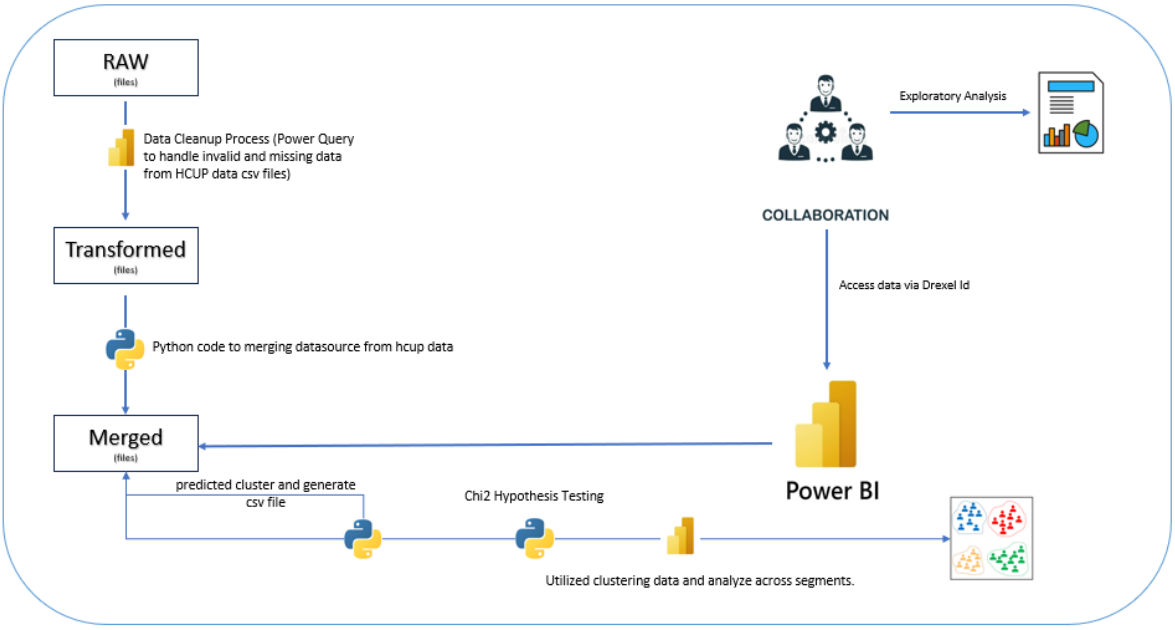
Prevalence of Type-1 Diabetes Diagnoses Based on Demographic Disparities

Drexel University LeBow College of Business
Fall Term 2023
Team 4

User Manual

Table of Contents

Data Flow Overview



- The data processing begins with raw files, which undergo cleanup in Power Query to rectify missing or invalid entries.
- Post-cleansing, Python scripts to merge the Health Care & Utilization Project (HCUP) datasources, creating a consolidated "Transformed" dataset.
- This data is then analyzed to predict clusters and is saved as a CSV file for further cluster analysis.
- Concurrently, utilized the merged data output file in exploratory analysis offers insights into data patterns.
- Collaboration through Drexel ID access, and the final insights are visualized in Power BI, utilizing the clustered data for segment analysis patient demographic and type 1 diabetes diagnosis.

Data Cleanup and Transformation Process in Power Query M Language

```
let
    Source = Csv.Document(File.Contents("Dataset\Original\KID_2019_Core.csv"),[Delimiter=",", Columns=129, Encoding=1252, QuoteStyle=QuoteStyle.None]),
    #"Promoted Headers" = Table.PromoteHeaders(Source, [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"HOSP_KID", Int64.Type}, {"RECNUM", Int64.Type}, {"AGE", Int64.Type}, {"AGE_NEONATE", Int64.Type}, {"AMONTH", Int64.Type},
    {"AWEEKEND", Int64.Type}, {"DIED", Int64.Type}, {"DISCWT", type number}, {"DISPUNIFORM", Int64.Type}, {"DQTR", Int64.Type}, {"DRG", Int64.Type}, {"DRGVER", Int64.Type}, {"DRG_NoPOA", Int64.Type},
    {"ELECTIVE", Int64.Type}, {"FEMALE", type text}, {"HCUP_ED", Int64.Type}, {"HOSP_REGION", Int64.Type}, {"I10_BIRTH", Int64.Type}, {"I10_DELIVERY", Int64.Type}, {"I10_HOSPBRTH", Int64.Type}, {"I10_INJURY", Int64.Type}, {"I10_MULTINJURY", Int64.Type},
    {"I10_NDX", Int64.Type}, {"I10_NPR", Int64.Type},
    {"I10_SERVICELINE", Int64.Type}, {"I10_UNCBRTH", Int64.Type}, {"KID_STRATUM", Int64.Type}, {"LOS", Int64.Type}, {"MDC", Int64.Type}, {"MDC_NoPOA", Int64.Type}, {"PAY1", Int64.Type}, {"PCLASS_ORPROC", Int64.Type},
    {"PL_NCHS", Int64.Type}, {"PRDAY1", Int64.Type}, {"PRDAY2", Int64.Type}, {"PRDAY3", Int64.Type}, {"PRDAY4", Int64.Type}, {"PRDAY5", Int64.Type}, {"PRDAY6", Int64.Type}, {"PRDAY7", Int64.Type}, {"PRDAY8", type text}, {"PRDAY9", type text}, {"PRDAY10", type text},
    {"PRDAY11", type text}, {"PRDAY12", type text}, {"PRDAY13", type text}, {"PRDAY14", type text}, {"PRDAY15", type text}, {"PRDAY16", type text}, {"PRDAY17", type text}, {"PRDAY18", type text}, {"PRDAY19", type text}, {"PRDAY20", type text}, {"PRDAY21", type text},
    {"PRDAY22", type text}, {"PRDAY23", type text}, {"PRDAY24", type text}, {"PRDAY25", type text}, {"RACE", Int64.Type}, {"TOTCHG", Int64.Type}, {"TRAN_IN", Int64.Type}, {"TRAN_OUT", Int64.Type}, {"YEAR", Int64.Type}, {"ZIPINC_QRTL", Int64.Type}}),
    #"Filtered Rows3" = Table.SelectRows(#"Changed Type", each [AWEEKEND] >= 0 and [AWEEKEND] <= 1),
    #"Filtered Rows5" = Table.SelectRows(#"Filtered Rows3", each [HCUP_ED] >= 0 and [HCUP_ED] <= 1),
    #"Filtered Rows6" = Table.SelectRows(#"Filtered Rows5", each [PL_NCHS] >= 1 and [PL_NCHS] <= 6),
    #"Removed Columns" = Table.RemoveColumns(#"Filtered Rows6",{"I10_DX6", "I10_DX7", "I10_DX8", "I10_DX9", "I10_DX10", "I10_DX11", "I10_DX12", "I10_DX13", "I10_DX14", "I10_DX15", "I10_DX16", "I10_DX17", "I10_DX18", "I10_DX19", "I10_DX20", "I10_DX21", "I10_DX22",
    "I10_DX23", "I10_DX24", "I10_DX25", "I10_DX26", "I10_DX27", "I10_DX28", "I10_DX29", "I10_DX30", "I10_DX31", "I10_DX32", "I10_DX33", "I10_DX34", "I10_DX35", "I10_DX36", "I10_DX37", "I10_DX38", "I10_DX39", "I10_DX40", "I10_PR6", "I10_PR7", "I10_PR8", "I10_PR9",
    "I10_PR10","I10_PR11" ,"I10_PR12","I10_PR13","I10_PR14","I10_PR15", "I10_PR16", "I10_PR17", "I10_PR18", "I10_PR19", "I10_PR20", "I10_PR21", "I10_PR22", "I10_PR23", "I10_PR24", "I10_PR25", "DRG", "DRGVER", "DRG_NoPOA", "DISCWT", "I10_BIRTH", "MDC", "MDC_NoPOA",
    "YEAR"}),
    #"Replaced Value" = Table.ReplaceValue(#"Removed Columns","","No Data Recorded",Replacer.ReplaceValue,{"I10_DX2"}),
    #"Replaced Value1" = Table.ReplaceValue(#"Replaced Value","","No Data Recorded",Replacer.ReplaceValue,{"I10_DX1"}),
    #"Replaced Value2" = Table.ReplaceValue(#"Replaced Value1","","No Data Recorded",Replacer.ReplaceValue,{"I10_DX3"}),
    #"Replaced Value3" = Table.ReplaceValue(#"Replaced Value2","","No Data Recorded",Replacer.ReplaceValue,{"I10_DX4"}),
    #"Replaced Value4" = Table.ReplaceValue(#"Replaced Value3","","No Data Recorded",Replacer.ReplaceValue,{"I10_DX5"}),
    #"Replaced Value5" = Table.ReplaceValue(#"Replaced Value4","","No Data Recorded",Replacer.ReplaceValue,{"I10_PR1"}),
    #"Replaced Value6" = Table.ReplaceValue(#"Replaced Value5","","No Data Recorded",Replacer.ReplaceValue,{"I10_PR2"}),
    #"Replaced Value7" = Table.ReplaceValue(#"Replaced Value6","","No Data Recorded",Replacer.ReplaceValue,{"I10_PR3"}),
    #"Replaced Value8" = Table.ReplaceValue(#"Replaced Value7","","No Data Recorded",Replacer.ReplaceValue,{"I10_PR4"}),
    #"Replaced Value9" = Table.ReplaceValue(#"Replaced Value8","","No Data Recorded",Replacer.ReplaceValue,{"I10_PR5"})
in
    #"Replaced Value9"
```

Data Source Import:

- Data imported from a ***KID_2019_Core.csv*** file located on your system and saved it as ***HCUP_Data_Cleanup.pibx*** format. Specified delimiters and encoding types ensure accurate data reading.

Header Promotion:

- The first row of the dataset is promoted to serve as headers, which helps in column identification.

Data Processing and Imputing:

- Each column is assigned a specific data type, such as text or integer. This ensures consistent data handling and reduces errors during analysis.
- Rows are filtered based on specific criteria. In this instance, it filters where PL_NCHS is greater than or equal to 1 and HCUP_ED is equal to 1 and also same process for AWEEKEND. This narrows down the dataset to more relevant entries.
- Specific columns are removed to declutter the dataset and focus on essential data. This can be especially helpful when dealing with extensive datasets with many variables.
- 'No Data Recorded' values are replaced in multiple columns to ensure data consistency. This step is vital for maintaining the integrity of the dataset and preparing it for further analysis.
- The cleaned data, after all the specified transformations, is presented as 'Replaced Value9'. This data is now ready for further analysis or visualization.

```
let
    Source = Csv.Document(File.Contents("Original\KID_2019_Hospital.csv"),[Delimiter=",", Columns=16, Encoding=1252, QuoteStyle=QuoteStyle.None]),
    #"Promoted Headers" = Table.PromoteHeaders(Source, [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"HOSP_KID", Int64.Type}, {"N_BRTH_U", Int64.Type}, {"N_DISC_U", Int64.Type}, {"N_HOSP_U", Int64.Type}, {"S_BRTH_U", Int64.Type}, {"S_CHLD_U", Int64.Type}, {"S_CMPB_U", Int64.Type},
    {"S_DISC_U", Int64.Type}, {"S_HOSP_U", Int64.Type}, {"S_UNCB_U", Int64.Type}, {"HOSP_LOCTEACH", Int64.Type}, {"H_CONTRL", Int64.Type}, {"HOSP_REGION", Int64.Type}, {"KID_STRATUM", Int64.Type}, {"YEAR", Int64.Type}, {"HOSP_BEDSIZE", Int64.Type}}),
    #"Removed Columns" = Table.RemoveColumns(#"Changed Type",{"S_BRTH_U", "S_CHLD_U", "S_CMPB_U", "S_DISC_U", "S_HOSP_U", "S_UNCB_U", "KID_STRATUM", "YEAR", "N_BRTH_U"})
in
    #"Removed Columns"
```

Data Source Import:

- Data imported from a ***KID_2019_Hospital.csv*** file located on your system.

Header Promotion:

- The first row of the dataset is promoted to headers, as identification throughout the process.

Data Processing and Imputing:

- Individual columns are given distinct data types, such as integer, to ensure uniform data handling.
- Certain columns are discarded to focus on the more relevant data points. This step helps simplify the dataset and make subsequent analysis more straightforward.
- After all transformations, the data is outputted as 'Removed Columns'. This cleaned data is now prepared for any subsequent operations or analyses.

```
let
    Source = Csv.Document(File.Contents("\Original\KID_2019_Severity.csv"),[Delimiter=",", Columns=5, Encoding=1252, QuoteStyle=QuoteStyle.None]),
    #"Promoted Headers" = Table.PromoteHeaders(Source, [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"HOSP_KID", Int64.Type}, {"RECNUM", Int64.Type}, {"APRDRG", Int64.Type}, {"APRDRG_Risk_Mortality", Int64.Type}, {"APRDRG_Severity", Int64.Type}}),
    #"Filtered Rows" = Table.SelectRows(#"Changed Type", each [APRDRG] >= 0),
    #"Filtered Rows1" = Table.SelectRows(#"Filtered Rows", each [APRDRG_Risk_Mortality] >= 0),
    #"Filtered Rows2" = Table.SelectRows(#"Filtered Rows1", each [APRDRG_Severity] >= 0),
    #"Filtered Rows3" = Table.SelectRows(#"Filtered Rows2", each [HOSP_KID] <> null or [HOSP_KID] <> ""),
    #"Filtered Rows4" = Table.SelectRows(#"Filtered Rows3", each [RECNUM] <> null or [RECNUM] <> "")
in
    #"Filtered Rows4"
```

Data Source Import:

- Data imported from a ***KID_2019_Severity.csv*** file located on your system.

Header Promotion:

- The initial row of the dataset is set as headers, enabling clearer column identification throughout the data processing phase.

Data Processing and Imputing:

- Columns, such as 'HOSP_KID', 'RECNUM', 'APRDRG', 'APRDRG_Risk_Mortality', and 'APRDRG_Severity', are assigned specific data types, predominantly integer, to ensure consistent data handling.
- Step 1: Rows where the 'APRDRG' column is not equal to 0 are selected.
- Step 2: From the filtered data in Step 1, rows with a value greater than 0 in the 'APRDRG_Risk_Mortality' column are further filtered.
- Step 3: Further refining the data, rows where the 'APRDRG_Severity' column has a value equal to or above 0 are selected.
- Step 4: From the previous selection, rows without a blank 'HOSP_KID' column are isolated.
- Step 5: Lastly, only rows where the 'RECNUM' column is not null or blank are fitlered.
- After all the transformation stages, the dataset is presented as 'Filtered Rows4'. This refined dataset is now set for data analysis.
- Lastly, ***HCUP_Data_Cleanup.pibx*** to extract the new cleanup and transform .csv files under the (*Data Cleaning/Transformed*) folder.

Merging Data in Python

Dataset Importation:

- Import ‘dask.dataframe’ library for parallel computing with larger datasets.
- Using Dask, three datasets — ***Hospital***, ***Core***, and ***Severity*** — are read into memory.

Merging Hospital with Severity Dataset:

- The Hospital and Severity datasets are merged on the 'HOSP_KID' column. It is a 'left join', which means all entries from the Hospital dataset and corresponding entries from the Severity dataset will be included.
- The resultant merged dataset is saved as a ***Hospital_Severity.csv*** file in the specified output directory (*Dataset/Data Cleaning/Merged*).

Merging Hospital with Core Dataset:

- Similarly, the Hospital and Core datasets are merged on the 'HOSP_KID' column using a 'left join'.
- This merged dataset is also saved as a ***Hospital_Core.csv*** file in the designated output location (Dataset/Data Cleaning/Merged).

Exploratory Data Analysis in PowerBI

Dataset Importation:

- Data imported from the location *\Dataset\Data Cleaning\Merged* folder **Hospital_Core.csv, Hospital_Severity.csv** and **icd10cm_Codes_Diagnosis.csv** file located on your system and saved it as ***HCUP_EDA.pibx*** format.

Visualization:

- *Distribution by Age Group and Gender used by Stacked column chart*: Y-Axis shows the percentage of hospital discharges (% grand total count of ***RECNUM*** columns). X-Axis is presented by ***GENDER*** column. In this chart, ***AGE_GROUPS*** custom column was created, and apply as in stack column chart Legend.
- *Distribution by Age Group and Race used by Stacked column chart:* Y-Axis shows the percentage of hospital discharges (% grand total count of ***RECNUM*** columns). X-Axis is presented by ***RACE_DESC*** column. In this chart, ***AGE_GROUPS*** custom column was created, and apply as in stack column chart Legend.
- *Distribution by Race used by Stacked bar chart:* X-Axis shows the percentage of hospital discharges (% grand total count of ***RECNUM*** columns). Y-Axis is presented by ***RACE_DESC*** column.
- *Number of Hospitals in US Regions used by Donut chart:* ***HOSPITAL_REGIONS*** column used as Legend and Value used the (count of ***HOSP_KID)*** column as total number of hospitals across regions.
- *Average age of LOS by Gender and Age Group used by Clustered column chart*: X-Axis shows ***Gender*** column and Y-Axis presented (***Average of LOS***) average length of stay column. ***Age Group*** column apply in the Legend in the clustered column chart.
- *Distribution of Diabetes in KID's Inpatient used by Pie chart*: ***Diabetes_Type*** column used as in Legend and Value used the (count of ***RECNUM****)* column as total number of diabetes.
- *Top 5 Primary Diagnoses in 2019 used by Stacked column chart*: X-Axis shows the diagnosis (***Short Name_DX1*** columns from icd10 table). Y-Axis is presented by total discharge which is from custom created column name called (% of ***Total Discharges***).

Apply page level filters for ***AGE*** column is less than or equal 18 and custom column ***Diabetes_Flag*** is selected as Diabetes.

Clustering with BIRCH in Python

Dataset Importation:

- Import libraries such as Pandas, Dask, and Numpy are imported to handle data manipulation and computations.
- Datasets are read into memory using Dask for efficient parallel computing.

Data Preprocessing:

- The **hospital_core_df** and **hospital_severity_df** dataframes are created.
- The **hospital_core_df** is filtered based on age, including only individuals aged 0 to 18.
- A vectorized approach is used to identify records with a pre-existing diagnosis of type 1 diabetes, searching across multiple diagnosis columns.

Feature Engineering:

- A custom function *extract_diagnosis_codes* is applied to extract all relevant diagnosis codes related to type 1 diabetes.
- The diagnosis codes are then one-hot encoded to prepare them for machine learning models.

Data Transformation and Cleaning:

- Columns are renamed for clarity, and null values are handled appropriately to ensure data integrity.
- The dataframes **hospital_core_df** and **hospital_severity_df** are merged on the '**RECNUM**' column using an inner join.

Data Scaling and Dimensionality Reduction:

- The StandardScaler function from Scikit-learn is used to scale the features.
- Principal Component Analysis (PCA) is performed to reduce dimensionality, and the explained variance is visualized to determine the number of components to retain.

Elbow Method:

- The Elbow Method is used to determine the optimal number of clusters, which is visualized through a plot.
- A summary of the clusters is obtained by grouping the data by the cluster labels and computing the mean for each feature.

Cluster Profiling with BIRCH Algorithm:

- After compare and evaluate with other method like **KMeans**, **DBSCAN**. **BIRCH** method is significantly better in *Silhouette Score*. So, we decided BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) algorithm is applied for further analysis.
- The data is then grouped by the 'BIRCH_Cluster' labels to profile the clusters. The mean of each feature within each cluster is calculated to provide insights into the characteristics of the clusters.

Clustering with KModes in Python

Dataset Importation:

- Import libraries such as Pandas, Dask, and Numpy and read the hospital datasets using Dask for efficient memory.
- Apply age filters to isolate patients between 0 to 18 years and filter diagnosis codes for type 1 diabetes.

Data Transformation:

- Categorize patients into age groups (*'Infants'*, *'Children'*, and *'Adolescents'*). Extract relevant diagnosis codes and convert Dask dataframes to Pandas dataframes for further processing.

Elbow Method:

- Use the Elbow method to determine the optimal number of clusters by plotting the cost function against the number of clusters.

KModes Algorithm:

- Apply the KModes clustering algorithm using the optimal number of clusters. Ensure that the algorithm is initialized with a consistent random seed for reproducibility.

Export Predicted Cluster Dataset:

- Export the predicted clustering dataframe with assigned clusters to a **HCUP_Clustering.csv** file for further cluster data analysis.

Cluster Profiles Analysis in PowerBI

Dataset Importation:

- Data imported from the location \Dataset\Data Cleaning\Merged folder **HCUP_Clustering.csv**, and **icd10cm_Codes_Diagnosis.csv** file located on your system and saved it as **HCUP_Cluster_Profiles.pibx** format.

Visualizations:

Distribution of Age

- **Cluster 1**: Distribution of Age used by Donut chart, ***AGE_GROUP*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **1** on visual level.
- **Cluster 2**: Distribution of Age used by Donut chart, ***AGE_GROUP*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **2** on visual level.
- **Cluster 3**: Distribution of Age used by Donut chart, ***AGE_GROUP*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **3** on visual level.

Distribution of Gender

- **Cluster 1**: Distribution of Gender used by Donut chart, ***GENDER*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **1** on visual level.
- **Cluster 2**: Distribution of Gender used by Donut chart, ***GENDER*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **2** on visual level.
- **Cluster 3**: Distribution of Gender used by Donut chart, ***GENDER*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **3** on visual level.

Distribution of Race

- **Cluster 1**: Distribution of Race used by Donut chart, ***RACE_DESC*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **1** on visual level.
- **Cluster 2**: Distribution of Race used by Donut chart, ***RACE_DESC*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **2** on visual level.
- **Cluster 3**: Distribution of Race used by Donut chart, ***RACE_DESC*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **3** on visual level.

Distribution of Hospital Region and Age Groups

- **Cluster 1**: Distribution of Hospital Region and Age Groups used by Clustered bar chart, Y-Axis shows the ***HOSPITAL_REGION*** column. X-Axis is presented by (count of ***RECNUM***) column as No. of Discharges. ***AGE_GROUP*** column as in the Legend in the clustered bar chart. Apply filter for ***CLUSTERS*** column equal **1** on visual level.
- **Cluster 2**: Distribution of Hospital Region and Age Groups used by Clustered bar chart, Y-Axis shows the ***HOSPITAL_REGION*** column. X-Axis is presented by (count of ***RECNUM***) column as No. of Discharges. ***AGE_GROUP*** column as in the Legend in the clustered bar chart. Apply filter for ***CLUSTERS*** column equal **2** on visual level.
- **Cluster 3**: Distribution of Hospital Region and Age Groups used by Clustered bar chart, Y-Axis shows the ***HOSPITAL_REGION*** column. X-Axis is presented by (count of ***RECNUM***) column as No. of Discharges. ***AGE_GROUP*** column as in the Legend in the clustered bar chart. Apply filter for ***CLUSTERS*** column equal **3** on visual level

Distribution of Discharges by Hospital Region

- **Cluster 1**: Distribution of Discharges by Hospital Region used by Donut chart, ***HOSPITAL_REGION*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **1** on visual level.
- **Cluster 2**: Distribution of Discharges by Hospital Region used by Donut chart, ***HOSPITAL_REGION*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **2** on visual level.
- **Cluster 3**: Distribution of Discharges by Hospital Region used by Donut chart, ***HOSPITAL_REGION*** column used as in Legend and Values used the (count of ***RECNUM***) column as No. of Discharges. Apply filter for ***CLUSTERS*** column equal **3** on visual level.

Average Length of Stay based on Severity

- **Cluster 1**: Average Length of Stay based on Severity used by Clustered column chart, X-Axis shows ***ALL_PATIENT_SEVERITY*** column and Y-Axis presented (***Average of LOS***) average length of stay column. Apply filter for ***CLUSTERS*** column equal **1** on visual level.
- **Cluster 2**: Average Length of Stay based on Severity used by Clustered column chart, X-Axis shows ***ALL_PATIENT_SEVERITY*** column and Y-Axis presented (***Average of LOS***) average length of stay column. Apply filter for ***CLUSTERS*** column equal **2** on visual level.
- **Cluster 3**: Average Length of Stay based on Severity used by Clustered column chart, X-Axis shows ***ALL_PATIENT_SEVERITY*** column and Y-Axis presented (***Average of LOS***) average length of stay column. Apply filter for ***CLUSTERS*** column equal **3** on visual level.

Chi-Square Hypothesis Testing in Python

Dataset Importation:

- Import libraries such as Pandas, and read the cluster datasets **HCUP_Clustering.csv** to dataframe. It includes relevant columns for the analysis, including 'GENDER', 'ALL_PATIENT_SEVERITY','TYPE1_DIABETES_CODES','HOSPITAL_REGION', 'RACE_DESC', 'LOS', and 'AGE_GROUP'.
- To conducting hypothesis tests, first to import scipy.stats library from stats.

Gender and Severity of Diagnosis:

- Create a contingency table for 'GENDER' and 'ALL_PATIENT_SEVERITY' whether the test is between two different categorical variables are associate or not.
- Perform a chi-square test and interpret the results.

Gender and Type 1 Diabetes Codes:

- Create a contingency table for 'GENDER' and 'TYPE1_DIABETES_CODES'.
- Perform a chi-square test and interpret the results.

Hospital Region vs. Ethnicity:

- Create a contingency table for 'HOSPITAL_REGION' and 'RACE_DESC'.
- Perform a chi-square test and interpret the results.

Ethnicity vs. Type 1 Diabetes Codes:

- Create a contingency table for 'RACE_DESC' and 'TYPE1_DIABETES_CODES'.
- Perform a chi-square test and interpret the results.

Length of Stay vs. Age Group:

- Create a contingency table for 'LOS' and 'AGE_GROUP'.
- Perform a chi-square test and interpret the results.

Length of Stay vs. Severity:

- Create a contingency table for 'LOS' and 'ALL_PATIENT_SEVERITY'.

- Perform a chi-square test and interpret the results.

Age Group vs. Severity:
- Create a contingency table for 'AGE_GROUP' and 'ALL_PATIENT_SEVERITY'.
- Perform a chi-square test and interpret the results.

Interpreting the Results:
- These hypothesis tests will output the Chi-square statistic, p-value, degrees of freedom, and a conclusion to accept or reject the null hypothesis based on 5% significance level

Appendix: How to get data import into PowerBI and Python

Prerequisites:
- Power BI Desktop and Python (Anaconda) Jupyter Notebook installed on your computer.

List of File Name and Type:
- Power BI
    1. HCUP_Data_Cleanup (pbix)
    2. HCUP_EDA (pbix)
    3. HCUP_Cluster_Profiles (pbix)
- Python Jupyter Notebook
    1. hcup_merging_datasource_v1(W1_Final)
    2. hcup_clustering_v2.8.1(W4_FInal)
    3. hcup_clustering_v2.6.1(W3_Final)
    4. hcup_chi_square_testing_v2(W5_Final)

Open Power BI Desktop
- Launch the Power BI Desktop application on your computer.

Get Data
- In the Home tab on the ribbon, click on the 'Get Data' option.
- Select 'Text/CSV' from the options presented.

Open CSV File
- Navigate through your directories and select the CSV file you want to import.
- Click 'Open' to proceed.

Preview Data
- Power BI will present a preview of the data.
- Review to ensure that the columns and rows are displayed correctly.

Load or Edit

- To make changes to the data before importing, click 'Transform Data'. This will open the Power Query Editor where you can perform data cleaning and transformation.
- If no changes are needed, click 'Load' to import the data directly into Power BI.

Data Model

- Once loaded, the dataset will appear in the 'Fields' pane.
- You can now use this data to create reports and visuals.

Save Your Work

- Save your Power BI project by clicking on 'File' and then 'Save', or by pressing Ctrl + S.
- Name your project and choose a location to save the file.

Launch Jupyter Notebook:

- Type jupyter notebook and hit Enter.
- Your default web browser will open with the Jupyter file explorer.
- Click on the "Open" in the menu of the Jupyter file explorer.

Importing CSV Data:

- Change your file path into pd.read_csv('your_data.csv')

----------End----------