two sheets.

**(4)** I used the import function in Tableau to add the medical\_clean and diabetic\_clean data. Before I started building the dashboard, I joined the two sheets on the encounter\_id and customer\_id columns. After the data was uploaded, I was able to drag and drop the contents of the datasets into Tableau to create all eight of my visualizations. I created a default web view sized dashboard and arranged my elements in a way that was readable and aesthetically pleasing.

**(5)** By evaluating the patient data for diabetic readmission, I have created an opportunity for executive decision makers to target those patients that were identified as having a higher likelihood of readmission for corrective action. In addition, I highlighted the fact that people who spend less time in their initial stays are more likely to have to come back to the medical center in question. This information has the potential to cause medical professionals to run more tests and spend more time evaluating potential concerns in order to lower the risk of readmission.

**(6)** The data sets that I used for my evaluation were not as complete as I would have liked. There were many Null values in the second set and I created additional ones in order to combine the sheets. This analysis is limited by the data that was available and by the way that data was gathered.

## **D. CITATIONS**

PostgreSQL Change Column Type: Step-by-Step Examples. (2020, July 19). PostgreSQL Tutorial. https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-change-column-type/

Error : ERROR: table name specified more than once. (2011, May 15). Stack Overflow. https://stackoverflow.com/questions/6005635/error-error-table-name-specified-more-than-once

The Facts, Stats, and Impacts of Diabetes. (2022, January 24). Centers for Disease Control and Prevention. https://www.cdc.gov/diabetes/library/spotlights/diabetes-facts-stats.html