Problem 3: Analysing Prescription Patterns Using Clustering

Objectives:

Identify Prescription Patterns:

- Discover the different patterns in which "Target Drug" is prescribed or administered to patients.
- Patterns could include the timing and frequency of prescriptions.

Cluster Analysis:

- Utilize clustering or other unsupervised techniques to group patients or prescriptions based on similar patterns.
- Identify dominant clusters that represent the most common prescription patterns.

Visualize Patterns Over Time:

- Create visualizations to represent the prescription patterns.
- Plot the patterns with and the number of prescriptions.

The extraction of dominant prescription patterns from the data is essential for gaining deeper insights.

- A drug is typically given to a patient in specific patterns or at regular time intervals.
- For instance, Chemotherapy, a drug treatment for Cancer, is generally administered to patients at 3-4 week intervals. In other words, patients receive the drug every 3-4 weeks.
- Similarly, the 'Target Drug' is also prescribed or administered in certain patterns. We aim to analyse the patterns in which the 'Target Drug' is prescribed or administered to patients. It's possible that the 'Target Drug' is administered or prescribed in several patterns.

PHASE 1: Analysis to extract the dominant patterns in the data using clustering or other unsupervised techniques.

- Firstly, we will create a 'month' column in the data frame. This will be used in the clustering process and in creating a pivot table, as shown below.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```
✓ 0.0s

```
df_train = pd.read_parquet("train.parquet")  # loading parquet file
```
✓ 1.7s

```
# only who are all taking target drug
df_pos = df_train[df_train['Incident'] == 'TARGET DRUG']

#removing space in the TARGET_DRUG which  create problems in model building
df_train['Incident'] = df_train['Incident'].replace('TARGET DRUG', 'TARGET_DRUG')
```
✓ 0.4s

```
# converting dates to month
df_pos['Month'] = pd.to_datetime(df_pos['Date']).dt.to_period('M')
```
✓ 0.0s

```
# Creating a pivot table with patient - month prescription counts
pivot_table = pd.pivot_table(df_pos, index='Patient-Uid', columns='Month', values='Incident', aggfunc='count', fill_value=0)
```
✓ 0.0s

```
pivot_table.head()
```
✓ 0.0s

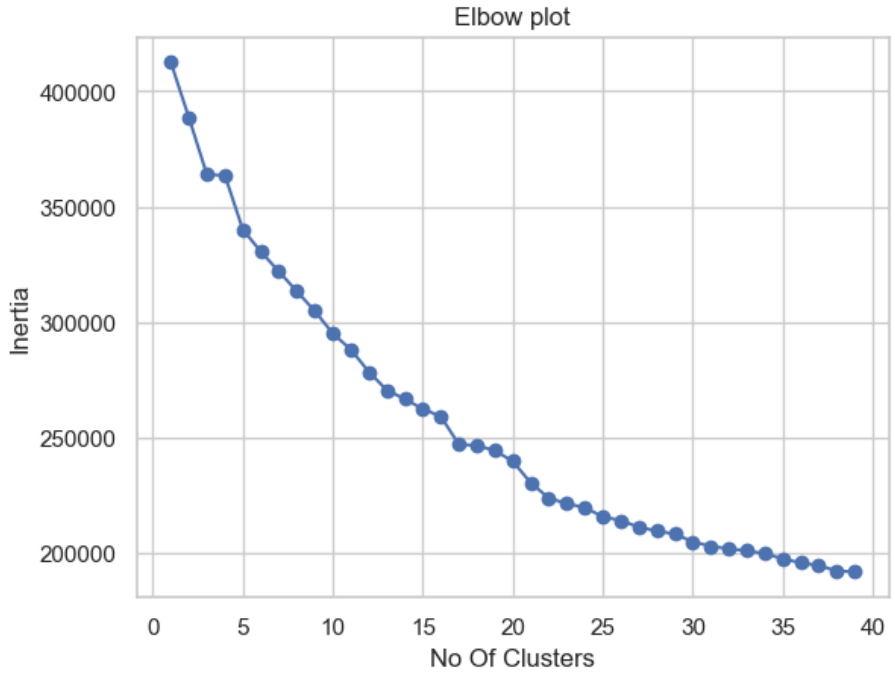
Month	2017-02	2017-03	2017-04	2017-05	2017-06	2017-07	2017-08	2017-09	2017-10	2017-11	2017-12	2018-01	2018-02	2018-03	2018-04	2018-05	2018-06	2018-07	2018-08	2018-09	2018-10
Patient-Uid																					
a0e9c384-1c7c-11ec-81a0-16262ee38c7f	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a0e9c3b3-1c7c-11ec-ae8e-16262ee38c7f	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	2	0	0
a0e9c3e3-1c7c-11ec-a8b9-16262ee38c7f	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a0e9c414-1c7c-11ec-889a-16262ee38c7f	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	2	0	0
a0e9c443-1c7c-11ec-9eb0-16262ee38c7f	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- Now, we proceed to apply K-means clustering. After normalizing the data, we aim to identify the optimal number of clusters.

```
# Determining the optimal number of clusters using elbow-method
uniform_line = []
for n_clusters in range(1, 40):
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    kmeans.fit(scaled_data)
    uniform_line.append(kmeans.inertia_)

# Plotting the elbow method to find the optimal number of clusters
plt.plot(range(1, 40), uniform_line)
plt.scatter(range(1, 40), uniform_line)
plt.xlabel('No Of Clusters')
plt.ylabel('Inertia')
plt.title('Elbow plot')
plt.show()
```

- We observe that the curve bends at the number of clusters equal to 3 and again at 17. Therefore, we will examine the patterns for both these variations.



PHASE 2:

- We will visualize the prescription patterns with the time (month) on the X-axis and prescriptions on the Y-axis for each of the patterns we are able to extract.

```
# Choosing the optimal number of clusters and applying K-Means clustering
no_of_clusters = 17  # can change 'no_of_clusters' as per need

kmeans = KMeans(n_clusters=no_of_clusters, random_state=42)# Applying K-Means clustering
patient_cluster_labels = kmeans.fit_predict(scaled_data)

# Creating dictionary to map patient IDs to cluster labels
patient_to_cluster = dict(zip(pivot_table.index, patient_cluster_labels))

# Adding cluster labels to the original DataFrame
df_pos['Cluster'] = df_pos['Patient-Uid'].map(patient_to_cluster)
```

✓ 0.4s

```
sns.set(style='whitegrid')
plt.figure(figsize=(25, 6))

# Looping through each cluster to create line plots
for clus in range(no_of_clusters):
    cluster_data = df_pos[df_pos['Cluster'] == clus] # Filtering the data for the current cluster
    prescription_counts = cluster_data.groupby('Month')['Incident'].count() # Plotting the prescription counts for the current cluster, with a label
    prescription_counts.plot(label=f'Cluster {clus}') # Plotting the prescription counts for the current cluster, with a label

plt.xlabel('Months')
plt.ylabel('Prescriptions')
plt.title('Different Clusters Prescription pattern')
plt.legend()
plt.show()
```

- This demonstrates that different patterns exist for different groups of patients due to various medical reasons. This indicates that the given data contains valuable information.