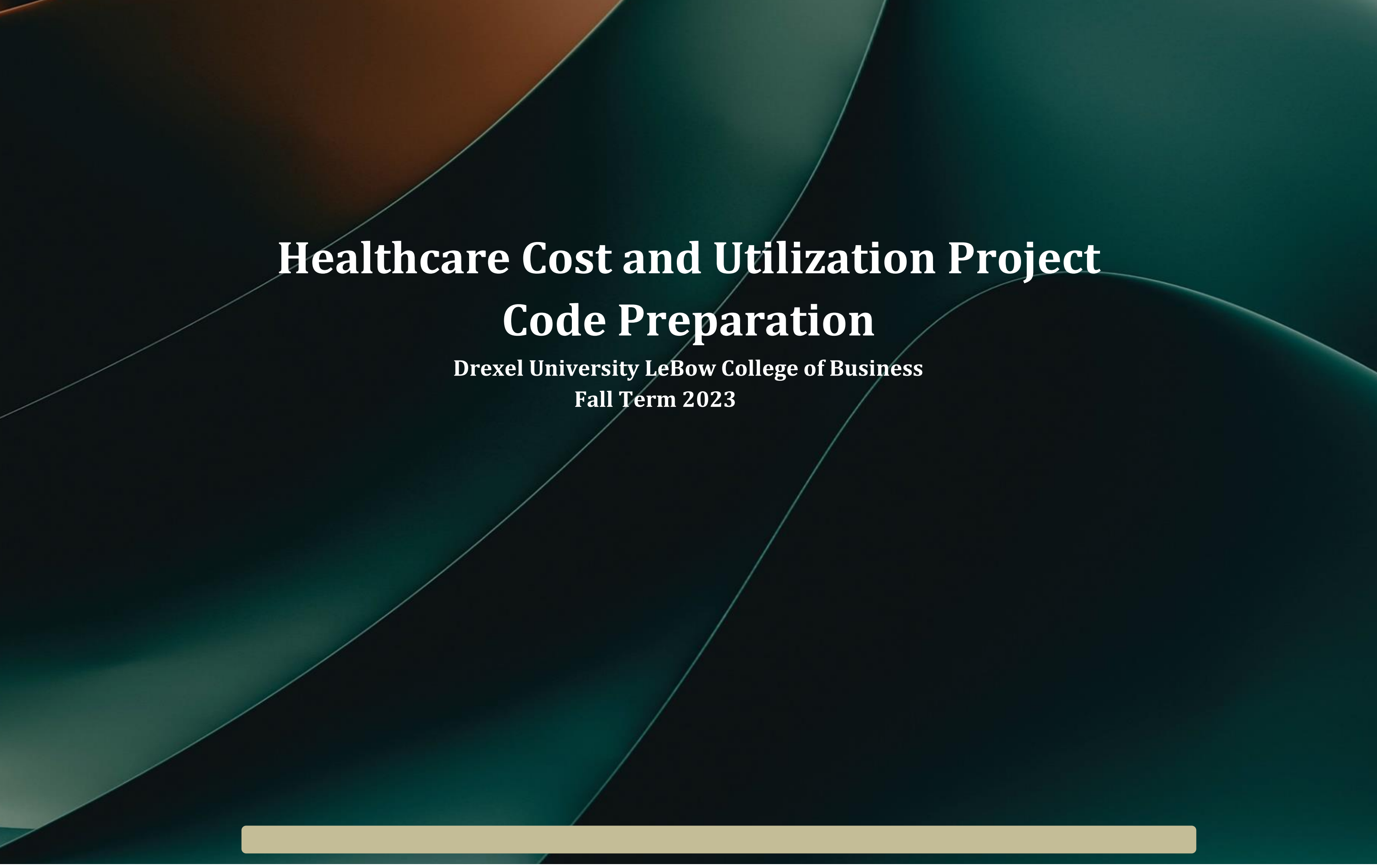


# Healthcare Cost and Utilization Project Code Preparation

**Drexel University LeBow College of Business**
**Fall Term 2023**

## Type 1 Diabetes Table of Contents

## Asthma Table of Contents

# Merging Data

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).

# Diabetes K-Modes Clustering

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.

## Diabetes Chi-Square Hypothesis Testing

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:

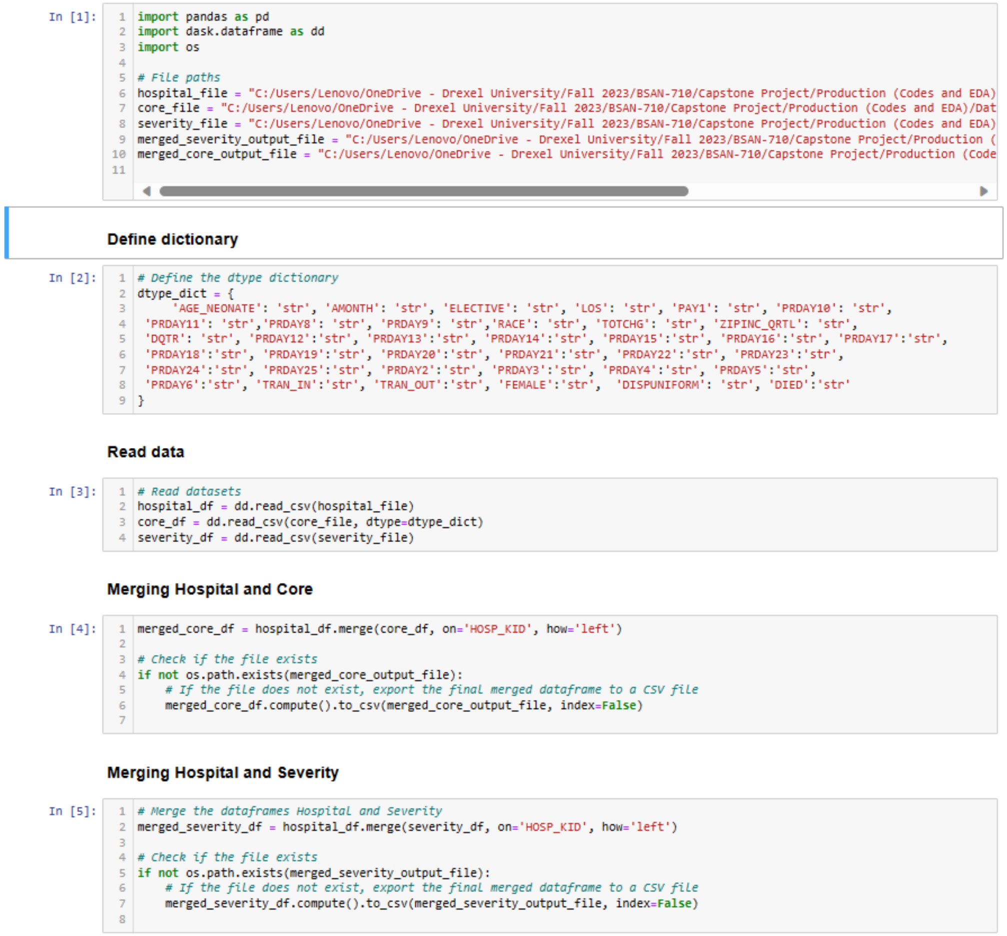
- Create a contingency table for 'AGE_GROUP' and 'ALL_PATIENT_SEVERITY'.
- Perform a chi-square test and interpret the results.

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

# Diabetes Codes: Jupyter Notebook

- Merging data source



In [1]:
```
import pandas as pd
import dask.dataframe as dd
import os

# File paths
hospital_file = "C:/Users/Lenovo/OneDrive - Drexel University/Fall 2023/BSAN-710/Capstone Project/Production (Codes and EDA)
core_file = "C:/Users/Lenovo/OneDrive - Drexel University/Fall 2023/BSAN-710/Capstone Project/Production (Codes and EDA)/Dat
severity_file = "C:/Users/Lenovo/OneDrive - Drexel University/Fall 2023/BSAN-710/Capstone Project/Production (Codes and EDA)
merged_severity_output_file = "C:/Users/Lenovo/OneDrive - Drexel University/Fall 2023/BSAN-710/Capstone Project/Production (
merged_core_output_file = "C:/Users/Lenovo/OneDrive - Drexel University/Fall 2023/BSAN-710/Capstone Project/Production (Code

```

#### Define dictionary

In [2]:
```
# Define the dtype dictionary
dtype_dict = {
     'AGE_NEONATE': 'str', 'AMONTH': 'str', 'ELECTIVE': 'str', 'LOS': 'str', 'PAY1': 'str', 'PRDAY10': 'str',
 'PRDAY11': 'str','PRDAY8': 'str', 'PRDAY9': 'str','RACE': 'str', 'TOTCHG': 'str', 'ZIPINC_QRTL': 'str',
 'DQTR': 'str', 'PRDAY12':'str', 'PRDAY13':'str', 'PRDAY14':'str', 'PRDAY15':'str', 'PRDAY16':'str', 'PRDAY17':'str',
 'PRDAY18':'str', 'PRDAY19':'str', 'PRDAY20':'str', 'PRDAY21':'str', 'PRDAY22':'str', 'PRDAY23':'str',
 'PRDAY24':'str', 'PRDAY25':'str', 'PRDAY2':'str', 'PRDAY3':'str', 'PRDAY4':'str', 'PRDAY5':'str',
 'PRDAY6':'str', 'TRAN_IN':'str', 'TRAN_OUT':'str', 'FEMALE':'str',  'DISPUNIFORM': 'str', 'DIED':'str'
}
```

#### Read data

In [3]:
```
# Read datasets
hospital_df = dd.read_csv(hospital_file)
core_df = dd.read_csv(core_file, dtype=dtype_dict)
severity_df = dd.read_csv(severity_file)
```

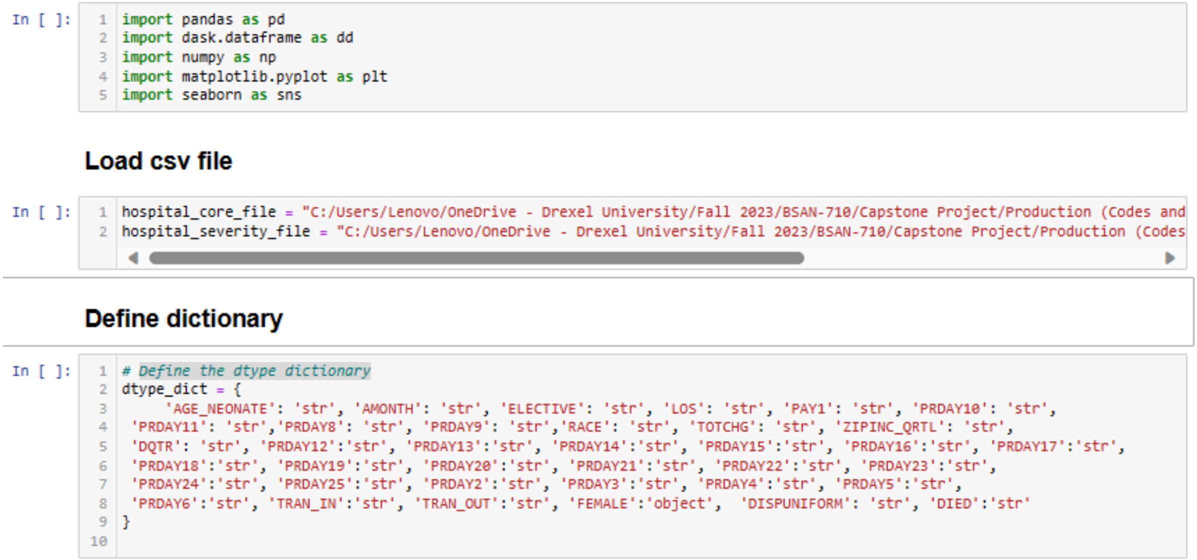
#### Merging Hospital and Core

In [4]:
```
merged_core_df = hospital_df.merge(core_df, on='HOSP_KID', how='left')

# Check if the file exists
if not os.path.exists(merged_core_output_file):
    # If the file does not exist, export the final merged dataframe to a CSV file
    merged_core_df.compute().to_csv(merged_core_output_file, index=False)

```

#### Merging Hospital and Severity

In [5]:
```
# Merge the dataframes Hospital and Severity
merged_severity_df = hospital_df.merge(severity_df, on='HOSP_KID', how='left')

# Check if the file exists
if not os.path.exists(merged_severity_output_file):
    # If the file does not exist, export the final merged dataframe to a CSV file
    merged_severity_df.compute().to_csv(merged_severity_output_file, index=False)

```

- Clustering with KModes



In [ ]:
```
import pandas as pd
import dask.dataframe as dd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

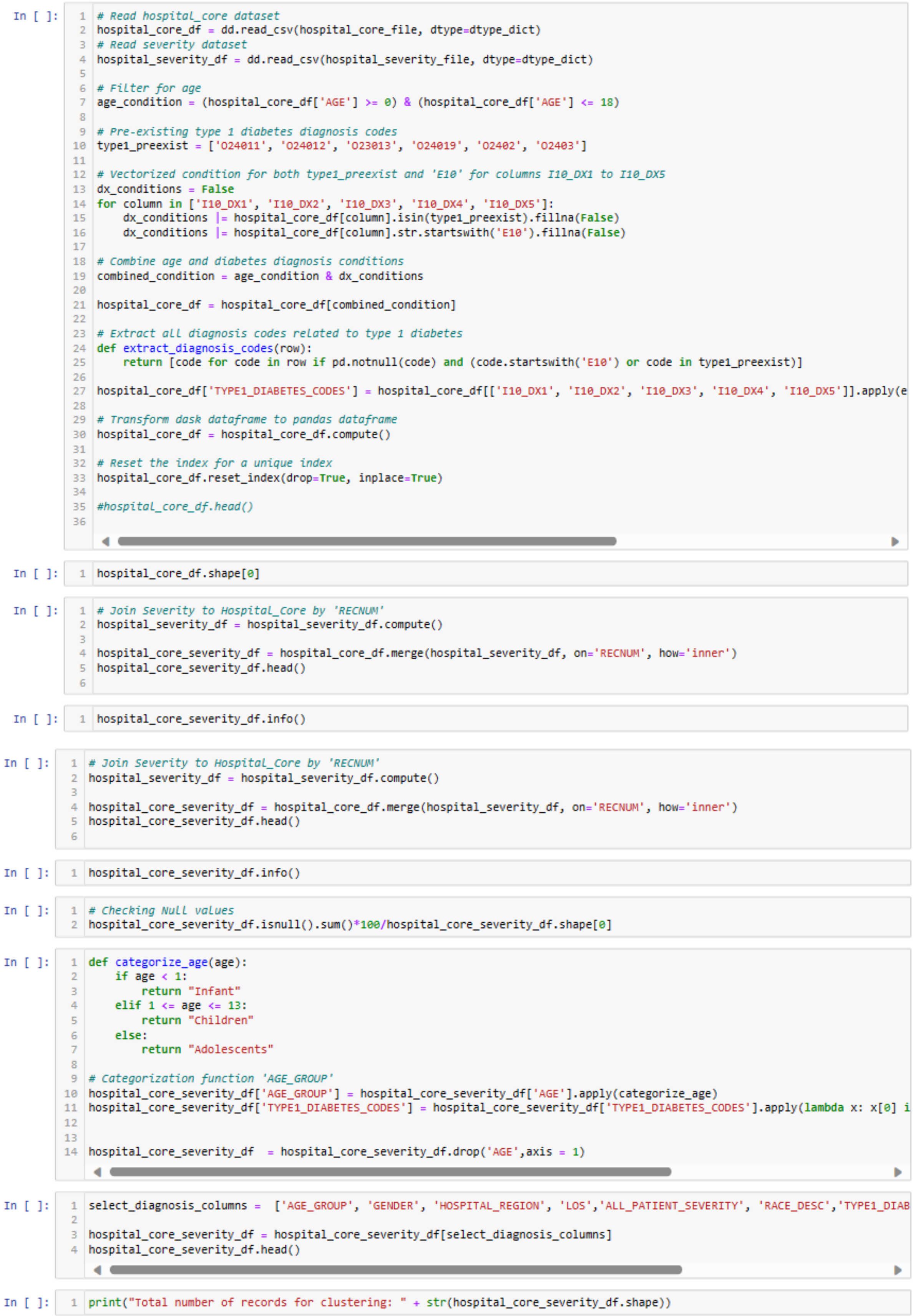
## Load csv file

In [ ]:
```
hospital_core_file = "C:/Users/Lenovo/OneDrive - Drexel University/Fall 2023/BSAN-710/Capstone Project/Production (Codes and
hospital_severity_file = "C:/Users/Lenovo/OneDrive - Drexel University/Fall 2023/BSAN-710/Capstone Project/Production (Codes
```

## Define dictionary

In [ ]:
```
# Define the dtype dictionary
dtype_dict = {
     'AGE_NEONATE': 'str', 'AMONTH': 'str', 'ELECTIVE': 'str', 'LOS': 'str', 'PAY1': 'str', 'PRDAY10': 'str',
 'PRDAY11': 'str','PRDAY8': 'str', 'PRDAY9': 'str','RACE': 'str', 'TOTCHG': 'str', 'ZIPINC_QRTL': 'str',
 'DQTR': 'str', 'PRDAY12':'str', 'PRDAY13':'str', 'PRDAY14':'str', 'PRDAY15':'str', 'PRDAY16':'str', 'PRDAY17':'str',
 'PRDAY18':'str', 'PRDAY19':'str', 'PRDAY20':'str', 'PRDAY21':'str', 'PRDAY22':'str', 'PRDAY23':'str',
 'PRDAY24':'str', 'PRDAY25':'str', 'PRDAY2':'str', 'PRDAY3':'str', 'PRDAY4':'str', 'PRDAY5':'str',
 'PRDAY6':'str', 'TRAN_IN':'str', 'TRAN_OUT':'str', 'FEMALE':'object',  'DISPUNIFORM': 'str', 'DIED':'str'
}

```

### Read dataset and filter only type 1 diabetes codes tranpose into one columns



In [ ]:

```python
# Read hospital_core dataset
hospital_core_df = dd.read_csv(hospital_core_file, dtype=dtype_dict)
# Read severity dataset
hospital_severity_df = dd.read_csv(hospital_severity_file, dtype=dtype_dict)

# Filter for age
age_condition = (hospital_core_df['AGE'] >= 0) & (hospital_core_df['AGE'] <= 18)

# Pre-existing type 1 diabetes diagnosis codes
type1_preexist = ['O24011', 'O24012', 'O23013', 'O24019', 'O2402', 'O2403']

# Vectorized condition for both type1_preexist and 'E10' for columns I10_DX1 to I10_DX5
dx_conditions = False
for column in ['I10_DX1', 'I10_DX2', 'I10_DX3', 'I10_DX4', 'I10_DX5']:
    dx_conditions |= hospital_core_df[column].isin(type1_preexist).fillna(False)
    dx_conditions |= hospital_core_df[column].str.startswith('E10').fillna(False)

# Combine age and diabetes diagnosis conditions
combined_condition = age_condition & dx_conditions

hospital_core_df = hospital_core_df[combined_condition]

# Extract all diagnosis codes related to type 1 diabetes
def extract_diagnosis_codes(row):
    return [code for code in row if pd.notnull(code) and (code.startswith('E10') or code in type1_preexist)]

hospital_core_df['TYPE1_DIABETES_CODES'] = hospital_core_df[['I10_DX1', 'I10_DX2', 'I10_DX3', 'I10_DX4', 'I10_DX5']].apply(e

# Transform dask dataframe to pandas dataframe
hospital_core_df = hospital_core_df.compute()

# Reset the index for a unique index
hospital_core_df.reset_index(drop=True, inplace=True)

#hospital_core_df.head()

```

In [ ]:

```python
hospital_core_df.shape[0]
```

In [ ]:

```python
# Join Severity to Hospital_Core by 'RECNUM'
hospital_severity_df = hospital_severity_df.compute()

hospital_core_severity_df = hospital_core_df.merge(hospital_severity_df, on='RECNUM', how='inner')
hospital_core_severity_df.head()

```

In [ ]:

```python
hospital_core_severity_df.info()
```

In [ ]:

```python
# Join Severity to Hospital_Core by 'RECNUM'
hospital_severity_df = hospital_severity_df.compute()

hospital_core_severity_df = hospital_core_df.merge(hospital_severity_df, on='RECNUM', how='inner')
hospital_core_severity_df.head()

```

In [ ]:

```python
hospital_core_severity_df.info()
```

In [ ]:

```python
# Checking Null values
hospital_core_severity_df.isnull().sum()*100/hospital_core_severity_df.shape[0]
```

In [ ]:

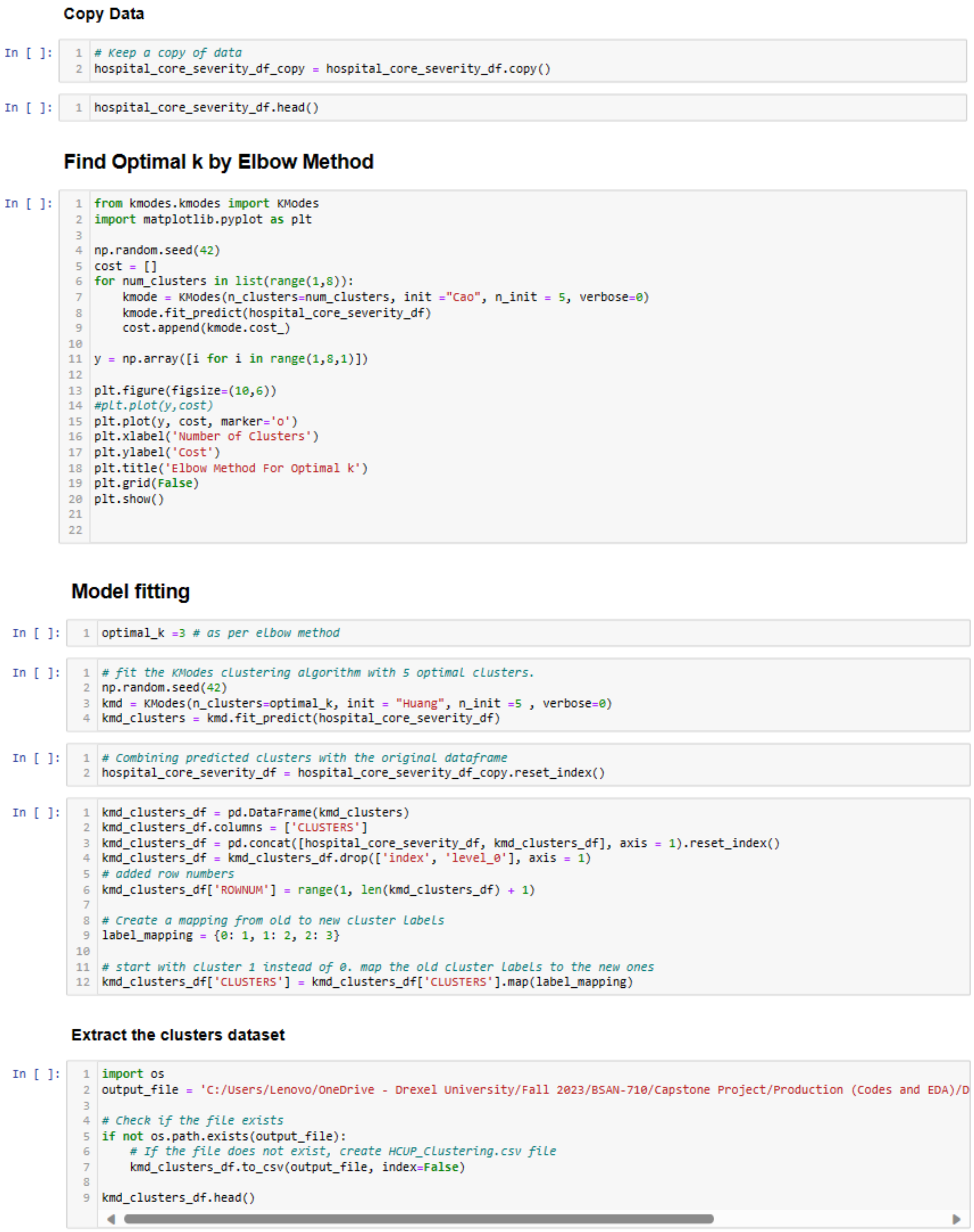
```python
def categorize_age(age):
    if age < 1:
        return "Infant"
    elif 1 <= age <= 13:
        return "Children"
    else:
        return "Adolescents"

# Categorization function 'AGE_GROUP'
hospital_core_severity_df['AGE_GROUP'] = hospital_core_severity_df['AGE'].apply(categorize_age)
hospital_core_severity_df['TYPE1_DIABETES_CODES'] = hospital_core_severity_df['TYPE1_DIABETES_CODES'].apply(lambda x: x[0] i


hospital_core_severity_df  = hospital_core_severity_df.drop('AGE',axis = 1)
```

In [ ]:

```python
select_diagnosis_columns =  ['AGE_GROUP', 'GENDER', 'HOSPITAL_REGION', 'LOS','ALL_PATIENT_SEVERITY', 'RACE_DESC','TYPE1_DIAB

hospital_core_severity_df = hospital_core_severity_df[select_diagnosis_columns]
hospital_core_severity_df.head()
```

In [ ]:

```python
print("Total number of records for clustering: " + str(hospital_core_severity_df.shape))
```



### Copy Data

In [ ]:

```python
# Keep a copy of data
hospital_core_severity_df_copy = hospital_core_severity_df.copy()
```

In [ ]:

```python
hospital_core_severity_df.head()
```

## Find Optimal k by Elbow Method

In [ ]:

```python
from kmodes.kmodes import KModes
import matplotlib.pyplot as plt

np.random.seed(42)
cost = []
for num_clusters in list(range(1,8)):
    kmode = KModes(n_clusters=num_clusters, init ="Cao", n_init = 5, verbose=0)
    kmode.fit_predict(hospital_core_severity_df)
    cost.append(kmode.cost_)

y = np.array([i for i in range(1,8,1)])

plt.figure(figsize=(10,6))
#plt.plot(y,cost)
plt.plot(y, cost, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Cost')
plt.title('Elbow Method For Optimal k')
plt.grid(False)
plt.show()


```

## Model fitting

In [ ]:

```python
optimal_k =3 # as per elbow method
```

In [ ]:

```python
# fit the KModes clustering algorithm with 5 optimal clusters.
np.random.seed(42)
kmd = KModes(n_clusters=optimal_k, init = "Huang", n_init =5 , verbose=0)
kmd_clusters = kmd.fit_predict(hospital_core_severity_df)
```

In [ ]:

```python
# Combining predicted clusters with the original dataframe
hospital_core_severity_df = hospital_core_severity_df_copy.reset_index()
```

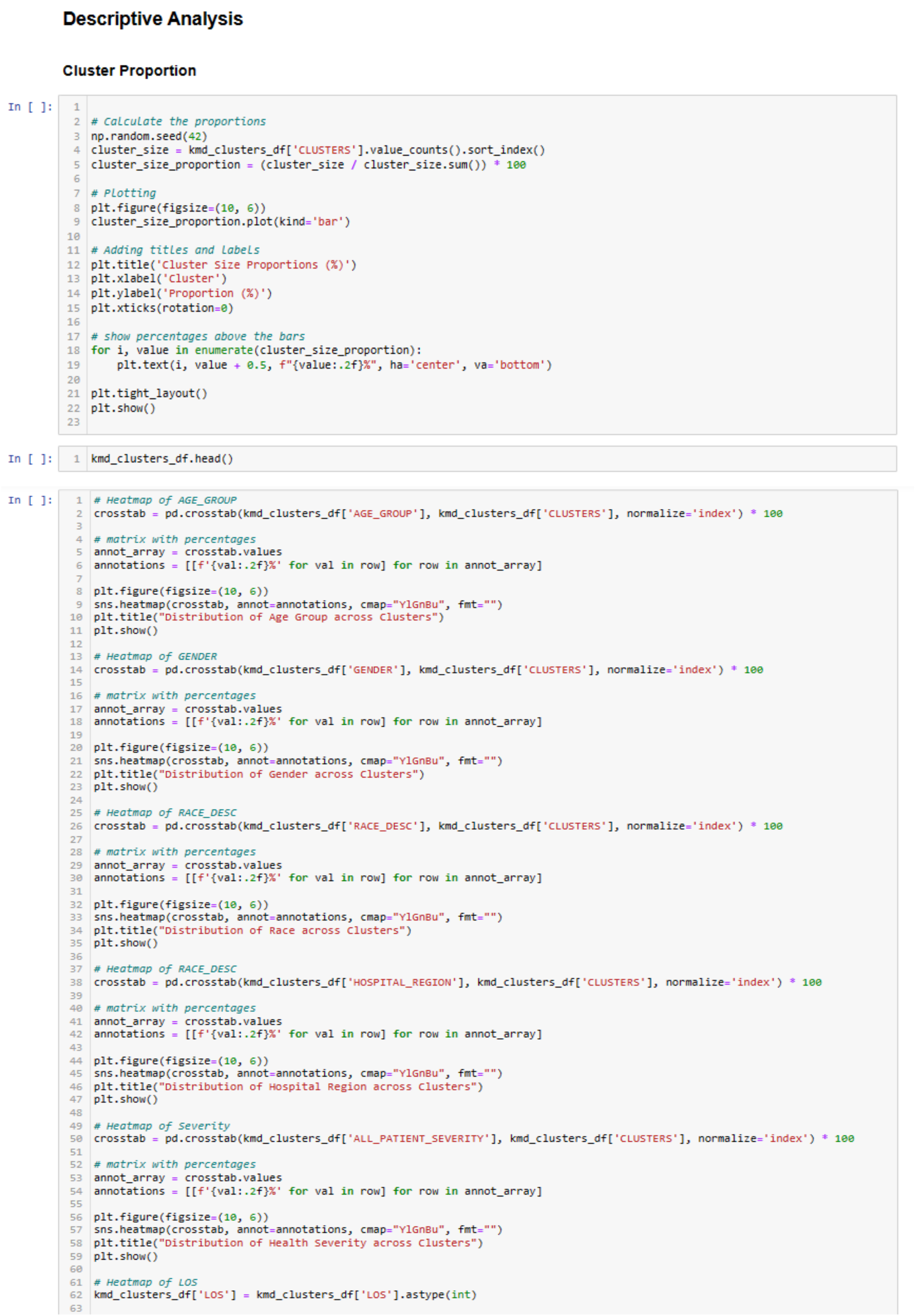
In [ ]:

```python
kmd_clusters_df = pd.DataFrame(kmd_clusters)
kmd_clusters_df.columns = ['CLUSTERS']
kmd_clusters_df = pd.concat([hospital_core_severity_df, kmd_clusters_df], axis = 1).reset_index()
kmd_clusters_df = kmd_clusters_df.drop(['index', 'level_0'], axis = 1)
# added row numbers
kmd_clusters_df['ROWNUM'] = range(1, len(kmd_clusters_df) + 1)

# Create a mapping from old to new cluster labels
label_mapping = {0: 1, 1: 2, 2: 3}

# start with cluster 1 instead of 0. map the old cluster labels to the new ones
kmd_clusters_df['CLUSTERS'] = kmd_clusters_df['CLUSTERS'].map(label_mapping)
```

### Extract the clusters dataset

In [ ]:

```python
import os
output_file = 'C:/Users/Lenovo/OneDrive - Drexel University/Fall 2023/BSAN-710/Capstone Project/Production (Codes and EDA)/D

# Check if the file exists
if not os.path.exists(output_file):
    # If the file does not exist, create HCUP_Clustering.csv file
    kmd_clusters_df.to_csv(output_file, index=False)

kmd_clusters_df.head()
```



## Descriptive Analysis

### Cluster Proportion

In [ ]:

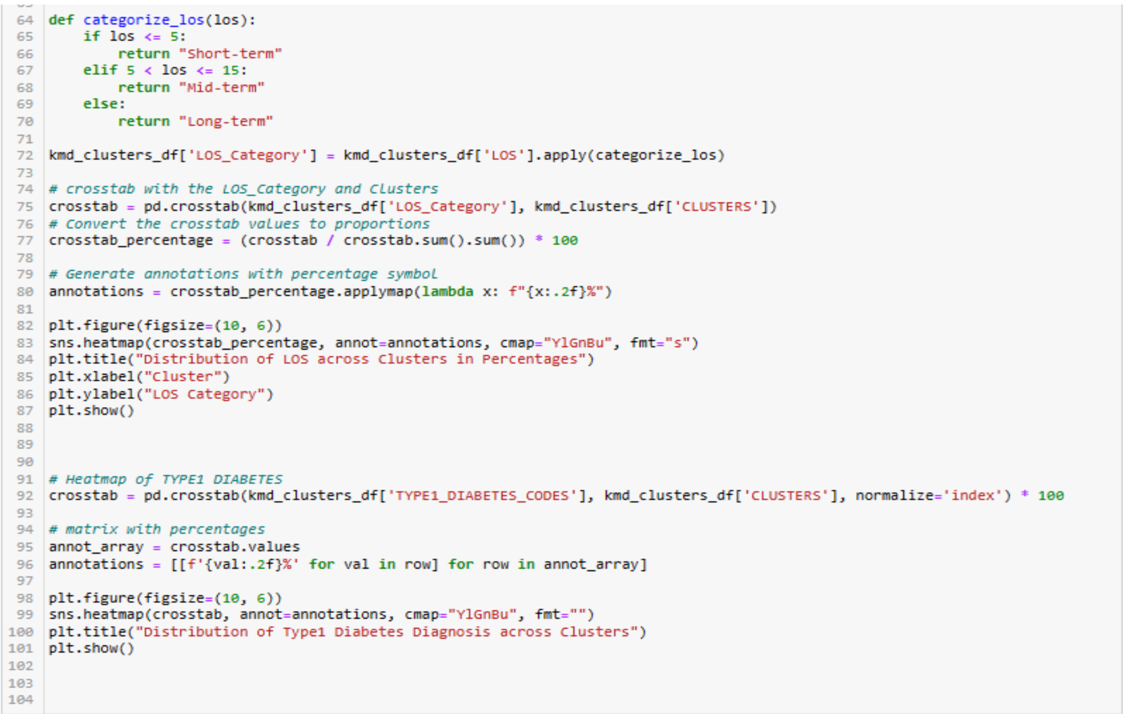
```

# Calculate the proportions
np.random.seed(42)
cluster_size = kmd_clusters_df['CLUSTERS'].value_counts().sort_index()
cluster_size_proportion = (cluster_size / cluster_size.sum()) * 100

# Plotting
plt.figure(figsize=(10, 6))
cluster_size_proportion.plot(kind='bar')

# Adding titles and labels
plt.title('Cluster Size Proportions (%)')
plt.xlabel('Cluster')
plt.ylabel('Proportion (%)')
plt.xticks(rotation=0)

# show percentages above the bars
for i, value in enumerate(cluster_size_proportion):
    plt.text(i, value + 0.5, f"{value:.2f}%", ha='center', va='bottom')

plt.tight_layout()
plt.show()

```

In [ ]:

```
kmd_clusters_df.head()
```

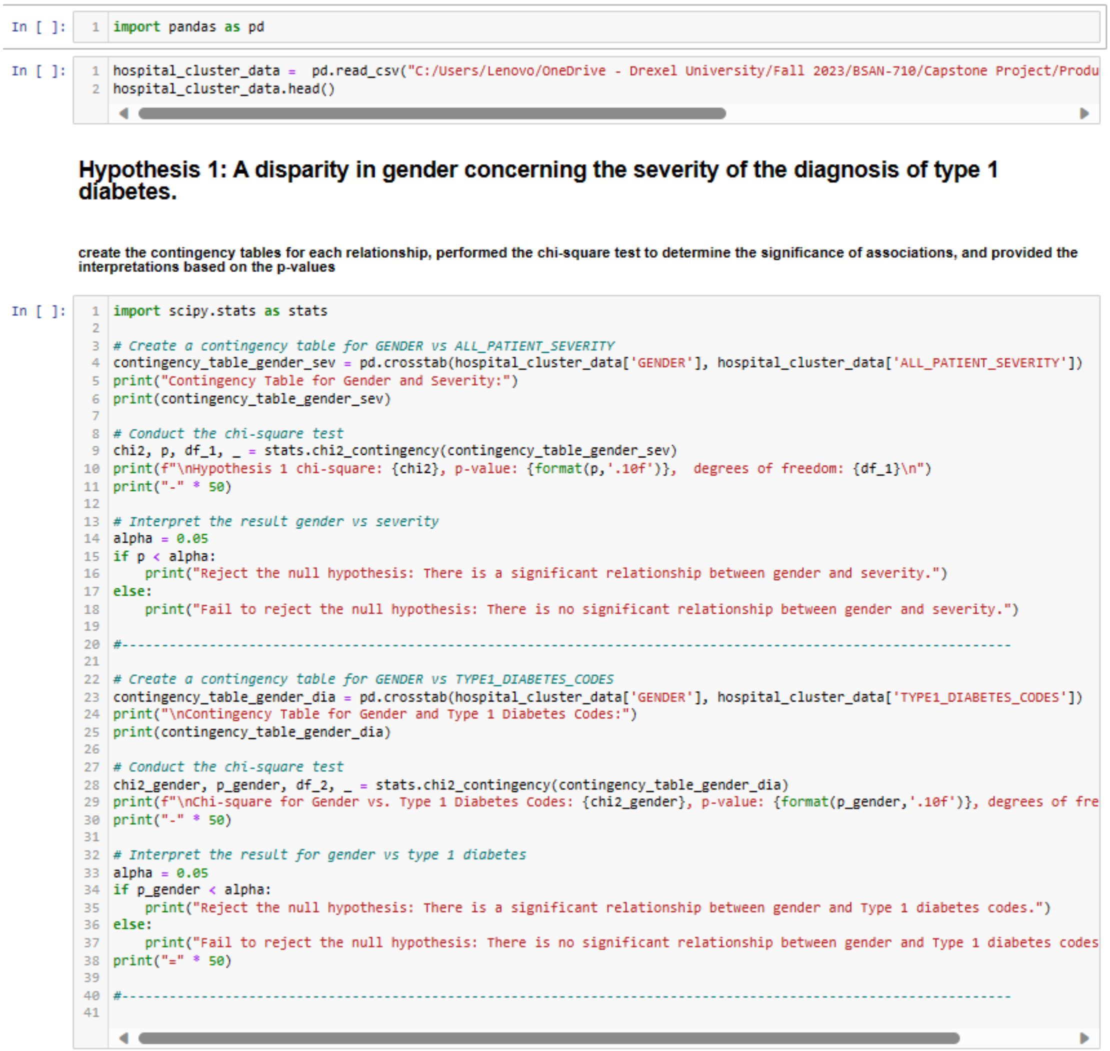
In [ ]:

```
# Heatmap of AGE_GROUP
crosstab = pd.crosstab(kmd_clusters_df['AGE_GROUP'], kmd_clusters_df['CLUSTERS'], normalize='index') * 100

# matrix with percentages
annot_array = crosstab.values
annotations = [[f'{val:.2f}%' for val in row] for row in annot_array]

plt.figure(figsize=(10, 6))
sns.heatmap(crosstab, annot=annotations, cmap="YlGnBu", fmt="")
plt.title("Distribution of Age Group across Clusters")
plt.show()

# Heatmap of GENDER
crosstab = pd.crosstab(kmd_clusters_df['GENDER'], kmd_clusters_df['CLUSTERS'], normalize='index') * 100

# matrix with percentages
annot_array = crosstab.values
annotations = [[f'{val:.2f}%' for val in row] for row in annot_array]

plt.figure(figsize=(10, 6))
sns.heatmap(crosstab, annot=annotations, cmap="YlGnBu", fmt="")
plt.title("Distribution of Gender across Clusters")
plt.show()

# Heatmap of RACE_DESC
crosstab = pd.crosstab(kmd_clusters_df['RACE_DESC'], kmd_clusters_df['CLUSTERS'], normalize='index') * 100

# matrix with percentages
annot_array = crosstab.values
annotations = [[f'{val:.2f}%' for val in row] for row in annot_array]

plt.figure(figsize=(10, 6))
sns.heatmap(crosstab, annot=annotations, cmap="YlGnBu", fmt="")
plt.title("Distribution of Race across Clusters")
plt.show()

# Heatmap of RACE_DESC
crosstab = pd.crosstab(kmd_clusters_df['HOSPITAL_REGION'], kmd_clusters_df['CLUSTERS'], normalize='index') * 100

# matrix with percentages
annot_array = crosstab.values
annotations = [[f'{val:.2f}%' for val in row] for row in annot_array]

plt.figure(figsize=(10, 6))
sns.heatmap(crosstab, annot=annotations, cmap="YlGnBu", fmt="")
plt.title("Distribution of Hospital Region across Clusters")
plt.show()

# Heatmap of Severity
crosstab = pd.crosstab(kmd_clusters_df['ALL_PATIENT_SEVERITY'], kmd_clusters_df['CLUSTERS'], normalize='index') * 100

# matrix with percentages
annot_array = crosstab.values
annotations = [[f'{val:.2f}%' for val in row] for row in annot_array]

plt.figure(figsize=(10, 6))
sns.heatmap(crosstab, annot=annotations, cmap="YlGnBu", fmt="")
plt.title("Distribution of Health Severity across Clusters")
plt.show()

# Heatmap of LOS
kmd_clusters_df['LOS'] = kmd_clusters_df['LOS'].astype(int)

```



```python
def categorize_los(los):
    if los <= 5:
        return "Short-term"
    elif 5 < los <= 15:
        return "Mid-term"
    else:
        return "Long-term"

kmd_clusters_df['LOS_Category'] = kmd_clusters_df['LOS'].apply(categorize_los)

# crosstab with the LOS_Category and Clusters
crosstab = pd.crosstab(kmd_clusters_df['LOS_Category'], kmd_clusters_df['CLUSTERS'])
# Convert the crosstab values to proportions
crosstab_percentage = (crosstab / crosstab.sum().sum()) * 100

# Generate annotations with percentage symbol
annotations = crosstab_percentage.applymap(lambda x: f"{x:.2f}%")

plt.figure(figsize=(10, 6))
sns.heatmap(crosstab_percentage, annot=annotations, cmap="YlGnBu", fmt="s")
plt.title("Distribution of LOS across Clusters in Percentages")
plt.xlabel("Cluster")
plt.ylabel("LOS Category")
plt.show()



# Heatmap of TYPE1 DIABETES
crosstab = pd.crosstab(kmd_clusters_df['TYPE1_DIABETES_CODES'], kmd_clusters_df['CLUSTERS'], normalize='index') * 100

# matrix with percentages
annot_array = crosstab.values
annotations = [[f'{val:.2f}%' for val in row] for row in annot_array]

plt.figure(figsize=(10, 6))
sns.heatmap(crosstab, annot=annotations, cmap="YlGnBu", fmt="")
plt.title("Distribution of Type1 Diabetes Diagnosis across Clusters")
plt.show()
```

- Chi-Square Testing



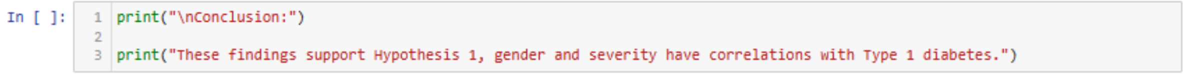
In [ ]:
```python
import pandas as pd
```

In [ ]:
```python
hospital_cluster_data =  pd.read_csv("C:/Users/Lenovo/OneDrive - Drexel University/Fall 2023/BSAN-710/Capstone Project/Produ
hospital_cluster_data.head()
```

## Hypothesis 1: A disparity in gender concerning the severity of the diagnosis of type 1 diabetes.

**create the contingency tables for each relationship, performed the chi-square test to determine the significance of associations, and provided the interpretations based on the p-values**

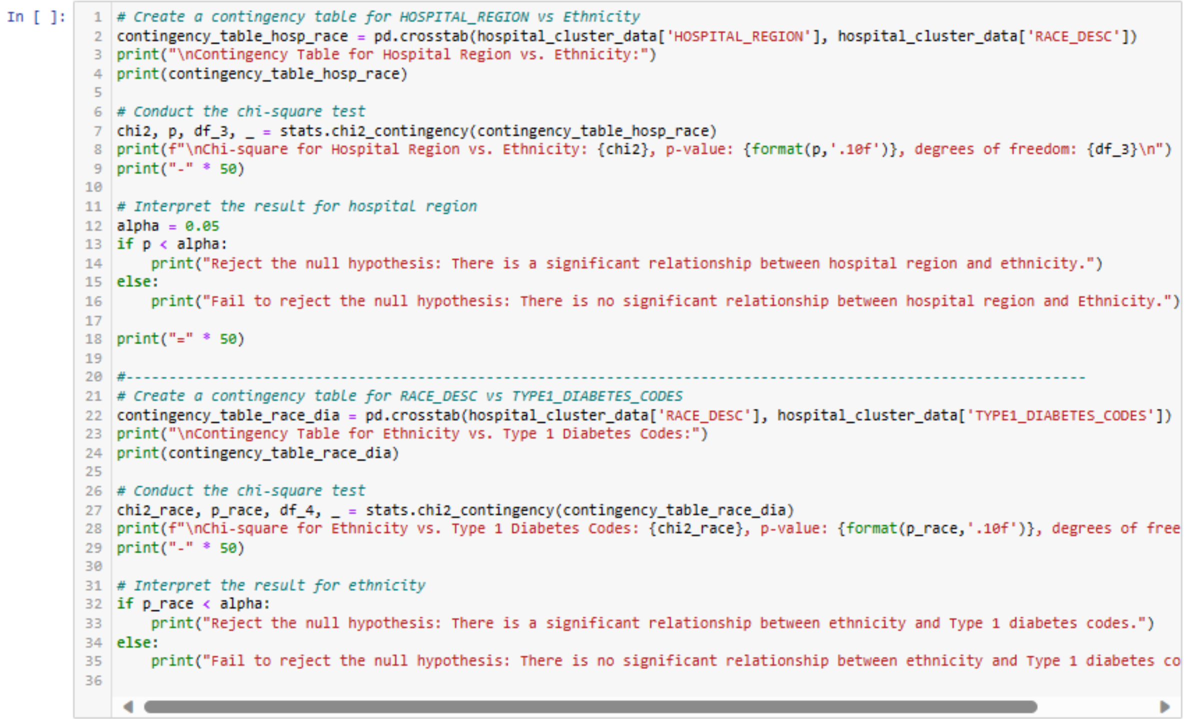
In [ ]:
```python
import scipy.stats as stats

# Create a contingency table for GENDER vs ALL_PATIENT_SEVERITY
contingency_table_gender_sev = pd.crosstab(hospital_cluster_data['GENDER'], hospital_cluster_data['ALL_PATIENT_SEVERITY'])
print("Contingency Table for Gender and Severity:")
print(contingency_table_gender_sev)

# Conduct the chi-square test
chi2, p, df_1, _ = stats.chi2_contingency(contingency_table_gender_sev)
print(f"\nHypothesis 1 chi-square: {chi2}, p-value: {format(p,'.10f')},  degrees of freedom: {df_1}\n")
print("-" * 50)

# Interpret the result gender vs severity
alpha = 0.05
if p < alpha:
    print("Reject the null hypothesis: There is a significant relationship between gender and severity.")
else:
    print("Fail to reject the null hypothesis: There is no significant relationship between gender and severity.")

#-----------------------------------------------------------------------------------------------------------

# Create a contingency table for GENDER vs TYPE1_DIABETES_CODES
contingency_table_gender_dia = pd.crosstab(hospital_cluster_data['GENDER'], hospital_cluster_data['TYPE1_DIABETES_CODES'])
print("\nContingency Table for Gender and Type 1 Diabetes Codes:")
print(contingency_table_gender_dia)

# Conduct the chi-square test
chi2_gender, p_gender, df_2, _ = stats.chi2_contingency(contingency_table_gender_dia)
print(f"\nChi-square for Gender vs. Type 1 Diabetes Codes: {chi2_gender}, p-value: {format(p_gender,'.10f')}, degrees of fre
print("-" * 50)

# Interpret the result for gender vs type 1 diabetes
alpha = 0.05
if p_gender < alpha:
    print("Reject the null hypothesis: There is a significant relationship between gender and Type 1 diabetes codes.")
else:
    print("Fail to reject the null hypothesis: There is no significant relationship between gender and Type 1 diabetes codes
print("=" * 50)

#-----------------------------------------------------------------------------------------------------------
```



In [ ]:

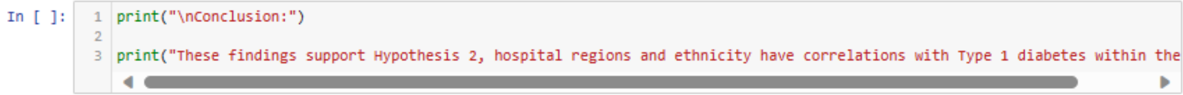
```python
print("\nConclusion:")

print("These findings support Hypothesis 1, gender and severity have correlations with Type 1 diabetes.")
```

## Hypothesis 2: Factors such as the regions of hospitals and ethnicity are found to have a correlation with Type 1 diabetes within the group.



In [ ]:

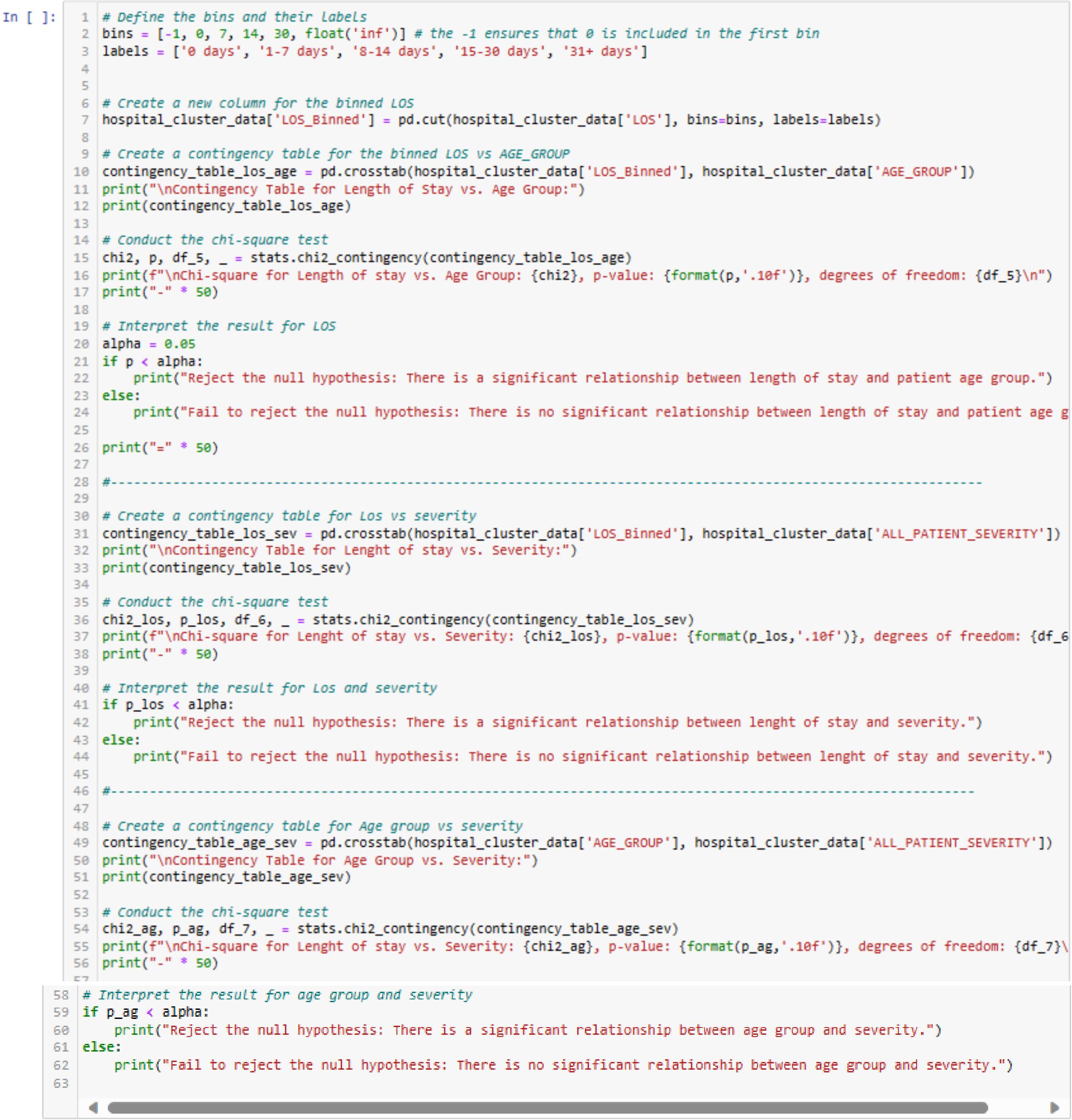
```python
# Create a contingency table for HOSPITAL_REGION vs Ethnicity
contingency_table_hosp_race = pd.crosstab(hospital_cluster_data['HOSPITAL_REGION'], hospital_cluster_data['RACE_DESC'])
print("\nContingency Table for Hospital Region vs. Ethnicity:")
print(contingency_table_hosp_race)

# Conduct the chi-square test
chi2, p, df_3, _ = stats.chi2_contingency(contingency_table_hosp_race)
print(f"\nChi-square for Hospital Region vs. Ethnicity: {chi2}, p-value: {format(p,'.10f')}, degrees of freedom: {df_3}\n")
print("-" * 50)

# Interpret the result for hospital region
alpha = 0.05
if p < alpha:
    print("Reject the null hypothesis: There is a significant relationship between hospital region and ethnicity.")
else:
    print("Fail to reject the null hypothesis: There is no significant relationship between hospital region and Ethnicity.")

print("=" * 50)

#-----------------------------------------------------------------------------------------------------------------
# Create a contingency table for RACE_DESC vs TYPE1_DIABETES_CODES
contingency_table_race_dia = pd.crosstab(hospital_cluster_data['RACE_DESC'], hospital_cluster_data['TYPE1_DIABETES_CODES'])
print("\nContingency Table for Ethnicity vs. Type 1 Diabetes Codes:")
print(contingency_table_race_dia)

# Conduct the chi-square test
chi2_race, p_race, df_4, _ = stats.chi2_contingency(contingency_table_race_dia)
print(f"\nChi-square for Ethnicity vs. Type 1 Diabetes Codes: {chi2_race}, p-value: {format(p_race,'.10f')}, degrees of free
print("-" * 50)

# Interpret the result for ethnicity
if p_race < alpha:
    print("Reject the null hypothesis: There is a significant relationship between ethnicity and Type 1 diabetes codes.")
else:
    print("Fail to reject the null hypothesis: There is no significant relationship between ethnicity and Type 1 diabetes co

```



In [ ]:

```python
print("\nConclusion:")

print("These findings support Hypothesis 2, hospital regions and ethnicity have correlations with Type 1 diabetes within the
```

### Hypothesis 3: Distribution of the length of a patient's stay in the hospital and severity of diabetes diagnoses among different age groups.



```
# Define the bins and their labels
bins = [-1, 0, 7, 14, 30, float('inf')] # the -1 ensures that 0 is included in the first bin
labels = ['0 days', '1-7 days', '8-14 days', '15-30 days', '31+ days']


# Create a new column for the binned LOS
hospital_cluster_data['LOS_Binned'] = pd.cut(hospital_cluster_data['LOS'], bins=bins, labels=labels)

# Create a contingency table for the binned LOS vs AGE_GROUP
contingency_table_los_age = pd.crosstab(hospital_cluster_data['LOS_Binned'], hospital_cluster_data['AGE_GROUP'])
print("\nContingency Table for Length of Stay vs. Age Group:")
print(contingency_table_los_age)

# Conduct the chi-square test
chi2, p, df_5, _ = stats.chi2_contingency(contingency_table_los_age)
print(f"\nChi-square for Length of stay vs. Age Group: {chi2}, p-value: {format(p,'.10f')}, degrees of freedom: {df_5}\n")
print("-" * 50)

# Interpret the result for LOS
alpha = 0.05
if p < alpha:
    print("Reject the null hypothesis: There is a significant relationship between length of stay and patient age group.")
else:
    print("Fail to reject the null hypothesis: There is no significant relationship between length of stay and patient age g

print("=" * 50)

#---------------------------------------------------------------------------------------------------------

# Create a contingency table for Los vs severity
contingency_table_los_sev = pd.crosstab(hospital_cluster_data['LOS_Binned'], hospital_cluster_data['ALL_PATIENT_SEVERITY'])
print("\nContingency Table for Lenght of stay vs. Severity:")
print(contingency_table_los_sev)

# Conduct the chi-square test
chi2_los, p_los, df_6, _ = stats.chi2_contingency(contingency_table_los_sev)
print(f"\nChi-square for Lenght of stay vs. Severity: {chi2_los}, p-value: {format(p_los,'.10f')}, degrees of freedom: {df_6
print("-" * 50)

# Interpret the result for Los and severity
if p_los < alpha:
    print("Reject the null hypothesis: There is a significant relationship between lenght of stay and severity.")
else:
    print("Fail to reject the null hypothesis: There is no significant relationship between lenght of stay and severity.")

#--------------------------------------------------------------------------------------------------------

# Create a contingency table for Age group vs severity
contingency_table_age_sev = pd.crosstab(hospital_cluster_data['AGE_GROUP'], hospital_cluster_data['ALL_PATIENT_SEVERITY'])
print("\nContingency Table for Age Group vs. Severity:")
print(contingency_table_age_sev)

# Conduct the chi-square test
chi2_ag, p_ag, df_7, _ = stats.chi2_contingency(contingency_table_age_sev)
print(f"\nChi-square for Lenght of stay vs. Severity: {chi2_ag}, p-value: {format(p_ag,'.10f')}, degrees of freedom: {df_7}\
print("-" * 50)

# Interpret the result for age group and severity
if p_ag < alpha:
    print("Reject the null hypothesis: There is a significant relationship between age group and severity.")
else:
    print("Fail to reject the null hypothesis: There is no significant relationship between age group and severity.")

```

## Asthma Initial Setup

Similar to the merging done for Diabetes, the datasets were combined together. Afterward, a basic summary of the dataset was created to determine the spread of the data. In addition, data cleaning and preprocessing were done prior to statistical testing and clustering analysis.

## Asthma Statistical Testing Using Chi-Squared Test

Similar to the statistical testing done for Diabetes, chi-squared test for statistical analysis, examining the relationships between categorical variables. The test generated a p-value and a Chi-Square value. A low p-value (<0.05) suggested significant associations, enabling us to make informed decisions about the validity of patterns within our dataset and test our initial hypothesis.

## Asthma K-Means Clustering

Applying K-Means clustering, we segmented the data into distinct groups based on similarities. This unsupervised machine learning technique allowed us to identify inherent patterns and groupings within the dataset.

For better results we created bins for variables like Age and Total Charges. The clustering algorithm divided the entire data for asthmatic patients into 5 clusters.

## Asthma K-Modes Clustering

We also employed K-Modes clustering, specifically designed for categorical data. This approach further enriched our understanding by revealing patterns in non-numeric variables, providing a holistic view of the dataset.

The clustering algorithm divided the entire data for asthmatic patients into 3 clusters.

## Asthma Codes: SQL and Jupyter Notebook

### SQL Queries:

```
--Query to create separate table for respiratory diseases based on diagnosis code:
CREATE TABLE capstone2.RESPIRATORY_DISEASES AS

SELECT HOSP_KID
            ,RECNUM
            ,DXCCSR_RSP001
            ,DXCCSR_RSP002
            ,DXCCSR_RSP003
            ,DXCCSR_RSP004
            ,DXCCSR_RSP005
            ,DXCCSR_RSP006
            ,DXCCSR_RSP007
            ,DXCCSR_RSP008
            ,DXCCSR_RSP009
            ,DXCCSR_RSP010
            ,DXCCSR_RSP011
            ,DXCCSR_RSP012
            ,DXCCSR_RSP013
            ,DXCCSR_RSP014
            ,DXCCSR_RSP015
            ,DXCCSR_RSP016
            ,DXCCSR_RSP017
FROM `capstone - 400517. capstone2.kid_GPRS`
```

---

```
--QUERY FOR asthma cases by Gender
SELECT FEMALE
            ,count(*)
FROM `capstone - 400517. capstone2.table_core`
WHERE Asthma <> '0'
            AND Asthma IS NOT NULL
GROUP BY 1
```

---

```
--QUERY FOR asthma cases by INCOME
SELECT ZIPINC_QRTL
            ,count(*)
FROM `capstone - 400517. capstone2.table_core`
WHERE Asthma <> '0'
GROUP BY 1
```

---

```
--QUERY FOR asthma cases by race
SELECT RACE
            ,count(*)
FROM `capstone - 400517. capstone2.table_core`
--WHERE Asthma <> '0'
GROUP BY 1
```

---

```
--query for asthma and infectious diseases
SELECT Asthma
            ,INF_All
            ,COUNT(*) AS INFANDASTHMA
FROM `capstone - 400517. capstone2.table_core`
WHERE Asthma <> '0'
            AND INF_All <> 0
GROUP BY 1
            ,2
```

---

```
--COMORBIDITY ANALYSIS FOR ASTHMA AND OTHER DISEASES
SELECT RECNUM
FROM `capstone - 400517. capstone2.table_core`
WHERE (
                        Blood_All <> 0
                        OR MAL_All <> 0
                        OR MBD_ALL <> 0
                        OR END_ALL <> 0
                        OR INF_All <> 0
```

```
        OR DIG_ALL <> 0
        OR NVS_ALL <> 0
        OR PRG_All <> 0
        OR CIR_All <> 0
        OR SKN_All <> 0
        OR MUS_All <> 0
        OR Tumor_All <> 0
        OR EYE_All <> 0
        OR EAR_All <> 0
        )
    AND Asthma <> '0'
```

```
--Data cleaning in gender
DELETE
FROM `capstone - 400517. capstone2.table_core`
WHERE FEMALE IN (
        'A'
        ,'C'
        ,'nan'
        );
```

```
--query to get total charges by payer
SELECT
    --RECNUM
    Pay1
    ,sum(TotalChargesInteger)
FROM `capstone - 400517. capstone2.table_core`
WHERE Asthma <> '0'
    AND (
        Blood_All = 0
        AND MAL_All = 0
        AND MBD_ALL = 0
        AND END_ALL = 0
        AND INF_All = 0
        AND DIG_ALL = 0
        AND NVS_ALL = 0
        AND PRG_All = 0
        AND CIR_All = 0
        AND SKN_All = 0
        AND MUS_All = 0
        AND Tumor_All = 0
        AND EYE_All = 0
        AND EAR_All = 0
        )
    AND PAY1 IN (
        '1'
        ,'2'
        )
--and race <> 'nan'
--and Race not like '%nan'
GROUP BY 1
```

```
--Query to check count of each endocrine disease
SELECT SUM(CASE
            WHEN CAST(DXCCSR_END001 AS INT64) = 1
```

```
                              THEN 1
                    ELSE 0
                    END) AS Count_1
,SUM(CASE
                    WHEN CAST(DXCCSR_END002 AS INT64) = 1
                              THEN 1
                    ELSE 0
                    END) AS Count_2
,SUM(CASE
                    WHEN CAST(DXCCSR_END003 AS INT64) = 1
                              THEN 1
                    ELSE 0
                    END) AS Count_3
,SUM(CASE
                    WHEN CAST(DXCCSR_END004 AS INT64) = 1
                              THEN 1
                    ELSE 0
                    END) AS Count_4
,SUM(CASE
                    WHEN CAST(DXCCSR_END006 AS INT64) = 1
                              THEN 1
                    ELSE 0
                    END) AS Count_6
,SUM(CASE
                    WHEN CAST(DXCCSR_END007 AS INT64) = 1
                              THEN 1
                    ELSE 0
                    END) AS Count_7
,SUM(CASE
                    WHEN CAST(DXCCSR_END008 AS INT64) = 1
                              THEN 1
                    ELSE 0
                    END) AS Count_8
,SUM(CASE
                    WHEN CAST(DXCCSR_END009 AS INT64) = 1
                              THEN 1
                    ELSE 0
                    END) AS Count_9
,SUM(CASE
                    WHEN CAST(DXCCSR_END010 AS INT64) = 1
                              THEN 1
                    ELSE 0
                    END) AS Count_10
,SUM(CASE
                    WHEN CAST(DXCCSR_END011 AS INT64) = 1
                              THEN 1
                    ELSE 0
                    END) AS Count_11
,SUM(CASE
                    WHEN CAST(DXCCSR_END012 AS INT64) = 1
                              THEN 1
                    ELSE 0
                    END) AS Count_12
,SUM(CASE
                    WHEN CAST(DXCCSR_END013 AS INT64) = 1
                              THEN 1
                    ELSE 0
                    END) AS Count_13
,SUM(CASE
                    WHEN CAST(DXCCSR_END014 AS INT64) = 1
                              THEN 1
                    ELSE 0
                    END) AS Count_14
            ,SUM(CASE
                    WHEN CAST(DXCCSR_END015 AS INT64) = 1
                              THEN 1
                    ELSE 0
                    END) AS Count_15
            ,SUM(CASE
                    WHEN CAST(DXCCSR_END016 AS INT64) = 1
                              THEN 1
                    ELSE 0
                    END) AS Count_16
            ,SUM(CASE
                    WHEN CAST(DXCCSR_END017 AS INT64) = 1
                              THEN 1
                    ELSE 0
                    END) AS Count_17
            ,
        FROM `capstone - 400517. capstone2.ENDOCRINE_DISEASES`
```

```
--Query to analyze asthma and length of stay
SELECT RECNUM
    ,LOS
FROM `capstone - 400517. capstone2.table_core`
WHERE (
        Blood_All <> 0
        OR MAL_All <> 0
        OR MBD_ALL <> 0
        OR END_ALL <> 0
        OR INF_All <> 0
        OR DIG_ALL <> 0
        OR NVS_ALL <> 0
        OR PRG_All <> 0
        OR CIR_All <> 0
        OR SKN_All <> 0
        OR MUS_All <> 0
        OR Tumor_All <> 0
        OR EYE_All <> 0
        OR EAR_All <> 0
        )
    AND Asthma <> '0'
    AND Asthma IS NOT NULL
```

```
--Query to get count of individual infectious diseases
SELECT SUM(CASE
            WHEN CAST(DXCCSR_INF001 AS INT64) <> 0
                THEN 1
            ELSE 0
            END) AS Count_INF001
    ,SUM(CASE
            WHEN CAST(DXCCSR_INF002 AS INT64) <> 0
                THEN 1
            ELSE 0
            END) AS Count_INF002
    ,SUM(CASE
            WHEN CAST(DXCCSR_INF003 AS INT64) <> 0
                THEN 1
            ELSE 0
            END) AS Count_INF003
```

```
        ,SUM(CASE
                        WHEN CAST(DXCCSR_INF004 AS INT64) <> 0
                            THEN 1
                        ELSE 0
                        END) AS Count_INF004
        ,SUM(CASE
                        WHEN CAST(DXCCSR_INF005 AS INT64) <> 0
                            THEN 1
                        ELSE 0
                        END) AS Count_INF005
        ,SUM(CASE
                        WHEN CAST(DXCCSR_INF006 AS INT64) <> 0
                            THEN 1
                        ELSE 0
                        END) AS Count_INF006
        ,SUM(CASE
                        WHEN CAST(DXCCSR_INF007 AS INT64) <> 0
                            THEN 1
                        ELSE 0
                        END) AS Count_INF007
        ,SUM(CASE
                        WHEN CAST(DXCCSR_INF008 AS INT64) <> 0
                            THEN 1
                        ELSE 0
                        END) AS Count_INF008
        ,SUM(CASE
                        WHEN CAST(DXCCSR_INF009 AS INT64) <> 0
                            THEN 1
                        ELSE 0
                        END) AS Count_INF009
        ,SUM(CASE
                        WHEN CAST(DXCCSR_INF010 AS INT64) <> 0
                            THEN 1
                        ELSE 0
                        END) AS Count_INF010
        ,SUM(CASE
                        WHEN CAST(DXCCSR_INF011 AS INT64) <> 0
                            THEN 1
                        ELSE 0
                        END) AS Count_INF011
        ,SUM(CASE
                        WHEN CAST(DXCCSR_INF012 AS INT64) <> 0
                            THEN 1
                        ELSE 0
                        END) AS Count_INF012
FROM `capstone - 400517. capstone2.Infectious_DISEASES`
```

```
--Query to clean total charges col.
--ALTER TABLE `capstone-400517.capstone2.table_core`
--ADD COLUMN TotalChargesInteger INT64;
-- Copy data from the existing column to the new column
UPDATE `capstone - 400517. capstone2.table_core`
SET TotalChargesInteger = CASE
                WHEN TOTCHG = 'nan'
                    THEN NULL -- or set to a default value
                WHEN TOTCHG = 'A'
                    THEN NULL
                WHEN TOTCHG = 'C'
                    THEN NULL
                WHEN TOTCHG = 'TOTCHG'
```

```
                        THEN NULL
                ELSE CAST(TOTCHG AS INT64)
                END
WHERE 1 = 1
```

---

```
--Query to get LOS AND TOTAL CHARGE FOR PATIENTS WITH ASTHMA AND OTHER DISEASES
SELECT t1.DXCCSR_RSP009
        ,COUNT(*)
FROM (
        SELECT RECNUM
                ,DXCCSR_Default_DX1
                ,DXCCSR_RSP009
        FROM `capstone - 400517. capstone2.kid_GPRS` A11
        WHERE A11.DXCCSR_Default_DX1 = 'RSP009'
        ) t1
GROUP BY 1
```

---

```
--query to create separate table for CongenitalMalfunction
CREATE TABLE capstone2.CongenitalMalfunction_DISEASES AS

SELECT HOSP_KID
        ,RECNUM
        ,DXCCSR_MAL001
        ,DXCCSR_MAL002
        ,DXCCSR_MAL003
        ,DXCCSR_MAL004
        ,DXCCSR_MAL005
        ,DXCCSR_MAL006
        ,DXCCSR_MAL007
        ,DXCCSR_MAL008
        ,DXCCSR_MAL009
        ,DXCCSR_MAL010
FROM `capstone - 400517. capstone2.kid_GPRS`
```

---

```
--query to make new table for infectious diseases
CREATE TABLE capstone2.Infectious _DISEASES AS

SELECT HOSP_KID
        ,RECNUM
        ,DXCCSR_INF001
        ,DXCCSR_INF002
        ,DXCCSR_INF003
        ,DXCCSR_INF004
        ,DXCCSR_INF005
        ,DXCCSR_INF006
        ,DXCCSR_INF007
        ,DXCCSR_INF008
        ,DXCCSR_INF009
        ,DXCCSR_INF010
        ,DXCCSR_INF011
        ,DXCCSR_INF012
FROM `capstone - 400517. capstone2.kid_GPRS`
```

---

```
--Query to make range for LOS
UPDATE capstone2.table_core
SET LOS_Range = CASE
                WHEN LOS BETWEEN '0'
```

```
                                AND '1'
                        THEN '0-1'
                WHEN LOS BETWEEN '2'
                                AND '3'
                        THEN '2-3'
                WHEN LOS BETWEEN '4'
                                AND '9'
                        THEN '4-9'
                WHEN LOS > '10'
                        THEN '10+'
                END
WHERE 1 = 1
```

---

```
--query to make range for total charges
UPDATE capstone2.table_core
SET TotalChargeRange = CASE
                WHEN TotalChargesInteger BETWEEN 0
                                AND 12000
                        THEN 1
                WHEN TotalChargesInteger BETWEEN 12001
                                AND 20000
                        THEN 2
                WHEN TotalChargesInteger BETWEEN 20001
                                AND 40000
                        THEN 3
                WHEN TotalChargesInteger BETWEEN 40001
                                AND 80000
                        THEN 4
                WHEN TotalChargesInteger > 80001
                        THEN 5
                END
WHERE 1 = 1
```

---

```
# Python script for calculating Chi-Squared value and p value for asthma by gender
#import library
from scipy.stats import chi2_contingency
import matplotlib.pyplot as plt
import numpy as np

observed_data = [[101427, 1333965], [103168, 1248246]]

# Perform chi-square test
chi2, p, _, _ = chi2_contingency(observed_data)


#print p value
print(f"P-value: {p}")

print(f"Chi-squared value: {chi2}")
```

---

```
#Python script to check chi-square and P value for asthma by hospital region
#import library
from scipy.stats import chi2_contingency
import matplotlib.pyplot as plt
import numpy as np

observed_data = [[46993, 463696], [50325, 645687], [81598, 1116237], [25679, 356591]]
```

```
# Perform chi-square test
chi2, p, dof, expected = chi2_contingency(observed_data)
#print p value
print(f"P-value: {p}")
```

```
# Python script for chi-sqared test for income and asthma
#import library
from scipy.stats import chi2_contingency
import matplotlib.pyplot as plt
import numpy as np
observed_data = [[74825, 804022], [48867, 634510], [45054, 617432], [33955, 499721]]
# Perform chi-square test
chi2, p, _, _ = chi2_contingency(observed_data)
#print p value
print(f"P-value: {p}")

print(f"Chi-squared value: {chi2}")
```

```
# Python script for chi-sqared test: asthma with race
#import library
from scipy.stats import chi2_contingency
import matplotlib.pyplot as plt
import numpy as np
observed_data = [[78861, 1227805], [60569, 434845], [38607, 458916], [4366, 94119], [1550, 18515], [10290, 160175]]
# Perform chi-square test
chi2, p, _, _ = chi2_contingency(observed_data)
#print p value
print(f"P-value: {p}")
```

# Code for clustering:

```
# Filtering only for RSP009
results = job.to_dataframe()
results = results[results['DXCCSR_Default_DX1']== 'RSP009']
```

```
# Import numpy to handle NaN values
import numpy as np

# Replace all instances of 'nan' with NaN in the results DataFrame
results.replace('nan', np.nan, inplace=True)
```

```
# Drop NA values from table
results.dropna(inplace=True)
results.replace('A', np.nan, inplace=True)
```

```
# Drop NA values form table
results.dropna(inplace=True)
```

```
# check if we have NA values among unique values

cols = results.columns
for i in cols:

  print(i, results[i].unique())
```

```
#Converting all values into numeric so that we can run ML algorithms

import pandas as pd

# Convert all columns to numeric, coercing errors to NaN
results = results.apply(pd.to_numeric, errors='coerce')


# Convert to Int64
results = results.astype('Int64')
```

```
#Checking if the values converted into numeric

cols = results.columns
for i in cols:

  print(i, results[i].unique())
```

```
#Converting all values into numeric so that we can run ML algorithms

import pandas as pd

# Convert all columns to numeric, coercing errors to NaN
results = results.apply(pd.to_numeric, errors='coerce')


# Convert to Int64
results = results.astype('Int64')
```

```
#Checking if the values converted into numeric

cols = results.columns
for i in cols:
 print(i, results[i].unique())
```

```
# Separating the features that  we need for K-Mode algorthm

df_kmode = results[['AGE','FEMALE', 'RACE', 'HOSP_REGION', 'ZIPINC_QRTL', 'LOS', 'TOTCHG',
'PAY1']]
```

```
# Creating bins for Length of stay column

## labels = ['0-1', '2-3', '4-9', '10 and over']


bins = [-1, 1, 3, 9, float('inf')]  # Using -1 as the lower bound to include 0
labels = [1, 2, 3, 4]

# Create a new column with the binned categories
df_kmode.loc[:, 'LOS'] = pd.cut(df_kmode['LOS'], bins=bins, labels=labels)
```

```
# Creating bins for "Total charge" feature

## labels = ['0-12000', '12,000-20,000' '20,000-40,000', '40,000-80,000', '80,000+']


bins = [-1, 12000, 20000, 40000, 80000, float('inf')]  # Using -1 as the lower bound to include 0
labels = [1, 2, 3, 4, 5]

# Create a new column with the binned categories
df_kmode.loc[:, 'TOTCHG'] = pd.cut(df_kmode['TOTCHG'], bins=bins, labels=labels)
```

```
# Creating bins for "AGE" feature of the model

## labels = ['0-1', '1-10' '11-20']


bins = [-1, 1, 10, 21]  # Using -1 as the lower bound to include 0
labels = [1, 2, 3]

# Create a new column with the binned categories
df_kmode.loc[:, 'AGE'] = pd.cut(df_kmode['AGE'], bins=bins, labels=labels)
```

```
# Install K-Modes ML library

!pip install kmodes
```

```
# Import necessary libraries from KModes and create clusters
from kmodes.kmodes import KModes
```

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder

encoder = OneHotEncoder(sparse=False)
df_encoded = encoder.fit_transform(df_kmode)

km = KModes(n_clusters=3, init='Huang', n_init=5, verbose=1)
```

```
#Check of the clusters created in vector format

df_encoded
```

```
# Fit the cluster

clusters = km.fit_predict(df_encoded)
```

```
# Checking the optimal number of clusters using Elbow method

import matplotlib.pyplot as plt
from kmodes.kmodes import KModes

costs = []
for num_clusters in range(1, 10):
    km = KModes(n_clusters=num_clusters, init='Huang', n_init=5, verbose=0)
    km.fit(df_encoded)
    costs.append(km.cost_)

plt.plot(range(1, 10), costs, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Cost')
plt.title('Elbow Method For Optimal Number of Clusters')
plt.show()
```

```
# Join the cluster results to the original table

df_kmode['Cluster'] = clusters
```

```
# Using dataframe df_kmode: save to csv file for future analysis and visualization

df_kmode.to_csv('df_kmode_Asthma_primary_3.csv', index=False)
```

```
#Importing Library for kMeans
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

```
#Scaling the dataset
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df_asthma)
```

```
# Elbow Method
inertia = []
for i in range(1, 11):
   kmeans = KMeans(n_clusters=i, random_state=0)
   kmeans.fit(scaled_features)
   inertia.append(kmeans.inertia_)

plt.plot(range(1, 11), inertia)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.show()
```

```
# Initialize KMeans with k clusters
kmeans = KMeans(n_clusters=5, random_state=0)

# Fit the model
kmeans.fit(scaled_features)

# Get the cluster labels for each data point
labels = kmeans.labels_
```

```
# Add the cluster labels as a new column in the original DataFrame
df_asthma['Cluster'] = labels


#Export result to CSV
df_asthma.to_csv("/df_cluster_new.csv")
```

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