A

1)

```
Can our customers be grouped into separate categories using k-means clustering so we may target them
with different advertising and promotional campaigns?
```

2

```
One goal would be to define at least two groups of customers from the churn data set using k-means
clustering.
```

B

1)

```
 K-means clustering analyzes the data set by partitioning a data set into a specified number of
clusters. Cluster centers are randomly chosen from points in the data set. All other data points are
then assigned to the cluster that is closest to it. Cluster centers are recalculated using the means
of the data points assigned to it. Data points are assigned to the clusters again. This process can
repeat many times. An expected outcome is that all data points will be assigned to a cluster and the
clusters will have minimum inertia when the algorithm has finished.
```

2)

```
 One assumption of k-means clustering is that the created clusters are spherical.
```

3)

numpy

```
 This is used for working with numpy arrays that are returned from the scaling function.
```

matplotlib.pyplot

```
 This is used for visualizing inertia using the elbow method to find the optimal number of clusters.
```

sklearn.cluster import KMeans

```
This is the actual clustering algorithm that creates the model to cluster our data.
```

from sklearn.preprocessing import StandardScaler

```
This will be used to standardize the continuous variables.
```

C

1)

```
 One data preprocessing goal is to standardize the variables before clustering them.
```

2) The inital data set variables will be:

```
Age continuous,
Income continuous,
Outage_sec_perweek continuous,
MonthlyCharge continuous,
Bandwidth_GB_year continuous,
Contacts continuous,
Yearly_equip_failure continuous,
Tenure continuous
```

In [ ]:

```

```

3)

prepare data

read in data and drop index column.

In [1]:

```
import pandas as pd
# Assuming your CSV file is named 'data.csv', adjust the file path as needed
file_path = '/home/dj/skewl/D212/1/churn_clean.csv'
pd.set_option('display.max_columns', None)
# Read the data from the CSV file into a DataFrame
df = pd.read_csv(file_path)
#drop index column
df = df.loc[:, ~df.columns.str.contains('Unnamed')]
```

check for missing values.

In [2]:
```
# Identify missing values using isna() method
missing_values = df.isna().sum()
# Print DataFrame with True for missing values and False for non-missing values
print(missing_values)
# no missing values.
```

```
CaseOrder               0
Customer_id             0
Interaction             0
UID                     0
City                    0
State                   0
County                  0
Zip                     0
Lat                     0
Lng                     0
Population              0
Area                    0
TimeZone                0
Job                     0
Children                0
Age                     0
Income                  0
Marital                 0
Gender                  0
Churn                   0
Outage_sec_perweek      0
Email                   0
Contacts                0
Yearly_equip_failure    0
Techie                  0
Contract                0
Port_modem              0
Tablet                  0
InternetService         0
Phone                   0
Multiple                0
OnlineSecurity          0
OnlineBackup            0
DeviceProtection        0
TechSupport             0
StreamingTV             0
StreamingMovies         0
PaperlessBilling        0
PaymentMethod           0
Tenure                  0
MonthlyCharge           0
Bandwidth_GB_Year       0
Item1                   0
Item2                   0
Item3                   0
Item4                   0
Item5                   0
Item6                   0
```

```
Item7                  0
Item8                  0
dtype: int64
```

separate continuous from categorical variables.

In [3]:
```
# separate continuous variables
dfcon = df[['Age','Income','Bandwidth_GB_Year','MonthlyCharge','Outage_sec_perweek','Contacts','Yearly_equip_failure','
```

Standardize continuous variables. Write prepared data to file.

In [4]:
```
#standardize data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data = df
df = dfcon
# scale the data frame
df = scaler.fit_transform(df)
#write the prepared data to .csv file
pd.DataFrame(df).to_csv('prepared-data.csv', index=False)
```

```
/usr/lib/python3/dist-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.17.3 and <1.25.0 is required for
this version of SciPy (detected version 1.26.4
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

D

1)

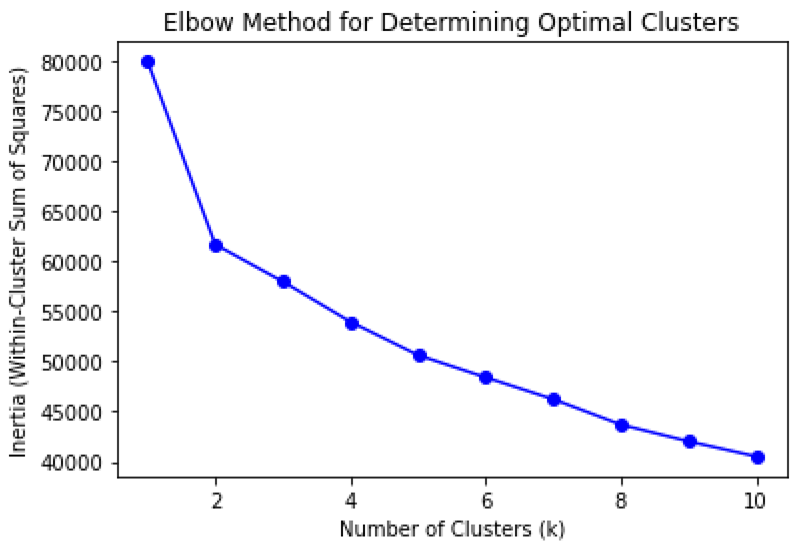
    I determined that 2 is the optimal number of clusters using the elbow method. This means that adding
    more than 2 clusters does not significantly decrease the inertia or within cluster sum of squares
    variance within the clusters.

2)

Code to plot the inertia of the clusters using the elbow method.

In [5]:
```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Define a range of cluster numbers to test
k_values = range(1, 11)  # Test cluster numbers from 1 to 10
```

```
inertiaArray = []

# Calculate inertia for each `k` value
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(df)
    inertiaArray.append(kmeans.inertia_)

# Plot inertia to find the "elbow"
plt.plot(k_values, inertiaArray, 'bo-')  # 'bo-' indicates blue circles with lines
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Inertia (Within-Cluster Sum of Squares)")
plt.title("Elbow Method for Determining Optimal Clusters")
plt.show()


## create clusters
kmeans = KMeans(n_clusters=2)
kmeans.fit(df)
```



Out[5]:

```
▾ KMeans
KMeans(n_clusters=2)
```

Inertia

In [6]:

```
inertia = kmeans.inertia_
print(inertia)
```

```
61704.925171052455
```

Silhouette Score

In [7]:
```python
from sklearn.metrics import silhouette_score

silhouette_score = silhouette_score(df, kmeans.labels_)
print(silhouette_score)
```

```
0.22818537888812102
```

Davies-Bouldin Index

In [8]:
```python
from sklearn.metrics import davies_bouldin_score

davies_bouldin_score = davies_bouldin_score(df, kmeans.labels_)
print(davies_bouldin_score)
```

```
1.7580509340625434
```

Calinski-Harabasz Index (Variance Ratio Criterion):

In [9]:
```python
from sklearn.metrics import calinski_harabasz_score

calinski_harabasz_score = calinski_harabasz_score(df, kmeans.labels_)
print(calinski_harabasz_score)
```

```
2964.336438829804
```

Print cluster stats

In [10]:
```python
data['Cluster'] = kmeans.labels_
for cluster_label in data['Cluster'].unique():
    # Subset data for the current cluster
    cluster_data = data[data['Cluster'] == cluster_label]

    # Compute cluster statistics
    cluster_stats = cluster_data.describe()

    # Print cluster statistics
    print(f"Cluster {cluster_label} Statistics:")
    print(cluster_stats)
```

```
Cluster 1 Statistics:
         CaseOrder           Zip           Lat           Lng    Population  \
count  5001.000000   5001.000000  5001.000000  5001.000000   5001.000000   
mean   2502.953409  49203.606879    38.834079   -90.829257   9756.015597   
std    1447.648689  27695.206694     5.476982    15.263438  14262.714108   
min       1.000000    601.000000    17.966120  -171.688150      0.000000   
25%    1251.000000  26222.000000    35.458930   -97.156460    722.000000   
50%    2502.000000  48836.000000    39.500680   -87.963530   2889.000000   
75%    3752.000000  71969.000000    42.120990   -80.003610  13489.000000   
max    8572.000000  99927.000000    70.640660   -65.943130  98660.000000   

          Children          Age         Income  Outage_sec_perweek  \
count  5001.000000  5001.000000    5001.000000         5001.000000   
mean      2.095381    52.677465   39737.006721            9.992615   
std       2.154507    20.698052   28029.785892            2.977881   
min       0.000000    18.000000     348.670000            0.120058   
25%       0.000000    35.000000   19287.420000            8.025412   
50%       1.000000    52.000000   33377.200000           10.016880   
75%       3.000000    71.000000   53517.120000           11.976770   
max      10.000000    89.000000  258900.700000           21.207230   

             Email     Contacts  Yearly_equip_failure       Tenure  \
count  5001.000000  5001.000000           5001.000000  5001.000000   
mean     12.051590     0.990202              0.392322     9.134829   
std       2.988467     0.983109              0.628373     6.041764   
min       1.000000     0.000000              0.000000     1.000259   
25%      10.000000     0.000000              0.000000     4.332135   
50%      12.000000     1.000000              0.000000     7.918063   
75%      14.000000     2.000000              1.000000    12.573060   
max      23.000000     7.000000              4.000000    37.119120   

       MonthlyCharge  Bandwidth_GB_Year        Item1        Item2  \
count    5001.000000        5001.000000  5001.000000  5001.000000   
mean      172.701463        1312.214450     3.499500     3.503099   
std        42.867453         572.374737     1.033465     1.025495   
min        79.978860         155.506715     1.000000     1.000000   
25%       139.981600         886.340074     3.000000     3.000000   
50%       169.937800        1236.530575     3.000000     4.000000   
75%       200.146524        1671.330908     4.000000     4.000000   
max       290.160419        3452.422228     7.000000     7.000000   

             Item3        Item4        Item5        Item6        Item7  \
count  5001.000000  5001.000000  5001.000000  5001.000000  5001.000000   
mean      3.485103     3.503299     3.470706     3.511098     3.511498   
std       1.026561     1.031717     1.018819     1.039195     1.021527   
min       1.000000     1.000000     1.000000     1.000000     1.000000   
25%       3.000000     3.000000     3.000000     3.000000     3.000000   
50%       3.000000     4.000000     3.000000     4.000000     4.000000   
```

75%	4.000000	4.000000	4.000000	4.000000	4.000000
max	7.000000	7.000000	7.000000	8.000000	7.000000

	Item8	Cluster
count	5001.000000	5001.0
mean	3.510898	1.0
std	1.033988	0.0
min	1.000000	1.0
25%	3.000000	1.0
50%	4.000000	1.0
75%	4.000000	1.0
max	8.000000	1.0

Cluster 0 Statistics:

	CaseOrder	Zip	Lat	Lng	Population \
count	4999.000000	4999.000000	4999.000000	4999.000000	4999.000000
mean	7499.045809	49103.012202	38.681024	-90.735796	9757.109422
std	1446.149085	27370.827026	5.396952	15.049419	14602.198349
min	1899.000000	683.000000	18.005430	-170.485200	0.000000
25%	6249.500000	26358.500000	35.190340	-97.001905	761.000000
50%	7500.000000	49037.000000	39.292620	-87.908510	2917.000000
75%	8750.500000	71824.500000	42.077680	-80.194705	12924.500000
max	10000.000000	99929.000000	70.368530	-65.667850	111850.000000

	Children	Age	Income	Outage_sec_perweek \
count	4999.000000	4999.000000	4999.000000	4999.000000
mean	2.080016	53.479496	39876.874795	10.011085
std	2.140054	20.694009	28371.726422	2.974425
min	0.000000	18.000000	643.200000	0.099747
25%	0.000000	35.000000	19167.905000	8.015255
50%	1.000000	53.000000	33016.710000	10.020680
75%	3.000000	71.000000	53007.500000	11.953570
max	10.000000	89.000000	256998.400000	20.625040

	Email	Contacts	Yearly_equip_failure	Tenure \
count	4999.000000	4999.000000	4999.000000	4999.000000
mean	11.980396	0.998200	0.403681	59.927706
std	3.062771	0.993877	0.643459	8.479329
min	1.000000	0.000000	0.000000	31.790270
25%	10.000000	0.000000	0.000000	54.379310
50%	12.000000	1.000000	0.000000	61.479870
75%	14.000000	2.000000	1.000000	66.857995
max	22.000000	7.000000	6.000000	71.999280

	MonthlyCharge	Bandwidth_GB_Year	Item1	Item2 \
count	4999.000000	4999.000000	4999.000000	4999.000000
mean	172.548138	5473.300867	3.482096	3.507101
std	43.022784	751.908006	1.042144	1.043808
min	79.978860	3170.023123	1.000000	1.000000

```
25%      139.967800      4967.359137    3.000000    3.000000
50%      167.456400      5586.428510    3.000000    4.000000
75%      202.443300      6036.215032    4.000000    4.000000
max      290.160400      7158.981530    7.000000    7.000000

             Item3        Item4        Item5        Item6        Item7  \
count  4999.000000  4999.000000  4999.000000  4999.000000  4999.000000
mean      3.488898     3.491698     3.515103     3.483497     3.507502
std       1.029491     1.019950     1.030411     1.027862     1.035530
min       1.000000     1.000000     1.000000     1.000000     1.000000
25%       3.000000     3.000000     3.000000     3.000000     3.000000
50%       3.000000     3.000000     4.000000     3.000000     4.000000
75%       4.000000     4.000000     4.000000     4.000000     4.000000
max       8.000000     7.000000     7.000000     7.000000     7.000000

             Item8  Cluster
count  4999.000000   4999.0
mean      3.480296      0.0
std       1.023123      0.0
min       1.000000      0.0
25%       3.000000      0.0
50%       3.000000      0.0
75%       4.000000      0.0
max       7.000000      0.0
```

In [ ]:

E

1)

The quality of the clusters that were created are being evaluated by several different metrics. The clusters have an inertia score of 61704.92517105245. A lower inertia indicates a tighter cluster. The clusters have a silhouette score of 0.22818537888812102. Positive silhouette scores indicate that the data points are closer to their own cluster center than to other cluster centers. A higher silhouette score indicates better clustering. The clusters have a davies-bouldin score of 1.7580509340625434. A lower davies-bouldin indicates well separated and compact clusters. The clusters have a Calinski-Harabasz Index score of 2964.336438829804. A higher Calinski-Harabasz score indicates better defined clusters. Based on the values of the metrics stated above I think the quality of the cluster analysis is very good. There is nothing indicating poor clustering such as a negative silhouette score.

2)

The results of the cluster analysis show that the customers in the churn data set can be grouped into 2 clusters. The number of clusters was dictated by using the elbow method against inertia values of different numbers of clusters. The analysis shows that the quality of the clusters is good so there are statistically significant differences in the characteristics of the customers assigned to each cluster. The cluster sizes are almost equal.

Some observations from each cluster are that the mean population of cluster 1 is slightly greater than cluster 0. The mean age of customers in cluster 1 is slightly greater than cluster 0. The mean income of customers in cluster 1 is slightly greater than that of customers in cluster 0. The mean outage_sec_perweek is slightly greater in cluster 1 compared to cluster 0. The mean monthly charge is slightly greater in cluster 0 compared to cluster 1.

Some implications of this cluster analysis are that the company can use this information to better serve customers in each cluster since the customers assigned to each cluster have different characteristics. For instance the mean outage_sec_perweek is greater in cluster 1. This implies that customers in that cluster should be targeted for services like online backup and should also be targeted for better service reliability to reduce customer churn. Another implication is that customers in cluster 0 have a higher monthly charge and therefore are good candidates for sales promotions that are designed by the marketing team to reduce customer churn.

3)

One limitation of my k-means cluster analysis is that the analysis only uses numeric continuous variables. I think that it would more informative if the clustering could use categorical variables such as 'Churn'
and 'Streaming_TV'. Because of this there is a lot of customer data not being used to form the clusters.

4)

One course of action based on the results of this analysis would be that we should spend more time and resources increasing reliability of services for customers assigned to cluster 1 because of the higher mean outage_sec_perweek. Sales and promotional campaigns can be tailored to reduce the monthly charge for customers assigned to cluster 0 because of the higher mean monthly charge. Advertisements can be created for a slightly older demographic and presented to the customers assigned to cluster 1 because of the higher mean age in that cluster.

```
In [ ]:
```