Project Journal

Natasha Sharma

Data Analyst Intern at IPRO

1) **Describe what is in the laboratory claim datasets: what elements are structured, semi-structured, and unstructured and how will you approach them?**

The set contains the following fields: -

1) **mbi_id_orig** = Medicare Beneficiary Identifier (MBI) - A unique identifier for individuals enrolled in Medicare.

2) **DOS** = Date of Service - The date on which medical services were rendered.

3) **Accession_Number** = A unique identifier assigned to a specific sample or test in a lab for tracking purposes.

4) **Requisition Number** = A unique identifier for the request form or order for lab tests or medical services.

5) **Lab_Code** = The code identifying the specific lab where the test or service was performed.

6) **Date_of_collection** = The date on which the sample (e.g., blood, urine) was collected from the patient.

7) **External_Pat_ID** = An identifier assigned to a patient by an external or non-primary system, often for tracking or reference purposes.

8) **Pat_State** = The state where the patient resides.

9) **Pat_Zip** = The ZIP code of the patient's residence.

10) **Date_of_Birth** = The patient's date of birth.

11) **Age** = The patient's Age.

12) **Gender** = The patient's Gender.

13) **Bill_Code** = Billing code used for invoicing and insurance claims.

14) **Policy_Number** = The patient's policy number, which is a unique identifier assigned to an insurance policy that helps the insurer track the insured's coverage.

15) **Medicaid_No** = The patient's Medicaid number, used for billing and identification purposes. (Health Coverage Number for the Patient provided by federal government. Medicaid provides health coverage to millions of Americans, including eligible low-income adults, children, pregnant women, elderly adults and people with disabilities. Medicaid is administered by states, according to federal requirements. The program is funded jointly by the states and the federal government.)

16) **Medicare_No** = The patient's Medicare number, used for billing and identification purposes. (Medicare card has a unique number that's distinct from your Social Security Number. This helps protect your identity.)

17) **Phy_Name** = The name of the physician who ordered the test or service.

18) **UPIN** = Unique Physician Identification Number - a deprecated identifier for physicians in the U.S., replaced by the NPI.

19) **DIAG_CODE1** = Primary diagnosis code (e.g., ICD-10) indicating the main reason for the test or service. (ICD-10 Code - The International Classification of Diseases, Tenth Revision (ICD-10) is used by healthcare providers to classify and code every disease, symptom, and injury to submit insurance claims or prior authorizations.

20) **DIAG_CODE2:** Secondary diagnosis code, indicating additional conditions or reasons for the test or service.

21) **DIAG_CODE3:** Tertiary diagnosis code, for further additional conditions or reasons for the test or service.

22) **DIAG_CODE4:** Additional diagnosis code.

23) **DIAG_CODE5:** Additional diagnosis code.

24) **DIAG_CODE6:** Additional diagnosis code.

25) **DIAG_CODE7:** Additional diagnosis code.

26) **DIAG_CODE8:** Additional diagnosis code.

27) **DIAG_CODE9:** Additional diagnosis code.

28) **DIAG_CODE10:** Additional diagnosis code.

29) **Local_Profile_Code** = A local code used to identify a specific profile of tests or services.

30) **Standard_Profile_Code** = A standardized code used to identify a specific profile of tests or services.

31) **Profile_Name** = Profile Name is a comprehensive summary of health-related information for an individual patient.

32) **Local_Order_Code** = A local code used to identify a specific test or service order.

33) **Standard_Order_Code** = A standardized code used to identify a specific test or service order.

34) **Order_Name** = The name of the test or service being ordered.

35) **LOINC_Code** = Logical Observation Identifiers Names and Codes - a universal code system for identifying laboratory and clinical observations.

36) **Local_Result_Code** = A local code used to identify a specific test result.

37) **Result_Name** = The name of the test result or parameter being reported.

38) **Result_Value_A** = The actual value or measurement obtained from the test.

39) **Units** = The units of measurement for the test result (e.g., mg/dL, mmol/L).

40) **Ref_Range_Low** = The lower limit of the reference range for the test result, indicating the minimum normal value.

41) **Ref_Range_High** = The upper limit of the reference range for the test result, indicating the maximum normal value.

42) **Ref_Range_Alpha** = The upper -lower limit reference ranges for the test results.

43) **Derived_Abnormal_Flag** = It indicates if the result value is out of the normal range or abnormal (H= High, L=Low and A=Abnormal), often derived from comparison with reference ranges.

44) **CPT_Code** = Current Procedural Terminology code - a standardized code used for reporting medical, surgical, and diagnostic procedures.

45) **COMM-TEXT** = Comments or textual notes related to the test or result.

46) **Ordering_Site_Code** = The code identifying the site or location where the test or service was ordered.

47) **Elig_Member_Id** = An identifier for the eligible member, often used in insurance contexts.

48) **npi** = National Provider Identifier - a unique identifier for healthcare providers in the U.S.

49) **Unique_linker** = A unique identifier used to link related records or data entries.

50) **DM** = Diabetes Mellitus - an indicator if the patient has a history of diabetes.

51) **HTN** = Hypertension - an indicator if the patient has a history of high blood pressure.

52) **DM-HTN** = An indicator if the patient has a history of both diabetes and hypertension.

Descriptive Analysis of the dataset: -

n-size of the datasets

HTNDM_202301Q.csv = **160906 rows × 52 columns**

HTNDM_202302Q.csv = **168480 rows × 52 columns**

HTNDM_202303Q.csv = **216252 rows × 52 columns**

HTNDM_202304Q.csv = **261171 rows × 52 columns**

HTNDM_202305Q.csv = **255828 rows × 52 columns**

HTNDM_202306Q.csv = **330331 rows × 52 columns**

HTNDM_202307Q.csv = **207462 rows × 52 columns**

HTNDM_202308Q.csv = **189111 rows × 52 columns**

HTNDM_202309Q.csv = **204430 rows × 52 columns**

HTNDM_202310Q.csv = **236954 rows × 52 columns**

HTNDM_202311Q.csv = **227390 rows × 52 columns**

HTNDM_202312Q.csv = **185278 rows × 52 columns**

HTNDM_202401Q.csv = **243622 rows × 52 columns**

HTNDM_202402Q.csv = **81963 rows × 52 columns**

HTNDM_202403Q.csv = **81963 rows × 52 columns**

HTNDM_202404Q.csv = **75971 rows × 52 columns**

HTNDM_202405Q.csv = **73851 rows × 52 columns**

Columns	Datatypes	Category
mbi_id_orig	object	Categorical
DOS	int64	Continuous
ACCESSION_NUMBER	object	Categorical
REQUISITION_NUMBER	object	Categorical
LAB_CODE	object	Categorical
DATE_OF_COLLECTION	float64	Continuous
EXTERNAL_PAT_ID	object	Categorical
PAT_STATE	object	Categorical
PAT_ZIP	float64	Continuous
DATE_OF_BIRTH	int64	Continuous
AGE	int64	Continuous
GENDER	object	Categorical
BILL_CODE	object	Categorical
POLICY_NUMBER	object	Categorical
MEDICAID_NO	object	Categorical
MEDICARE_NO	object	Categorical
PHY_NAME	object	Categorical
UPIN	object	Categorical
DIAG_CODE1 - DIAG_CODE10	object	Categorical
LOCAL_PROFILE_CODE	object	Categorical
STANDARD_PROFILE_CODE	object	Categorical
PROFILE_NAME	object	Categorical
LOCAL_ORDER_CODE	object	Categorical
STANDARD_ORDER_CODE	object	Categorical
ORDER_NAME	object	Categorical
LOINC_CODE	object	Categorical
LOCAL_RESULT_CODE	object	Categorical
RESULT_NAME	object	Categorical
RESULT_VALUE_A	object	Continuous
UNITS	object	Categorical
REF_RANGE_LOW	float64	Continuous
REF_RANGE_HIGH	float64	Continuous
REF_RANGE_ALPHA	object	Categorical
DERIVED_ABNORMAL_FLAG	object	Categorical
CPT_CODE	object	Categorical
COMM_TEXT	object	Text
ORDERING_SITE_CODE	object	Categorical
Elig_Member_Id	object	Categorical
npi	float64	Continuous
unique_linker	object	Categorical
DM	int64	Binary
HTN	int64	Binary

DM_HTN	float64	Binary

Note: - RESULT_VALUE_A could potentially be continuous, but as it's of type object, it suggests mixed or categorical data

#Classification of dataset fields: -

Almost all the columns in the dataset were structured, as these were organized and easily searchable in a tabular format (rows and columns). However, some of the columns like DIAG_CODE1 - DIAG_CODE10 may fall under the category of semi-structured, as semi-structured data does not conform to a rigid structure but has some organizational properties that make it easier to analyze. Therefore, depending on if these are free-text entries or codes that might have some structure but can vary in format.

Moreover, the column COMM_TEXT would fall under the unstructured format, as this contains unstructured textual comments or notes.

With this project, the approach that used was: -

- The first step was ensuring we understood the data through its description. So, we searched for the descriptions to understand what these fields mean.
- The second step was ensuring that the data was ready for analysis. This step involved cleaning the dataset and data wrangling, which was not too much for this project because most of the data was already in the right shape.
- Once the data is ready, it will be analyzed through visualization using Tableau.

2) What actions did you take to successfully prepare the dataset for analysis?

Since most of the data was already in the structured format, we don't need to make a lot of changes. Also, as per the project requirement, the following steps were performed to create the data for CKD Heatmap: -

- We created a list of all the data files.
- Created a data frame to read all the .csv files and concatenated all the datafiles to create the final data frame 'final_df' so that all the data from all 17 files is aggregated into one data frame. (There were some extra unwanted columns in the concatenated

dataframe, like 54 columns, which should be 52, so we also handled those unwanted columns)

- Then, we created a subset of the final_df' named 'df1' to keep only the required columns. For example: - 'mbi_id_orig', 'DOS', 'RESULT_NAME', 'RESULT_VALUE_A', 'CPT_CODE', 'LOINC_CODE', 'ORDER_NAME', 'DM', 'HTN' and 'DM_HTN'.
- Then, we created a separate CSV file named "Final_data.csv," which speeds up the process of loading the data.
- We loaded the Final_data.csv file to create a data frame named "df," which contained all 10 columns and 3200963 rows.
- Changed the date format for "DOS" column from integer to datetime. For example: - '20240625' to '2024-06-25', so that the code can read the date and provide the result values associated with it.
- Created two separate dataframe named 'df_egfr' and 'df_uacr', which contains all the result values for the LOINC_CODE '98979-8 for egfr and '9318-7' for uacr.
- Also removed the duplicate values of the 'mbi_id_orig' to get the unique patients with those LOINC codes.
- Finally merged the two dataframes and created a df_final dataframe.
- Then, checked if there are any string values in the EGFR and UACR columns, as we only required numeric values of the test results to condition them with the LOINC_Code. After checking this, we ensured to convert the EGFR and UACR columns into numeric and dropping non-numeric and null values.
- Created two new columns CKD_Stage and UACR_Level.
- The final dataframe named df_final contains 9973 rows and 7 columns named 'mbi_id_orig', 'DOS_egfr','EGFR', 'DOS_uacr','UACR', 'CKD_Stage' and 'UACR_Level'.
- Finally, created the CSV file named 'EGFR_UACR', which is our final data to be used for generating CKD HeatMap.

Code for CKD Heatmap: -

```
import pandas as pd

import numpy as np


# List of filenames

datafiles = ['N:\DATA FILES\HTNDM_202301Q.csv', 'N:\DATA
FILES\HTNDM_202302Q.csv', 'N:\DATA FILES\HTNDM_202303Q.csv', 'N:\DATA
```

FILES\HTNDM_202304Q.csv', 'N:\DATA FILES\HTNDM_202305Q.csv', 'N:\DATA FILES\HTNDM_202306Q.csv', 'N:\DATA FILES\HTNDM_202307Q.csv', 'N:\DATA FILES\HTNDM_202308Q.csv', 'N:\DATA FILES\HTNDM_202309Q.csv', 'N:\DATA FILES\HTNDM_202310Q.csv','N:\DATA FILES\HTNDM_202311Q.csv', 'N:\DATA FILES\HTNDM_202312Q.csv', 'N:\DATA FILES\HTNDM_202401Q.csv', 'N:\DATA FILES\HTNDM_202402Q.csv', 'N:\DATA FILES\HTNDM_202403Q.csv', 'N:\DATA FILES\HTNDM_202404Q.csv', 'N:\DATA FILES\HTNDM_202405Q.csv']

#Fixing the number of columns

```
# Inspect columns of each file

for file in datafiles:

    df = pd.read_csv(file, low_memory=False)

    print(f"{file} columns: {df.columns.tolist()}")
```

 # Define the correct column names

```
correct_columns = ["mbi_id_orig", "DOS","ACCESSION_NUMBER",
"REQUISITION_NUMBER", "LAB_CODE","DATE_OF_COLLECTION",
"EXTERNAL_PAT_ID", "PAT_STATE", "PAT_ZIP", "DATE_OF_BIRTH", "AGE",
"GENDER", "BILL_CODE", "POLICY_NUMBER", "MEDICAID_NO", "MEDICARE_NO",
"PHY_NAME", "UPIN", "DIAG_CODE1", "DIAG_CODE2", "DIAG_CODE3",
"DIAG_CODE4", "DIAG_CODE5", "DIAG_CODE6", "DIAG_CODE7", "DIAG_CODE8",
"DIAG_CODE9", "DIAG_CODE10", "LOCAL_PROFILE_CODE",
"STANDARD_PROFILE_CODE", "PROFILE_NAME", "LOCAL_ORDER_CODE",
"STANDARD_ORDER_CODE", "ORDER_NAME", "LOINC_CODE",
"LOCAL_RESULT_CODE", "RESULT_NAME", "RESULT_VALUE_A", "UNITS",
"REF_RANGE_LOW", "REF_RANGE_HIGH", "REF_RANGE_ALPHA",
"DERIVED_ABNORMAL_FLAG", "CPT_CODE", "COMM_TEXT",
"ORDERING_SITE_CODE", "Elig_Member_Id", "npi", "unique_linker", "DM", "HTN",
"DM_HTN"]
```

 # Function to standardize columns

```
def standardize_columns(df, correct_columns):
```

```
    # Rename columns to match the correct ones
    df.columns = [col.strip() for col in df.columns]
    # Reindex DataFrame to have missing columns filled with NaN and extra columns dropped
    return df.reindex(columns=correct_columns)
```

 # Read, standardize, and concatenate all files

```
dfs = []
for file in datafiles:
    df = pd.read_csv(file, low_memory=False)
    df = standardize_columns(df, correct_columns)
    dfs.append(df)
```

 # Concatenate all DataFrames

```
final_df = pd.concat(dfs, ignore_index=True)
```

#Keeping only the required columns and setting the columns as copy.

```
df = final_df[['mbi_id_orig', 'DOS','RESULT_NAME', 'RESULT_VALUE_A', 'CPT_CODE',
'LOINC_CODE', 'ORDER_NAME','DM', 'HTN', 'DM_HTN']].copy()
```

#Saving the dataframe to a csv file.

```
df1.to_csv ('Final_data.csv', index=False)
```

#Reading the Final_data.csv data and converting it into a dataframe df.

```
df = pd.read_csv("Final_data.csv")
```

#Changing the date format

```
df.loc[:, 'DOS'] = pd.to_datetime(df['DOS'], format='%Y%m%d')
```

#Seperating EGFR values based on LOIN_CODE

```
df_egfr = df[df['LOINC_CODE'] == '98979-8']
```

#Dropping the duplicates based on mbi_id_org and sorting the value based on DOS

```
df_egfr = df_egfr.sort_values('DOS').drop_duplicates('mbi_id_orig', keep = 'last')

df_final_egfr = df_egfr[['mbi_id_orig','DOS','RESULT_VALUE_A']]

df_final_egfr.rename(columns={'RESULT_VALUE_A':'EGFR'}, inplace=True)
```

#Seperating UACR values

```
df_uacr = df[df['LOINC_CODE'] == '9318-7']
```

#Dropping the duplicates based on mbi_id_org and sorting the value based on DOS

```
df_uacr = df_uacr.sort_values('DOS').drop_duplicates('mbi_id_orig', keep = 'last')

df_final_uacr = df_uacr[['mbi_id_orig','DOS','RESULT_VALUE_A']]

df_final_uacr.rename(columns={'RESULT_VALUE_A':'UACR'}, inplace=True)
```

#Merging the two dataframes

```
df_final = pd.merge(df_final_egfr, df_final_uacr, on='mbi_id_orig',how='left', suffixes=('_egfr',
'_uacr'))
```

#Converting the EGFR and UACR columns into numeric datatype and dropping the null values

```
df_final['EGFR'] = pd.to_numeric(df_final['EGFR'], errors='coerce')

df_final = df_final.dropna(subset=['EGFR'])
```

```
df_final['UACR'] = pd.to_numeric(df_final['UACR'], errors='coerce')
df_final = df_final.dropna(subset=['UACR'])

# Creating CKD_Stage based on EGFR values
conditions_ckd = [
    (df_final['EGFR'] >= 90),
    (df_final['EGFR'] >= 60) & (df_final['EGFR'] < 90),
    (df_final['EGFR'] >= 45) & (df_final['EGFR'] < 60),
    (df_final['EGFR'] >= 30) & (df_final['EGFR'] < 45),
    (df_final['EGFR'] >= 15) & (df_final['EGFR'] < 30),
    (df_final['EGFR'] < 15)
]
choices_ckd = ['G1', 'G2', 'G3a', 'G3b', 'G4', 'G5']
df_final['CKD_Stage'] = np.select(conditions_ckd, choices_ckd, default='nan')

# Create UACR_Level based on UACR values
conditions_uacr = [
    (df_final['UACR'] <= 30),
    (df_final['UACR'] > 30) & (df_final['UACR'] <= 301),
    (df_final['UACR'] > 300)
]
choices_uacr = ['A1', 'A2', 'A3']
df_final['UACR_Level'] = np.select(conditions_uacr, choices_uacr, default='nan')
```

#The final dataframe with the required columns (mbi_id_orig, DOS_egfr, EGFR, DOS_uacr, UACR, CKD_Stage, UACR_Level)

```
df_final
```

#Coverting the df_final into csv file to create the CKD Heatmap

```
df_final.to_csv('EGFR_UACR.csv', index=False)
```

Validation check

 # Define the function

```
def determine_score(ckd_stage, uacr_level):
    if ckd_stage in ('G1', 'G2') and uacr_level == 'A1':
        return 1
    elif (ckd_stage == 'G3a' and uacr_level == 'A1') or (ckd_stage in ('G1', 'G2') and uacr_level == 'A2'):
        return 2
    elif (ckd_stage == 'G3b' and uacr_level == 'A1') or (ckd_stage == 'G3a' and uacr_level == 'A2') or (ckd_stage in ('G1', 'G2') and uacr_level == 'A3'):
        return 3
    else:
        return 4
```

 # Apply the function to the DataFrame

```
df_final['CKD Crosswalk'] = df_final.apply(lambda row: determine_score(row['CKD_Stage'], row['UACR_Level']), axis=1)
```

#Checking the unique values of new column CKD Crosswalk

```
df_final['CKD Crosswalk'].unique()
```

 # Get the count of each unique value in 'CKD Crosswalk'

```
value_counts = df_final['CKD Crosswalk'].value_counts()

value_counts
```

 # Group by the columns and count the occurrences

```
counts = df_final.groupby(['CKD_Stage', 'UACR_Level', 'CKD
Crosswalk']).size().reset_index(name='Count')
```

 # Print the resulting DataFrame

```
print(counts)
```

#Validation: Checking if we have any values of G5A1

```
df_final['new_column'] = df_final['CKD_Stage'] + df_final['UACR_Level']

df_final['new_column'].unique()

# There are no values, therefore our data is correct.
```

#Validating the min and max dos for egfr and uacr

```
df_final['DOS_egfr'].min()

df_final['DOS_uacr'].min()

df_final['DOS_uacr'].max()

df_final['DOS_uacr'].max()
```

#Reason to choose LOINC_CODE

The initial CPT_CODE was 82565, 82610, 80047, 80048 for EGFR and 82043, 82570 for UACR.

With the CPT_CODE 82565 for EGFR, we have two associated LOINC_CODE '2160-0' and '98979-8'. '2160-0' has the result name CREATININE and abnormal values less than 1, like 0.61 mg/dl, etc., whereas the '98979-8' code has the correct result values and contains the result name 'EGFR.'

The other CPT_CODEs for EGFR that we found were 80053 and 80048. However, they have many different LOINC_CODEs associated with them, different result names, and abnormal result values. Similarly, for the CPT_CODE 82610 and 80047 has no LOINC_CODE associated with it.

Since we are only looking for the EGFR values, we will only use '98979-8' LOINC_CODE to find all the EGFR values. Similarly, the LOINC_CODE for UACR values is '9318-7' with no associated CPT_CODE value. Therefore, we are not using the CPT_CODE but only the LOINC_CODE '9318-7' for UACR values. Since the LOINC_CODE is more specific, that is why we are not using CPT_CODE.

Similarly, the same issue occurred with the order and result names. We found that the RESULT_NAME was more specific than the ORDER_NAME. For example- The RESULT_NAME is EGFR, and the ORDER_NAME associated with it is COMPREHENSIVE METABOLIC PANEL. However, the RESULT_NAME CALCIUM, CHLORIDE, GLUCOSE, etc. also have the ORDER_NAME as COMPREHENSIVE METABOLIC PANEL, which makes it confusing to use ORDER_NAME. Therefore, we will not be using the ORDER_NAME.

Code for Geo-spatial Map: -

#Importing the required libraries

```
import pandas as pd

import numpy as np
```

 # List of filenames

```
datafiles = ['N:\DATA FILES\HTNDM_202301Q.csv', 'N:\DATA
FILES\HTNDM_202302Q.csv', 'N:\DATA FILES\HTNDM_202303Q.csv', 'N:\DATA
FILES\HTNDM_202304Q.csv', 'N:\DATA FILES\HTNDM_202305Q.csv', 'N:\DATA
```

FILES\HTNDM_202306Q.csv', 'N:\DATA FILES\HTNDM_202307Q.csv', 'N:\DATA FILES\HTNDM_202308Q.csv', 'N:\DATA FILES\HTNDM_202309Q.csv', 'N:\DATA FILES\HTNDM_202310Q.csv','N:\DATA FILES\HTNDM_202311Q.csv', 'N:\DATA FILES\HTNDM_202312Q.csv', 'N:\DATA FILES\HTNDM_202401Q.csv', 'N:\DATA FILES\HTNDM_202402Q.csv', 'N:\DATA FILES\HTNDM_202403Q.csv', 'N:\DATA FILES\HTNDM_202404Q.csv', 'N:\DATA FILES\HTNDM_202405Q.csv']

datafiles

Fixing the number of Columns. (In the output, the number of columns were 54, but it should be 52, that means we have two extra columns, so we are fixing it here)

 # Inspect columns of each file

```
for file in datafiles:
    df = pd.read_csv(file, low_memory=False)
    print(f"{file} columns: {df.columns.tolist()}")
```

 # Define the correct column names

```
correct_columns = ["mbi_id_orig", "DOS","ACCESSION_NUMBER",
"REQUISITION_NUMBER", "LAB_CODE","DATE_OF_COLLECTION",
"EXTERNAL_PAT_ID", "PAT_STATE", "PAT_ZIP", "DATE_OF_BIRTH", "AGE",
"GENDER", "BILL_CODE", "POLICY_NUMBER", "MEDICAID_NO", "MEDICARE_NO",
"PHY_NAME", "UPIN", "DIAG_CODE1", "DIAG_CODE2", "DIAG_CODE3",
"DIAG_CODE4", "DIAG_CODE5", "DIAG_CODE6", "DIAG_CODE7", "DIAG_CODE8",
"DIAG_CODE9", "DIAG_CODE10", "LOCAL_PROFILE_CODE",
"STANDARD_PROFILE_CODE", "PROFILE_NAME", "LOCAL_ORDER_CODE",
"STANDARD_ORDER_CODE", "ORDER_NAME", "LOINC_CODE",
"LOCAL_RESULT_CODE", "RESULT_NAME", "RESULT_VALUE_A", "UNITS",
"REF_RANGE_LOW", "REF_RANGE_HIGH", "REF_RANGE_ALPHA",
"DERIVED_ABNORMAL_FLAG", "CPT_CODE", "COMM_TEXT",
"ORDERING_SITE_CODE", "Elig_Member_Id", "npi", "unique_linker", "DM", "HTN",
"DM_HTN"]
```

```
# Function to standardize columns
def standardize_columns(df, correct_columns):
    # Rename columns to match the correct ones
    df.columns = [col.strip() for col in df.columns]
 # Reindex DataFrame to have missing columns filled with NaN and extra columns dropped
    return df.reindex(columns=correct_columns)

# Read, standardize, and concatenate all files
dfs = []
for file in datafiles:
    df = pd.read_csv(file, low_memory=False)
    df = standardize_columns(df, correct_columns)
    dfs.append(df)

# Concatenate all Data Frames
final_df = pd.concat(dfs, ignore_index=True)
final_df

#Keeping only the required column and setting the columns as copy.
df1 = final_df[['mbi_id_orig', 'DOS','RESULT_NAME', 'PAT_STATE', 'PAT_ZIP', 'AGE',
'RESULT_VALUE_A','DM', 'HTN', 'DM_HTN']].copy()
#Saving the dataframe to a csv file.
df1.to_csv('GeoSpatial_Map_Data.csv', index=False)

#Importing the required libraries
```

```
import pandas as pd
import numpy as np
#Reading the GeoSpatial_Map_Data dile as a dataframe
new_df = pd.read_csv("GeoSpatial_Map_Data.csv")
new_df
```

#Changing the date format

```
new_df.loc[:, 'DOS'] = pd.to_datetime(new_df['DOS'], format='%Y%m%d')
```

#Replacing all the blank rows of DM_HTN with 0

```
new_df['DM_HTN'] = new_df['DM_HTN'].fillna(0)
new_df
```

 # Filter for Hemoglobin A1c tests

```
df_hba1c = new_df[new_df['RESULT_NAME'] == 'HEMOGLOBIN A1c']
```

 # Filter for patients with Diabetes (DM = 1)

```
df_hba1c = df_hba1c[df_hba1c['DM'] == 1]
df_hba1c
```

#Sorting the data based on DOS and dropping the duplicate mbi_id_orig to keep the unique patients.

```
df_hba1c = df_hba1c.sort_values('DOS').drop_duplicates('mbi_id_orig', keep = 'last')
df_hba1c
```

 # Ensure Test_result_value is numeric and remove non-numeric entries

```
df_hba1c['RESULT_VALUE_A'] = pd.to_numeric(df_hba1c['RESULT_VALUE_A'], errors='coerce')

df_hba1c = df_hba1c.dropna(subset=['RESULT_VALUE_A'])

df_hba1c
```

#dropping null values from PAT_STATE and PAT_ZIP column.

```
df_hba1c = df_hba1c.dropna(subset=['PAT_STATE'])

df_hba1c = df_hba1c.dropna(subset=['PAT_ZIP'])

df_hba1c
```

#Coverting the PAT_ZIP into string to remove the Zip codes, which are greater than 5 digits

```
df_hba1c['PAT_ZIP'] = df_hba1c['PAT_ZIP'].astype(str)


df_hba1c['PAT_ZIP'] = df_hba1c['PAT_ZIP'].apply(lambda x: x[:5] if pd.notnull(x) else x)
```

 # Convert zip_code to numeric, setting errors='coerce' to convert non-numeric values to NaN

```
df_hba1c['PAT_ZIP'] = pd.to_numeric(df_hba1c['PAT_ZIP'], errors='coerce')
```

 # Finally, convert to integer

```
df_hba1c['PAT_ZIP'] = df_hba1c['PAT_ZIP'].astype(int)
```

#The final dataframe that contains the data to create the Geo Spatial map

```
df_hba1c
```

#The final Geo Spatial data in csv

```
df_hba1c.to_csv ('Final_GeoSpatial_Map_Data.csv', index=False)
```

Validation Check

 # Group by 'PAT_STATE' and 'PAT_ZIP' and calculate the average of 'RESULT_VALUE_A' and 'AGE'

 # Also, count the 'mbi_id_org'

```
average_values = df_hba1c.groupby(['PAT_STATE', 'PAT_ZIP']).agg({
    'RESULT_VALUE_A': 'mean',
    'AGE': 'mean',
    'mbi_id_orig': 'count'
}).reset_index()
```

 # Define the function to categorize based on the average Result_Value_A

```
def categorize_a1c(avg_value):
    if 5.47 <= avg_value <= 5.69:
        return 'Normal'
    elif 5.70 <= avg_value <= 6.00:
        return 'Pre-Diabetes'
    elif 6.01 <= avg_value <= 6.25:
        return 'Pre-Diabetes'
    elif 6.26 <= avg_value <= 6.49:
        return 'Pre-Diabetes'
    elif 6.50 <= avg_value <= 7.21:
```

```
        return 'Diabetes'
    elif 7.22 <= avg_value <= 8.99:
        return 'Diabetes High'
    elif avg_value >= 9.0:
        return 'Diabetes Poor Control'
    else:
        return 'Unknown'
```

 # Apply the function to the average values

```
average_values['A1c_Category'] =
average_values['RESULT_VALUE_A'].apply(categorize_a1c)
```

 # Print the resulting DataFrame

```
print(average_values)
```

 # Filter the DataFrame for 'MD' state

```
md_values = average_values[average_values['PAT_STATE'] == 'MD']
md_values
```

 # Group by 'PAT_STATE' and calculate the average of 'RESULT_VALUE_A' and 'AGE'

```
average_values_by_state = df_hba1c.groupby('PAT_STATE').agg({
    'RESULT_VALUE_A': 'mean',
    'AGE': 'mean'
}).reset_index()
```

#Checking the data frame

```
average_values_by_state
```

3) **Manage, organize, and summarize these files to:**

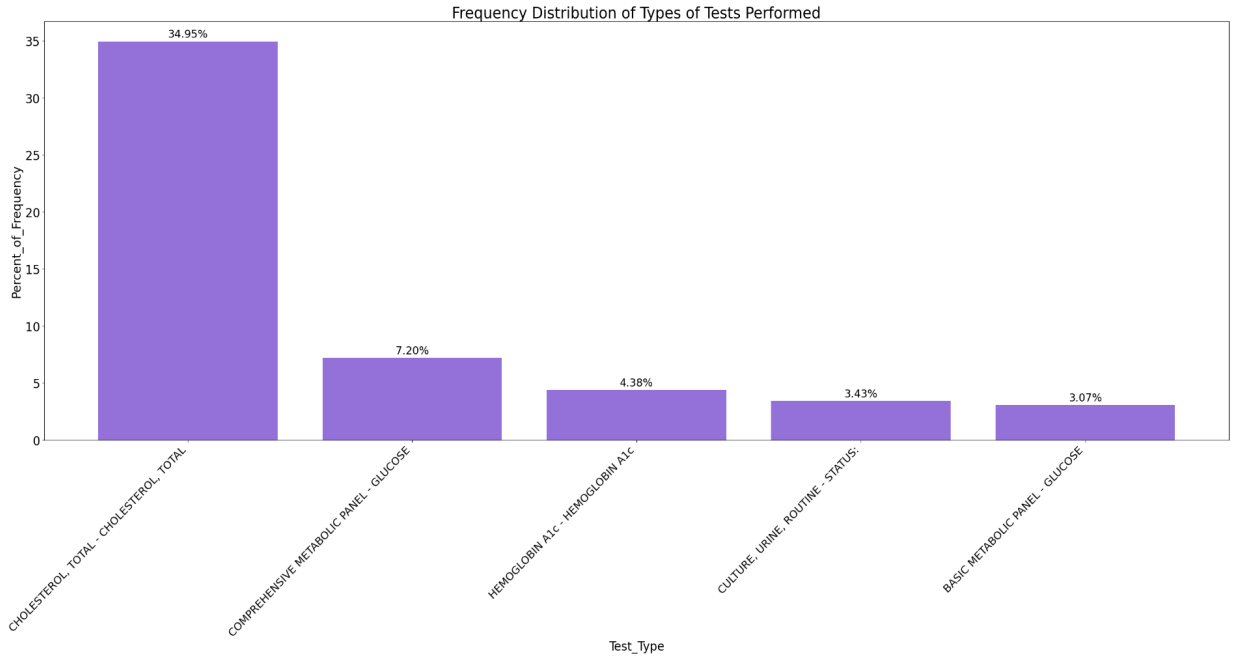
   a. **For the unique patients with "DM=1" [diabetes] and/or HTN=1 [hypertension] create an overall summary of the frequency of the types of tests performed.**

#Denominator (unique_patients_df) = 54775
#Year: 2022-2024

	Test_Type	Frequency	Percent_of_Frequency
0	CHOLESTEROL, TOTAL - CHOLESTEROL, TOTAL	19133	34.95
1	COMPREHENSIVE METABOLIC PANEL - GLUCOSE	3944	7.20
2	HEMOGLOBIN A1c - HEMOGLOBIN A1c	2399	4.38
3	CULTURE, URINE, ROUTINE - STATUS:	1876	3.43
4	BASIC METABOLIC PANEL - GLUCOSE	1683	3.07
...	...	...	...
1627	11-DEOXYCORTISOL - 11-DEOXYCORTISOL	1	0.00
1628	NICOTINE AND COTININE, SERUM/PLASMA - NICOTINE...	1	0.00
1629	HMGCR AB (IGG) - HMGCR AB (IGG)	1	0.00
1630	GLUCOSE, RANDOM - GLUCOSE, RANDOM	1	0.00
1631	Cytomegalovirus Antibodies (IgG,IgM) - Cytomeg...	1	0.00

1632 rows × 3 columns

This is the summary result for unique patients, where DM= 1 and/or HTN = 1 with the types of tests performed.



CHOLESTEROL is the most frequent test out of all tests with a frequency of 34.95 %, which is the major portion out of all the tests followed by COMPREHENSIVE METABOLIC PANEL and HEMOGLOBIN A1c. with significantly small amount of 7.20% and 4.38% respectively.

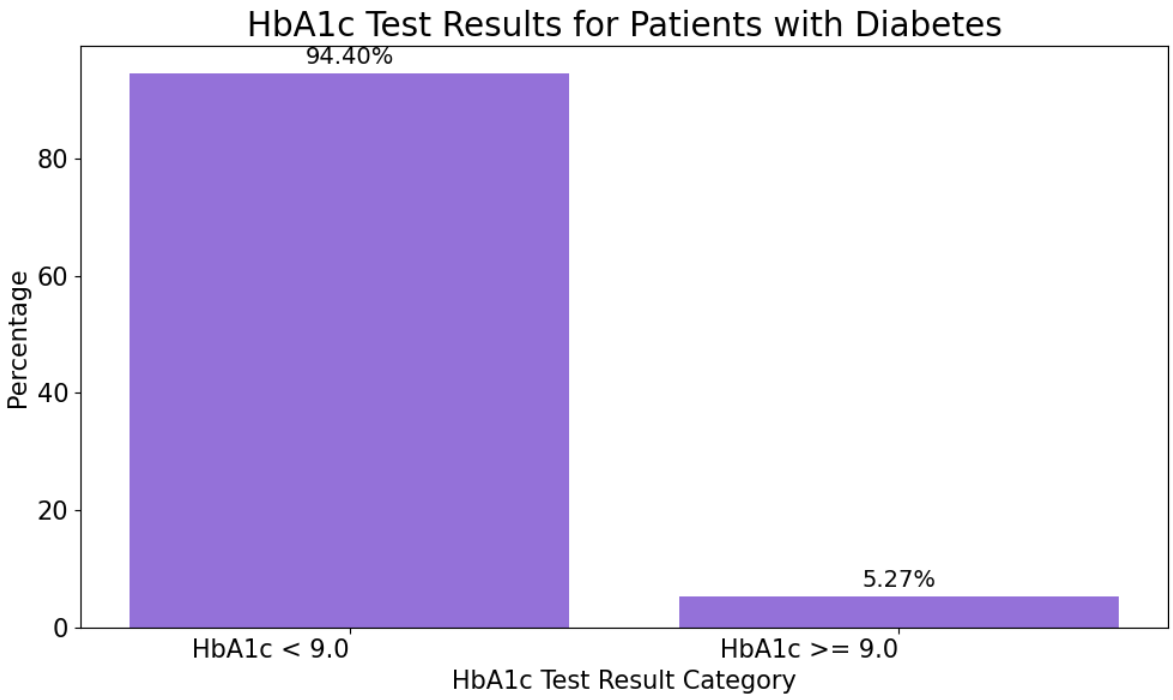
b. For the patients with a DM=1; how many received HbA1c Tests that had a value below 9.0, at or above 9.0, or result was not presented

```
# Denominator (hba1c_tests_df) = 38815
# Year: 2022-2024
```

Patients with HbA1c < 9.0: 94.4

Patients with HbA1c >= 9.0: 5.27

These are the results for all the patients with DM=1, who received the HbA1c test with value below 9.0, at or above 9.0, or result was not presented.



Majority of the patients with diabetes fall under the category of HbA1c < 9.0 with a frequency of 94.40%, which indicates well-controlled diabetes.

Approximately 5.27% of patients fall into the category of HbA1c >= 9.0, that suggests poorer glucose control.

Implications:

- Patients with HbA1c < 9.0 have better glucose management. Lower risk of diabetes-related complications.
- Patients with HbA1c >= 9.0 may face increased risks. Complications include neuropathy, kidney disease, and cardiovascular issues.

c. For patients with a DM=1, HTN=1, what were the common tests that were performed? Frequency distribution from highest volume to lowest volume.

```
# Denominator (filtered_df) = 701864
# Year: 2022-2024
```

	Test_Type	Common_Test_Frequency	Percent_of_Frequency
0	COMPREHENSIVE METABOLIC PANEL - GLUCOSE	11159	1.59
5	COMPREHENSIVE METABOLIC PANEL - SODIUM	11143	1.59
8	COMPREHENSIVE METABOLIC PANEL - ALBUMIN	11143	1.59
7	COMPREHENSIVE METABOLIC PANEL - ALKALINE PHOSP...	11143	1.59
6	COMPREHENSIVE METABOLIC PANEL - CREATININE	11143	1.59
...	...	...	...
5060	Chromosome Analysis with/without reflex to FIS...	1	0.00
5059	Chromosome Analysis with/without reflex to FIS...	1	0.00
5058	Chromosome Analysis with/without reflex to FIS...	1	0.00
5057	Chromosome Analysis with/without reflex to FIS...	1	0.00
6157	ALCOHOL,ETHYL,QL,W/CONF, URINE - ALCOHOL, ETHY...	1	0.00

6158 rows × 3 columns

This is the summary result for all the patients with diabetes and hypertension with the types of tests performed with the frequency distribution of all types of tested performed from highest to lowest volume.

The most frequent test for patients with both diabetes and hypertension is COMPREHENSIVE METABOLIC PANEL – GLUCOSE with a frequency of 11159 (1.59%) followed by other CMP tests like SODIUM, ALBUMIN etc.

Additional questions: -

1) **Question: What are the most frequent laboratory tests that are ordered for patients with Diabetes?**
    a. **Filter to DM=1;**
    b. **Create a value of Order Name – Result Name**
    c. **Count distinct on Date of Collection**
    d. **Order from Highest volume of Orders – Results, to lowest Number of Orders – Results**

#Denominator (order_results_counts_sorted['DistinctCounts'].sum()) = 140330

Year: 2022-2024

	Test_Type	DistinctCounts	Percent_of_Frequency
810	CBC (INCLUDES DIFF/PLT) - ABSOLUTE EOSINOPHILS	553	0.39
808	CBC (INCLUDES DIFF/PLT) - ABSOLUTE BASOPHILS	553	0.39
856	CBC (INCLUDES DIFF/PLT) - RED BLOOD CELL COUNT	553	0.39
854	CBC (INCLUDES DIFF/PLT) - RDW	553	0.39
848	CBC (INCLUDES DIFF/PLT) - NEUTROPHILS	553	0.39
...	...	...	...
1781	DRUG MONITOR, TAPENTADOL, QN, URINE - Tapentadol	0	0.00
1820	DRUG TOX COCAINE, W/CONF, ORAL FLUID - Cocaine	0	0.00
1819	DRUG TOX COCAINE, W/CONF, ORAL FLUID - Benzoyl...	0	0.00
1811	DRUG TOX AMPHETAMINES, W/CONF, ORAL FLUID - Me...	0	0.00
2607	LAMB (F88) IGE - LAMB (F88) IGE	0	0.00

4292 rows × 3 columns

Test Type	Distinct Counts
CBC	553
CMP	548
HEMOGLOBIN A1c	546
NON HDL CHOLESTEROL	533
TSH	517

The most frequent test for patients with diabetes are CBC (INCLUDES DIFF/PLT) - ABSOLUTE EOSINOPHILS with a frequency of 553 (0.39 %) followed by CMP, HEMOGLOBIN A1c, NON-HDL CHOLESTEROL, TSH etc.

Implications:

- High HbA1c indicates poor blood sugar control, increasing the risk of complications like neuropathy, kidney disease, and cardiovascular issues.
- Abnormal results of CMP may indicate organ dysfunction or metabolic disorders.
- Higher HDL levels are desirable, as they reduce cardiovascular risk.
- Abnormalities in CBC can signal anemia, infection, or blood disorders.
- Elevated TSH suggests hypothyroidism, while low levels may indicate hyperthyroidism.

2) **Question: What are the most frequent laboratory tests that are ordered for patients with Hypertension?**
   a. **Filter to HTN=1;**
   b. **Create a value of Order Name – Result Name**
   c. **Count distinct on Date of Collection**
   d. **Order from Highest volume of Orders – Results, to lowest Number of Orders – Results**

#Denominator (order_results_counts_sorted['DistinctCounts'].sum()) = 100725

#Year: 2022=2024

	Order_Result	DistinctCounts	Percent_of_DistinctCounts
1929	HEMOGLOBIN A1c - HEMOGLOBIN A1c	536	0.53
1030	COMPREHENSIVE METABOLIC PANEL - BILIRUBIN, TOTAL	531	0.53
1047	COMPREHENSIVE METABOLIC PANEL - SODIUM	531	0.53
1036	COMPREHENSIVE METABOLIC PANEL - CARBON DIOXIDE	531	0.53
1035	COMPREHENSIVE METABOLIC PANEL - CALCIUM	531	0.53
...	...	...	...
2251	LAMB (F88) IGE - LAMB (F88) IGE	0	0.00
1917	HEAVY METALS 24 HOUR URINE WITH CADMIUM - ARSE...	0	0.00
1847	GALACTOSE ALPHA 1,3 GALACTOSE IGE - GALACTOSE ...	0	0.00
1580	DRUG TOX MONITORING 5 W/CONF, URINE - Buprenor...	0	0.00
1600	DRUG TOX MONITORING 9 W/CONF, URINE - Norbupre...	0	0.00

3735 rows × 3 columns

Test Type	Distinct Counts
HEMOGLOBIN A1c	536
CMP	531
CBC	525
LDL-CHOLESTEROL	524
TSH	509

The most frequent test for patients with hypertension is HEMOGLOBIN A1c with a frequency of 536 (0.53%) followed by CMP, CBC, LDL-CHOLESTEROL, TSH etc.

3) **Question: What are the most frequent laboratory tests that are ordered for patients with both Hypertension and Diabetes?**
   a. **Filter to HTN=1 and DM=1;**
   b. **Create a value of Order Name – Result Name**
   c. **Count distinct on Date of Collection**
   d. **Order from Highest volume of Orders – Results, to lowest Number of Orders – Results**

```
#Denominator (order_results_counts_sorted['DistinctCounts'].sum()) = 73239

# Year: 2022-2024
```

	Order_Result	DistinctCounts	Percent_of_DistinctCounts
1400	HEMOGLOBIN A1c - HEMOGLOBIN A1c	493	0.67
763	COMPREHENSIVE METABOLIC PANEL - SODIUM	484	0.66
760	COMPREHENSIVE METABOLIC PANEL - POTASSIUM	484	0.66
754	COMPREHENSIVE METABOLIC PANEL - CREATININE	484	0.66
753	COMPREHENSIVE METABOLIC PANEL - CHLORIDE	484	0.66
...	...	...	...
1149	DRUG MONITOR, TAPENTADOL, QN, URINE - Tapentadol	0	0.00
2024	PORK (F26) IGE - CLASS	0	0.00
1196	DRUG TOX MONITORING 9 W/CONF, URINE - Aminoclo...	0	0.00
1666	LAMB (F88) IGE - LAMB (F88) IGE	0	0.00
1334	GALACTOSE ALPHA 1,3 GALACTOSE IGE - GALACTOSE ...	0	0.00

2765 rows × 3 columns

Test Type	Distinct Counts
HEMOGLOBIN A1c	493
CMP	484
HDL CHOLESTEROL	479
CBC	477
TSH	462

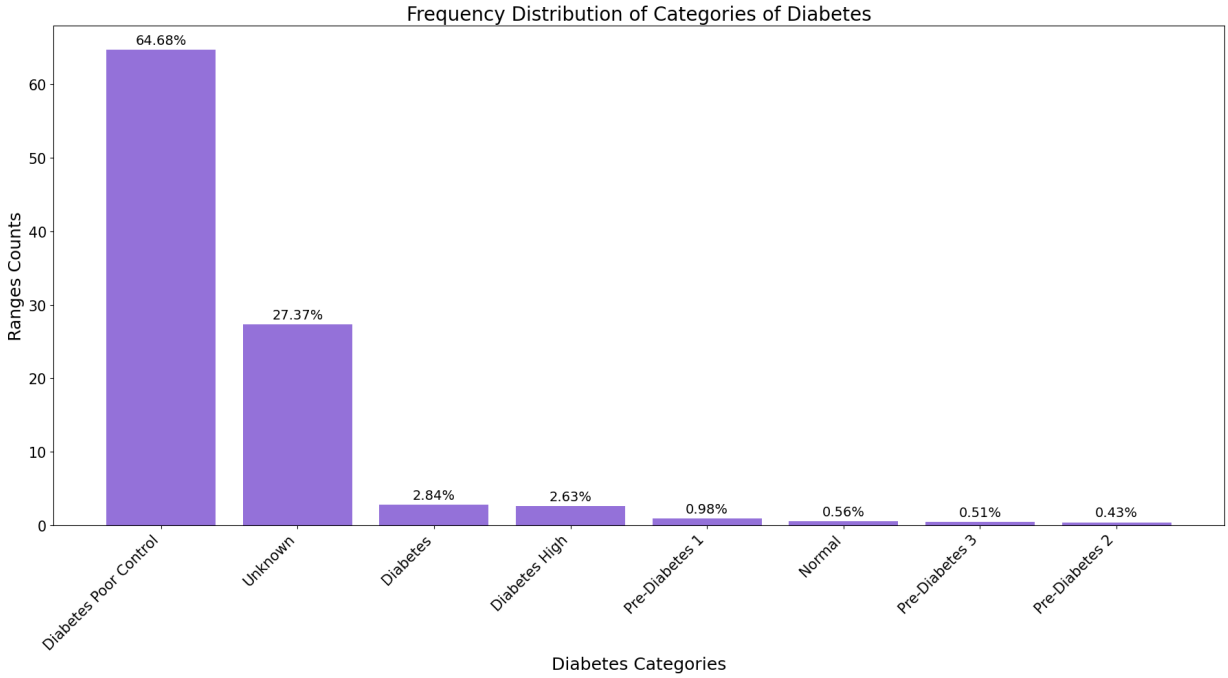
The most frequent laboratory tests that are ordered for patients with diabetes and hypertension are HEMOGLOBIN A1c with a frequency of 493 (0.67%) followed by CMP, HDL-CHOLESTEROL, CBC, TSH etc.

4) **For the standardized test values – let's start small here and this will also allow you to create a cleansed/standardized data set to do the HbA1c map and Kidney Health Heat Map:**
    a. **For patients with Diabetes, how may had lab test values in the following ranges:**
        i. **5.47 – 5.69 (Normal)**
        ii. **5.70 – 6.00 (Pre-Diabetes)**
        iii. **6.01 – 6.25 (Pre-Diabetes)**
        iv. **6.26 – 6.49 (Pre-Diabetes)**
        v. **6.50 – 7.21 (Diabetes)**
        vi. **7.22 – 8.99 (Diabetes High)**
        vii. **>9.0 Diabetes Poor Control**
        viii. **Unknown (This will be all the results that you will not be able to convert ("standardize") to one of the categories above.**

Denominator (category_counts['Count'].sum()) = 2067878

Year: 2022-2024

	Category	Count	Percent_of_Frequency
0	Diabetes Poor Control	1337473	64.68
1	Unknown	565903	27.37
2	Diabetes	58822	2.84
3	Diabetes High	54374	2.63
4	Pre-Diabetes 1	20195	0.98
5	Normal	11593	0.56
6	Pre-Diabetes 3	10565	0.51
7	Pre-Diabetes 2	8953	0.43



With 64.68% (1337473) of cases falling under "Diabetes Poor Control," it is evident that a significant majority of individuals with diabetes are not managing their condition effectively. This suggests widespread issues in diabetes care, such as inadequate access to healthcare, poor patient adherence to treatment plans, or insufficient patient education on diabetes management.

The "Unknown" category accounts for 27.37% (565903) of the cases. This substantial portion indicates a gap in data collection or classification, which can hinder effective disease management and policymaking.

Categories like "Diabetes," "Diabetes High," and various "Pre-Diabetes" stages have very low percentages, each below 3%. The "Normal" category is also minimal.

Implications:

The graph indicates a dire need for improvements in diabetes management and monitoring. Addressing the high rate of poor diabetes control and significant unknown cases should be a priority. This requires a combination of better healthcare practices, patient education, and systemic changes to ensure individuals receive the support and resources they need to manage their diabetes effectively.

b. For Patients with Hypertension and/or Diabetes (this will be your entire cohort), how many had lab values in the following ranges:

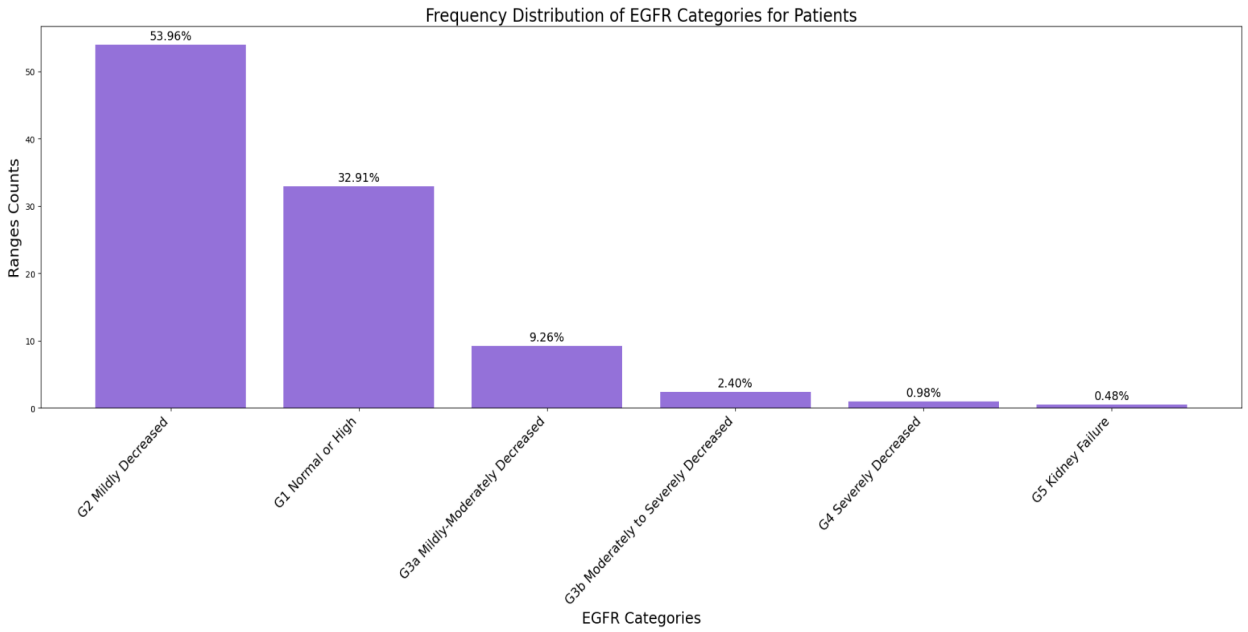
i. GFR Categories

1. **G1 Normal or High >=90**
2. **G2 Mildly Decreased 60-89**
3. **G3a Mildly-Moderately Decreased 45-59**
4. **G3b Moderately to severely decreased 30-44**
5. **G4 Severely decreased 15-29**
6. **G5 Kidney Failure <15**
7. **Unknown (This will be all the results that you will not be able to convert ("standardize") to one of the categories above.**

#Denominator (egfr_category_counts['Count'].sum()) = 33025

#Year: 2022-2024

	EGFR_Category	Count	Percent_of_Frequency
0	G2 Mildly Decreased	17821	53.96
1	G1 Normal or High	10870	32.91
2	G3a Mildly-Moderately Decreased	3058	9.26
3	G3b Moderately to Severely Decreased	791	2.40
4	G4 Severely Decreased	325	0.98
5	G5 Kidney Failure	160	0.48



The majority of the patients 17821 (53.96%) fall into the G2 Mildly Decreased category with GFR value between 60-89. This indicates that more than half of the patients have mildly decreased kidney function, which may require monitoring and potential lifestyle modifications to prevent further decline.

A significant portion of the patients 10870 (32.91%) have normal or high kidney function under category G1 with GFR value greater than or equal to 90. This suggests that nearly a third of the patients have healthy kidneys or are functioning well above the threshold for concern.

A smaller group of patients 3058 (9.26%) fall into the G3a category with GFR value between 45-59. These patients have mildly to moderately decreased kidney function, indicating a higher risk of progression to more severe stages of kidney disease.

An even smaller group of 791 (2.40%) is in the G3b Moderately to Severely Decreased category. This group has moderately to severely decreased kidney function, and they require closer medical attention to manage their condition and prevent further decline.

A very small percentage of patients 325 (0.98%) are in the G4 category. These patients have severely decreased kidney function and are at significant risk for complications. They need intensive medical management.

The smallest group 160 (0.48%) falls into the G5 category. These patients have kidney failure and likely require dialysis or a kidney transplant. They represent the most critical cases in this dataset.

Implications:

- Healthcare providers should focus on maintaining kidney health in the G1 and G2 groups through regular monitoring and preventive measures.
- Patients in the G3a and G3b categories require closer observation and potentially more aggressive management to slow disease progression.
- Those in the G4 and G5 categories need intensive medical care and possibly interventions such as dialysis or transplantation.

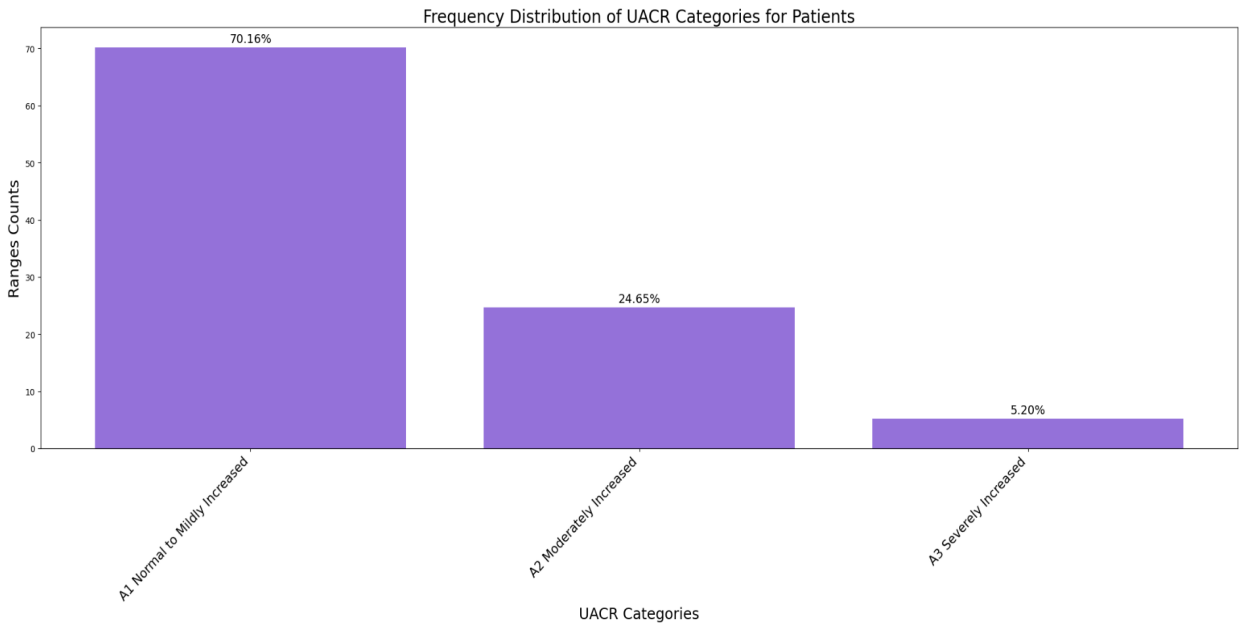
ii. Albuminuria Categories

1. **A1 Normal to Mildly Increased <30 mg/g < 3 mg/mmol**
2. **A2 Moderately Increased 30-299 mg/g 3-29 mg/mmol**
3. **A3 Severely Increased ≥300 mg/g ≥30 mg/mmol**
4. **Unknown (This will be all the results that you will not be able to convert ("standardize") to one of the categories above.**

#Denominator (uacr_category_counts['Count'].sum()) = 33025

Year: 2022-2024

	UACR_Category	Count	Percent_of_Frequency
0	A1 Normal to Mildly Increased	23169	70.16
1	A2 Moderately Increased	8140	24.65
2	A3 Severely Increased	1716	5.20



There are 23169 patients, which is about 70.16% of all the patients with Diabetes and/or Hypertension, who have the Albuminuria value between <30 mg/g < 3 mg/mmol and fall under the category of A1 Normal to Mildly Increased, this indicates that most patients have normal kidney function or only mild increases in albumin levels, suggesting low levels of kidney damage or risk.

Similarly, a smaller but still considerable portion of the patients 8140 (24.65%) are in this category of A2 Moderately Increased with the Albuminuria value between 30-299 mg/g 3-29 mg/mmol, indicating a higher risk of kidney damage compared to the A1 category. This group may require closer monitoring and potentially more aggressive management to prevent further kidney damage.

A minority of the patients 1716 (5.20%) fall under the most severe category A3 Severely Increased with Albuminuria value range ≥300 mg/g ≥30 mg/mmol. These patients have high levels of albumin in their urine, which is a strong indicator of significant kidney damage and a higher risk of progressing to chronic kidney disease or end-stage renal disease. These patients likely need intensive medical management and possibly interventions to slow or prevent further kidney damage.

Implications:

- The high percentage of patients in the A1 category suggests that most of the patient population has either healthy kidneys or is at an early stage of kidney damage, which is more easily manageable.
- The healthcare providers should focus on monitoring patients in the A2 category closely to prevent progression to more severe stages.

- Immediate and possibly intensive interventions may be necessary for those in the A3 category to manage advanced kidney disease and prevent further deterioration.

5) What gaps are you seeing in the data?

1) Null Values - The dataset contained many null values. For example, in the columns RESULT_VALUE_A, we removed many null values to ensure that the result values were not null. For the Geospatial map, we also removed the null values from PAT_ZIP and PAT_STATE. Moreover, DM_HTN had null values, so we filled it by 0 since the rows without DM = 1 and HTN=1 together would be 0.
2) Non-Numeric - The dataset also contained non-numeric values, particularly in the RESULT_VALUE_A column. These non-numeric values posed a significant challenge in comparing the result values to obtain the desired data. To address this, we removed all non-numeric values from the dataset, ensuring the accuracy of our analysis.
3) Datatype - Data quality is paramount in the data analysis process. This is why we paid close attention to the datatype of certain fields in the dataset. For instance, RESULT_VALUE_A initially contained alpha-numeric values, but we required only numeric values. Therefore, we converted its datatype from object to numeric. Similarly, we converted the datatype of DOS from integer to datetime, ensuring the quality of our data for analysis.
4) Duplicate values - There were duplicate values in the dataset. For example, mbi_id_orig had many duplicate values. However, we need the results for the unique patients, so we removed the duplicate values.

Other Gaps: -

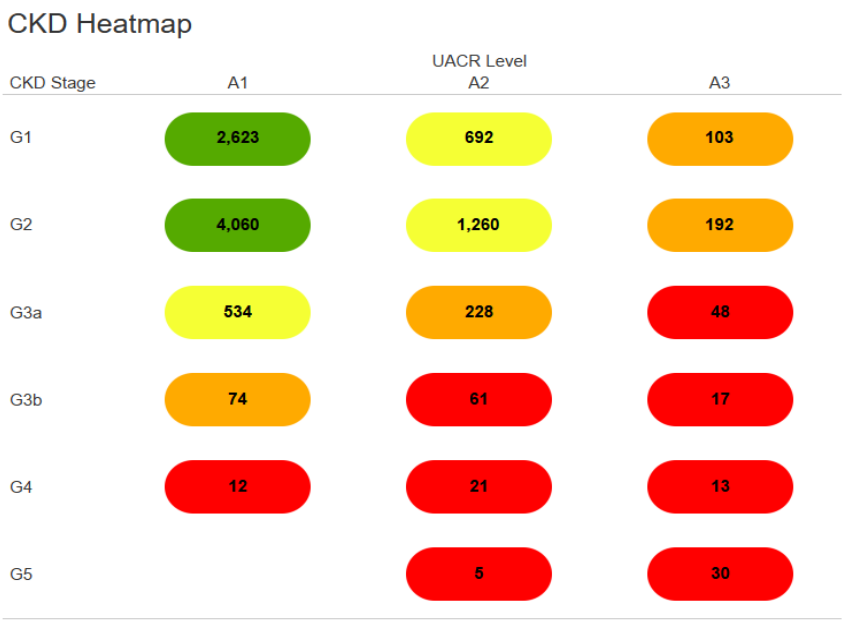
1) Working on the CKD Analysis, we found many discrepancies with the CPT_CODE values. Since LOINC_CODE is more specific to the type of test that has been performed, we used LOINC_CODE to filter the data to find EGFR and UACR result values.

- There is more than one LOINC_CODE associated with a specific CPT_CODE that doesn't even fall under the EGFR and UACR value range, which means it has abnormal values like 0.2, etc., whereas the range of values for EGFR goes from less than 15 to 90 or above, and for UACR, it goes from less than 30 to 300.
- Also, there are specific LOINC_CODE, which are EGFR and UACR, but there is no associated CPT_CODE value.

2) The ORDER_NAME column contains the same name for the different tests, which makes it confusing to decide which ORDER_NAME the program should choose. Example – For EGFR, the LOINC_CODE is '98979-8', the ORDER_NAME for this is 'Comprehensive Metabolic Panel (CMP),' and somewhere it is 'CREATININE' and 'Basic Metabolic Panel.' However, The CMP order name was also presented in the other LOINC_CODES, which were not associated with EGFR and UACR values. Hence, we did not use the ORDER_NAME column in our analysis.

6) **A bonus challenge will be to wrangle, organize and summarize the data into the CKD "Heat Map" that the NKF recommends – here is one link for background and more information:**
| National Kidney Foundation https://www.kidney.org/news/nkf-launches-new-kidney-disease-public-education-series
heat_map_card.pdf (kidney.org)
| National Kidney Foundation https://www.kidney.org/atoz/content/understanding-your-lab-values

CKD Heatmap



eGFR Table

CKD Stage	eGFR Number
G1	90 or Higher
G2	60-89
G3a	45-59
G3b	30-44
G4	15-29
G5	15 or Lower

uACR Table

uACR Number	uACR Level Text
30-300	A2, moderately increased
Higher than 300	A3, severely increased
Lower than 30	A1, normal – mildly increased

On the left side of the map, your eGFR number matches up with a CKD stage. A higher eGFR number is better because it means you have a lower CKD stage.

On the top of the map, your uACR number matches up with a uACR level. A lower uACR is better because that means less albumin in the urine.

A **Green box** means you do NOT have chronic kidney disease, or that you are at the lowest risk for CKD getting worse. **Yellow** means increased risk for CKD getting worse. **Orange** means high risk for CKD getting worse. **Red** means the highest risk for CKD getting worse.

Quest Date Range: **October 2022 - May 2024**

CKD Stage	UACR Level	No of Patients	eGFR Number	UACR Number
G1	A1	2623	90 or Higher	Lower than 30
G2	A1	4060	60-89	Lower than 30
G3a	A1	534	45-59	Lower than 30
G3b	A1	74	30-44	Lower than 30
G4	A1	12	15-29	Lower than 30
G1	A2	692	90 or Higher	30-300
G2	A2	1260	60-89	30-300
G3a	A2	228	45-59	30-300
G3b	A2	61	30-44	30-300
G4	A2	21	15-29	30-300
G5	A2	5	15 or Lower	Higher than 300
G1	A3	103	90 or Higher	Higher than 300

G2	A3	192	60-89	Higher than 300
G3a	A3	48	45-59	Higher than 300
G3b	A3	17	30-44	Higher than 300
G4	A3	13	15-29	Higher than 300
G5	A3	30	15 or Lower	Higher than 300

This map is the Chronic Kidney Disease Heatmap between the CKD Stage and UACR Level.
We separated the EGFR and UACR values using LOINC_CODE '98979-8' and '9318-7' respectively. And then used the condition to create the separate CKD Stage using conditions for EGFR values as follows:
90 or Higher, then G1
60-89, then G2
45-59, then G3a
30-44, then G3b
15-29, then G4
15 or lower than G5

Similarly, for UACR values, the condition is as follows:
Lower than 30, then A1
30-300, then A2
Higher than 300 than A3

Based on this, we created the CKD Heatmap.

The CKD (chronic kidney disease) heatmap visualizes the risk levels associated with different stages of CKD based on two parameters: eGFR (estimated Glomerular Filtration Rate) and uACR (urine Albumin-to-Creatinine Ratio).

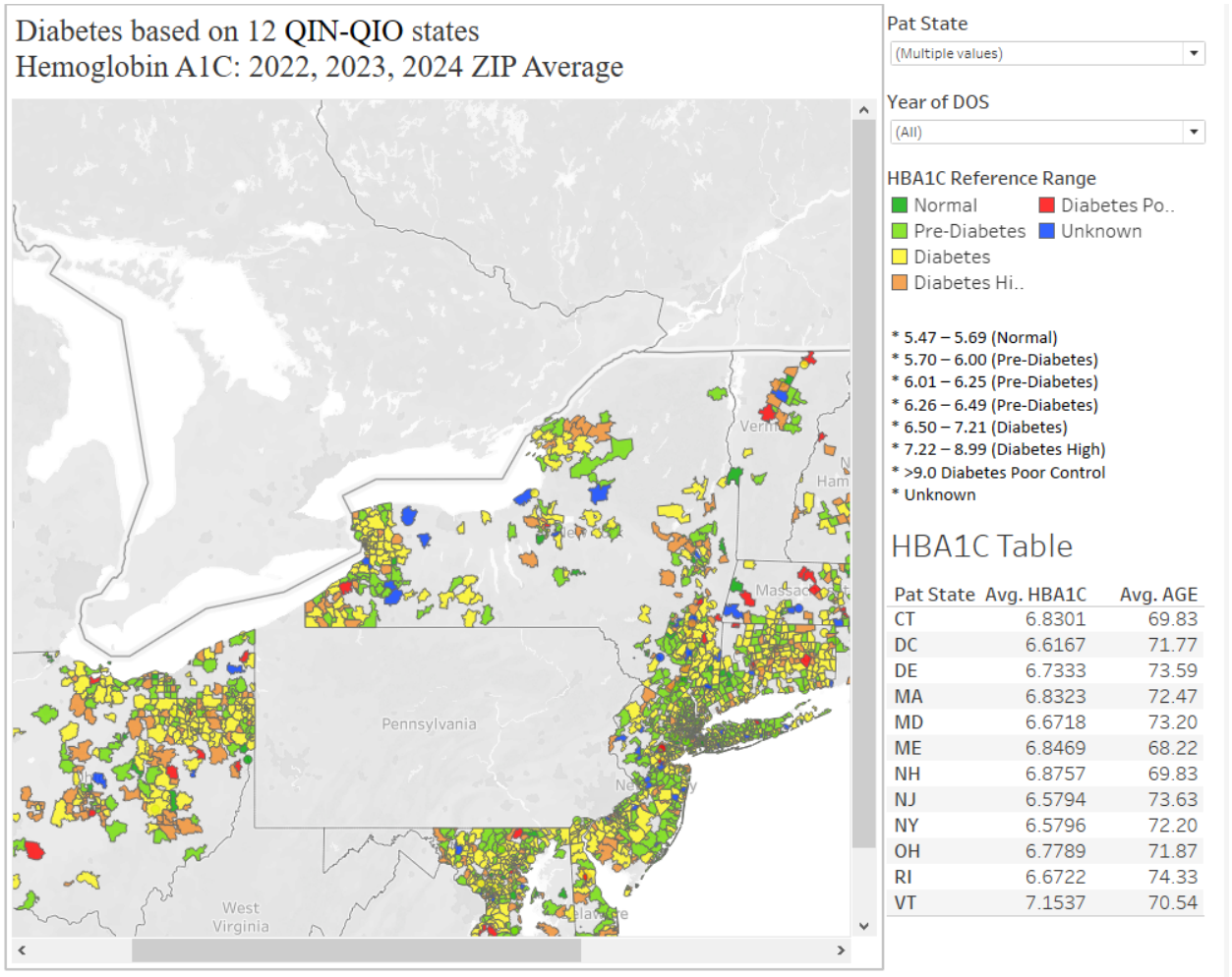
- The left side of the map indicates the CKD stage based on eGFR levels, with higher eGFR indicating better kidney function.
- The top of the map indicates the uACR level, with lower uACR indicating less albumin in the urine, which is better.
- The color coding indicates the risk of CKD progression, with green being the lowest risk and red being the highest risk.

The result shows 2623 patients with CKD Stage G1 and UACR Level A1. Similarly, there are 4060, 534, 74, 12, 692, 1260, 228, 61, 21, 5, 103, 192, 48, 17, 13, and 30 patients with CKD Stage and UACR Level G2 A1, G3a A1, G3a A1, G3b A1, G4 A1, G1

A2, G2 A2, G3a A2, G3b A2, G4 A2, G5 A2, G1 A3, G2 A3, G3a A3, G3b A3, G4 A3, G5 A3 respectively.

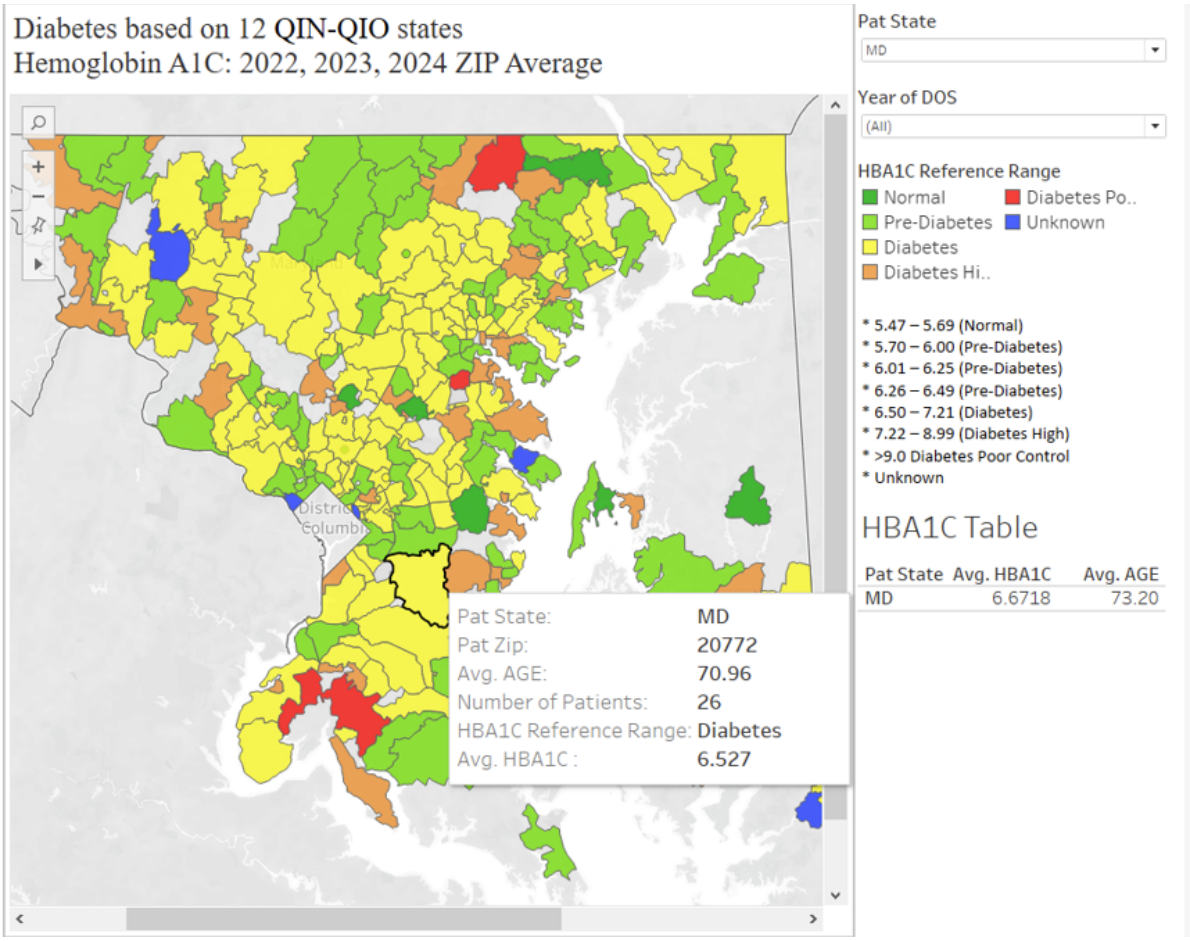
Geo-Spatial Map:

The overall geo-spatial map of all the 12 states, which fall under the QIN-QIO program contract.



Pat State	Avg. HBA1C	Avg. AGE
CT	6.8301	69.83
DC	6.6167	71.77
DE	6.7333	73.59
MA	6.8323	72.47
MD	6.6718	73.20
ME	6.8469	68.22
NH	6.8757	69.83
NJ	6.5794	73.63
NY	6.5796	72.20
OH	6.7789	71.87
RI	6.6722	74.33
VT	7.1537	70.54

Geo-Spatial Map of Maryland:



With CMS, we work with 12 states under the QIN-QIO program contract. Therefore, we created a Geo-Spatial Map for the states (CT, DC, DE, MA, MD, ME, NH, NJ, NY, OH, RI, VT) to show the Diabetes Hemoglobin A1c results based on zip codes.

The green in the map shows the Normal range of HbA1c, with values ranging from 5.47 to 5.69. Light green indicates Pre-Diabetes (5.70-6.49), yellow indicates Diabetes (6.50-7.21), Orange indicates Diabetes High (7.22-8.99), Red indicates Diabetes Poor Control (>9.0), and blue indicates Unknown.

The average HbA1c of CT is 6.8301. Similarly, the average HbA1c for DC, DE, MA, MD, ME, NH, NJ, NY, OH, RI, VT are 6.6167, 6.7333, 6.8323, 6.6718, 6.8469, 6.8757, 6.5794, 6.5796, 6.7789, 6.6722 and 7.1537 respectively.

A more precise map of one of the state's MDs for the Pat Zip ‘20772’ shows that there are 26 patients with Diabetes, with an average age of 70.96 and an average HbA1c of 6.527.