Import Required Libraries

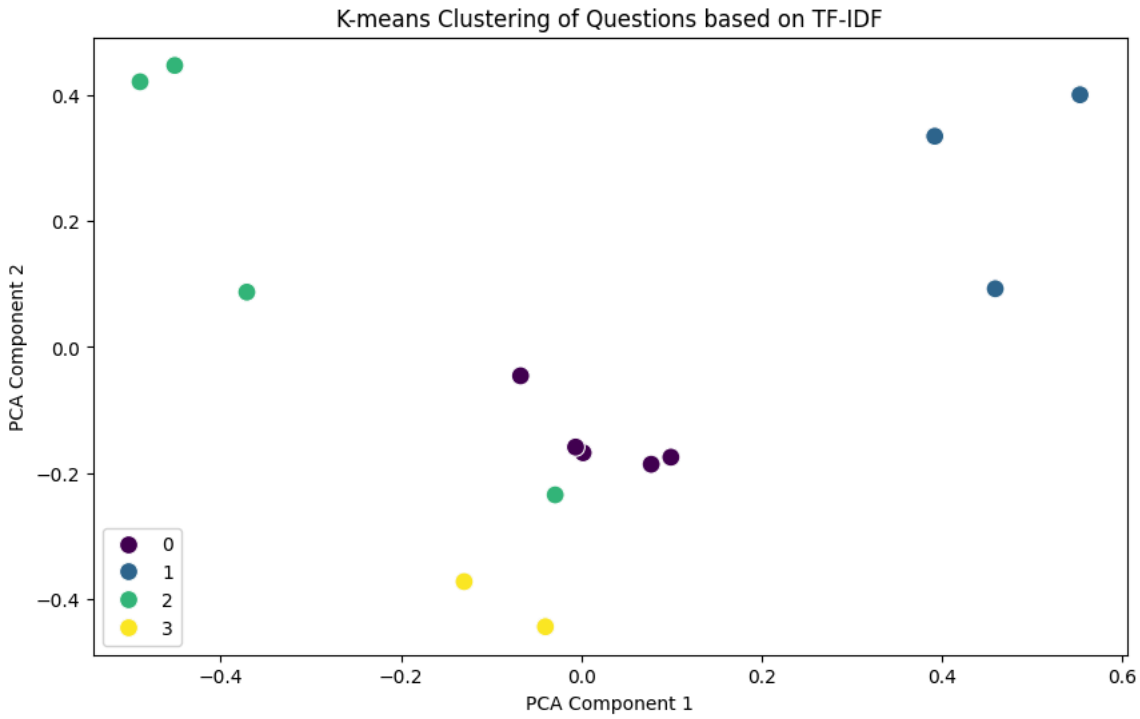
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import chi2_contingency
import seaborn as sns
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
```

Retrieve Datasets as Dataframes

```
df_2024 = pd.read_csv('Dataset Generation (2024) (Responses) - Form Responses 1.csv')
df_fardina = pd.read_csv('Dataset Generation (Fardina) (Responses) - Form Responses 1.csv')
df_max = pd.read_csv('Dataset Generation (Max) (Responses) - Form Responses 1.csv')


df_2024.drop_duplicates(inplace=True)
df_fardina.drop_duplicates(inplace=True)
df_max.drop_duplicates(inplace=True)

df_2024.dropna(inplace=True)
df_fardina.dropna(inplace=True)
df_max.dropna(inplace=True)

short_cols = ['Time', 'Year', 'Age', 'Politics of Parents', 'Politics of Self', 'Spirituality', 'Gender', 'Q1: Ignored Knee Pain', 'Q2: Reje

df_fardina_non_priming = pd.DataFrame()
df_fardina_non_priming[df_fardina.columns.to_list()[0:7] + df_fardina.columns.to_list()[8:]] = df_fardina[df_fardina.columns.to_list()[0:7]

df_fardina['How old are you?'] = df_fardina['How old are you?'].apply(func=lambda x: int(x) if x != '50+' else x)
df_max['How old are you?'] = df_max['How old are you?'].apply(func=lambda x: int(x) if x != '50+' else x)

fardina_cols = df_fardina_non_priming.columns.to_list()
max_cols = df_max.columns.to_list()
fardina_col_mapping = dict(zip(fardina_cols, short_cols))
max_col_mapping = dict(zip(max_cols, short_cols))

df_fardina_final = df_fardina_non_priming.rename(columns=fardina_col_mapping)
df_max_final = df_max.rename(columns=max_col_mapping)

df_2023_final = pd.concat([df_fardina_final, df_max_final])
df_2023_final['Gender'] = df_2023_final['Gender'].apply(func=lambda s: s.replace('Famale', 'Female') if isinstance(s, str) else s)

str_replace = lambda s: s.replace('religious', 'spiritual') if isinstance(s, str) else s
df_2024['How would you rate your religiousness?'] = df_2024['How would you rate your religiousness?'].apply(func=str_replace)
df_2024['How old are you?'] = df_2024['How old are you?'].apply(func=lambda x: int(x) if x != '50+' else x)

d2024_cols = df_2024.columns.to_list()
d2024_col_mapping = dict(zip(d2024_cols, short_cols))
df_2024_final = df_2024.rename(columns=d2024_col_mapping)

df_all = pd.concat([df_2023_final, df_2024_final])
df_all['Year'] = df_all['Year'].apply(func=lambda s: s.replace('Other', 'Graduate Student'))

df_fardina_priming_final = df_fardina_final.copy()
df_fardina_priming_final['Compassionate'] = df_fardina['Would you describe yourself as compassionate?']


# Sample list of questions as they appear in your data (e.g., Q1, Q2, ..., Q14)
questions = max_cols[7:]

# Step 1: Vectorize the text using TF-IDF
vectorizer = TfidfVectorizer(stop_words='english')
tfidf_matrix = vectorizer.fit_transform(questions)

# Step 2: Apply K-Means Clustering
num_clusters = 4  # Adjust the number of clusters based on your analysis
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
kmeans.fit(tfidf_matrix)

# Step 3: Analyze the clusters
```

```
clusters = kmeans.labels_

# Create a DataFrame with the questions and their assigned clusters
df_clusters = pd.DataFrame({'Question': questions, 'Cluster': clusters})

# Display the clustered questions
for cluster in range(num_clusters):
    print(f"\nCluster {cluster}:")
    print(df_clusters[df_clusters['Cluster'] == cluster]['Question'].tolist())


# Visualize the TF-IDF vectors in 2D using PCA for better understanding
pca = PCA(n_components=2)
reduced_tfidf = pca.fit_transform(tfidf_matrix.toarray())

plt.figure(figsize=(10, 6))
sns.scatterplot(x=reduced_tfidf[:, 0], y=reduced_tfidf[:, 1], hue=clusters, palette='viridis', s=100)
plt.title('K-means Clustering of Questions based on TF-IDF')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.show()
```

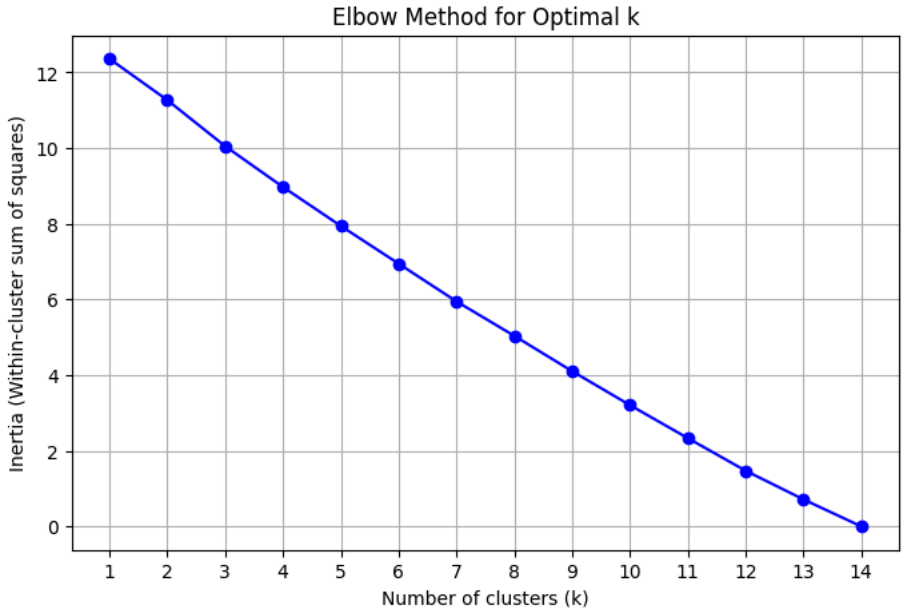
```
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default valu
  super()._check_params_vs_input(X, default_n_init=10)

Cluster 0:
["My girlfriend is a doctor. Lately she's been complaining about pain in her right knee and constantly taking TONS of ibuprofen to treat

Cluster 1:
["My wife and I have separate finances, but I pay for almost everything. My son starts school next year, and I'm planning on sending him

Cluster 2:
["I saw a poster for a lost cat advertising a 500 dollar reward. I saw the cat, tracked it down, and called the owner. When I met with t

Cluster 3:
["I'm a trust fund kid; I get a healthy 'allowance' from my parents, but I mostly sock it away since I don't really feel like I deserve
```




```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import numpy as np

# Assuming you have a TF-IDF matrix or any other feature matrix (e.g., question_vectors)
X = tfidf_matrix  # Replace this with your actual data matrix

# Step 1: Define range of k values to test
k_values = range(1, 15)  # Testing k from 1 to 10

# Step 2: Fit K-means models for each k and store the inertia (WCSS) values
```

```
inertia = []  # List to store the WCSS (inertia) for each k

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X)
    inertia.append(kmeans.inertia_)  # Append the inertia (within-cluster sum of squares)

# Step 3: Plot the elbow graph
plt.figure(figsize=(8, 5))
plt.plot(k_values, inertia, 'bo-', color='blue', marker='o')
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia (Within-cluster sum of squares)')
plt.xticks(k_values)
plt.grid(True)
plt.show()
```

```
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default valu
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\golde\AppData\Local\Temp\ipykernel_41288\2707131864.py:21: UserWarning: marker is redundantly defined by the 'marker' keyword a
  plt.plot(k_values, inertia, 'bo-', color='blue', marker='o')
C:\Users\golde\AppData\Local\Temp\ipykernel_41288\2707131864.py:21: UserWarning: color is redundantly defined by the 'color' keyword arg
  plt.plot(k_values, inertia, 'bo-', color='blue', marker='o')
```




```
df_clusters["Mapped Question"] = df_clusters["Question"].apply(func=lambda s: max_col_mapping[s])
```

```
df_clusters
```

	Question	Cluster	Mapped Question
0	My girlfriend is a doctor. Lately she's been c...	0	Q1: Ignored Knee Pain
1	My daughter is getting married soon. I only le...	0	Q2: Rejected Aisle Walk
2	I'm a trust fund kid; I get a healthy 'allowan...	3	Q3: Trust Fund Split 50/50
3	My wife and I have separate finances, but I pa...	1	Q4: Schooling Seperate Finances
4	I saw a poster for a lost cat advertising a 50...	2	Q5: Lost Cat, No Reward
5	My sister's nine year old daughter is poorly b...	0	Q6: Surprisig Child Drop-Off
6	My parents want us to come out for their anniv...	0	Q7: Business Class Over Kids
7	I'm a single mom with four kids, one of whom h...	2	Q8: Single Parent, Four Kids
8	I have a child with a mother who never wanted ...	2	Q9: Split Spouse Child Support
9	One of my children wants to go to an expensive...	1	Q10: Expensive School, Cheap Career
10	I was in a conflict with my mother-in-law's bo...	2	Q11: Ex Wedding Invitation Revoked
11	\nSome of my relatives refuse to come to my we...	3	Q12: Relatives Unapprove Wedding
12	My wife has decided that since she can't drink...	1	Q13: Pregnant Wife, No Drinks
13	My sister is going to be a bridesmaid at my we...	0	Q14: Rejected Dyed Hair

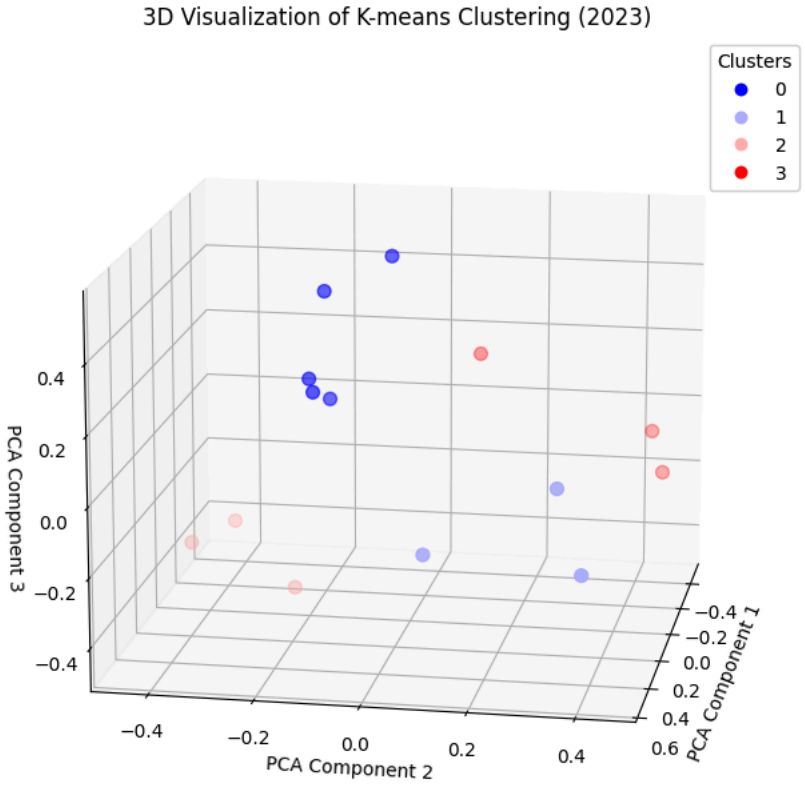
```
# Display the clustered mapped questions
for cluster in range(num_clusters):
    print(f"\nCluster {cluster}:")
    print(df_clusters[df_clusters['Cluster'] == cluster]['Mapped Question'].tolist())
```

```
Cluster 0:
['Q1: Ignored Knee Pain', 'Q2: Rejected Aisle Walk', 'Q6: Surprisig Child Drop-Off', 'Q7: Business Class Over Kids', 'Q14: Rejected Dyed

Cluster 1:
['Q4: Schooling Seperate Finances', 'Q10: Expensive School, Cheap Career', 'Q13: Pregnant Wife, No Drinks']

Cluster 2:
['Q5: Lost Cat, No Reward', 'Q8: Single Parent, Four Kids', 'Q9: Split Spouse Child Support', 'Q11: Ex Wedding Invitation Revoked']

Cluster 3:
['Q3: Trust Fund Split 50/50', 'Q12: Relatives Unapprove Wedding']
```

```
from mpl_toolkits.mplot3d import Axes3D

# Sample list of questions as they appear in your data (e.g., Q1, Q2, ..., Q14)
questions = max_cols[7:]

# Step 1: Vectorize the text using TF-IDF
vectorizer = TfidfVectorizer(stop_words='english')
tfidf_matrix = vectorizer.fit_transform(questions)

# Visualize the TF-IDF vectors in 2D using PCA for better understanding
pca = PCA(n_components=3)
reduced_tfidf = pca.fit_transform(tfidf_matrix.toarray())

# Step 2: Apply K-Means Clustering
num_clusters = 4  # Adjust the number of clusters based on your analysis
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
kmeans.fit(reduced_tfidf)

# Step 3: Analyze the clusters
clusters = kmeans.labels_

# Create a DataFrame with the questions and their assigned clusters
df_clusters2 = pd.DataFrame({'Question': questions,
                             'Cluster': clusters,
                             'PCA1': reduced_tfidf[:, 0],
                             'PCA2': reduced_tfidf[:, 1],
                             'PCA3': reduced_tfidf[:, 2]})

df_clusters2["Mapped Question"] = df_clusters2["Question"].apply(func=lambda s: max_col_mapping[s])

d2023_cluster_lst = []
# Display the clustered questions
```

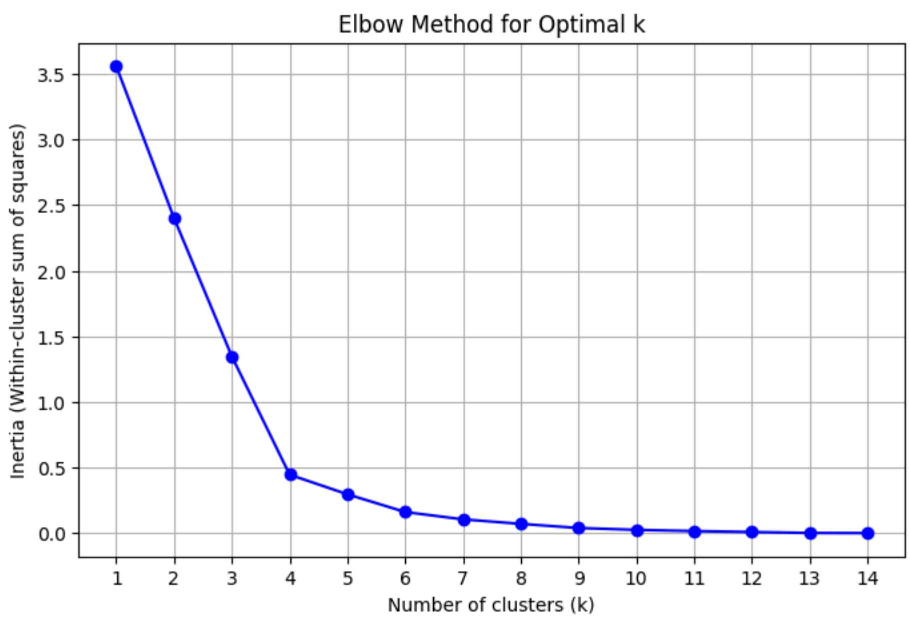
```
for cluster in range(num_clusters):
    print(f"\nCluster {cluster}:")
    print(df_clusters2[df_clusters2['Cluster'] == cluster]['Mapped Question'].tolist())
    d2023_cluster_lst.append(df_clusters2[df_clusters2['Cluster'] == cluster]['Mapped Question'].tolist())


# Step 4: Visualize the 3D K-means clustering
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')

# Create scatter plot
scatter = ax.scatter(df_clusters2['PCA1'], df_clusters2['PCA2'], df_clusters2['PCA3'],
                     c=df_clusters2['Cluster'], cmap='bwr', s=50)

# Set labels and title
ax.set_title('3D Visualization of K-means Clustering (2023)')
ax.set_xlabel('PCA Component 1')
ax.set_ylabel('PCA Component 2')
ax.set_zlabel('PCA Component 3')

# Create a legend with cluster labels
legend1 = ax.legend(*scatter.legend_elements(), title="Clusters")
ax.add_artist(legend1)

ax.view_init(elev = 15,azim=10)

plt.show()
```

```
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default valu
  super()._check_params_vs_input(X, default_n_init=10)

Cluster 0:
['Q1: Ignored Knee Pain', 'Q2: Rejected Aisle Walk', 'Q6: Surprisig Child Drop-Off', 'Q7: Business Class Over Kids', 'Q14: Rejected Dyed

Cluster 1:
['Q4: Schooling Seperate Finances', 'Q10: Expensive School, Cheap Career', 'Q13: Pregnant Wife, No Drinks']

Cluster 2:
['Q3: Trust Fund Split 50/50', 'Q5: Lost Cat, No Reward', 'Q12: Relatives Unapprove Wedding']

Cluster 3:
['Q8: Single Parent, Four Kids', 'Q9: Split Spouse Child Support', 'Q11: Ex Wedding Invitation Revoked']
```



```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import numpy as np

X = reduced_tfidf

k_values = range(1, 15)  # Testing k from 1 to 14

inertia = []

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X)
    inertia.append(kmeans.inertia_)

# Plot the elbow graph
plt.figure(figsize=(8, 5))
plt.plot(k_values, inertia, 'bo-', color='blue', marker='o')
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia (Within-cluster sum of squares)')
plt.xticks(k_values)
plt.grid(True)
plt.show()
```

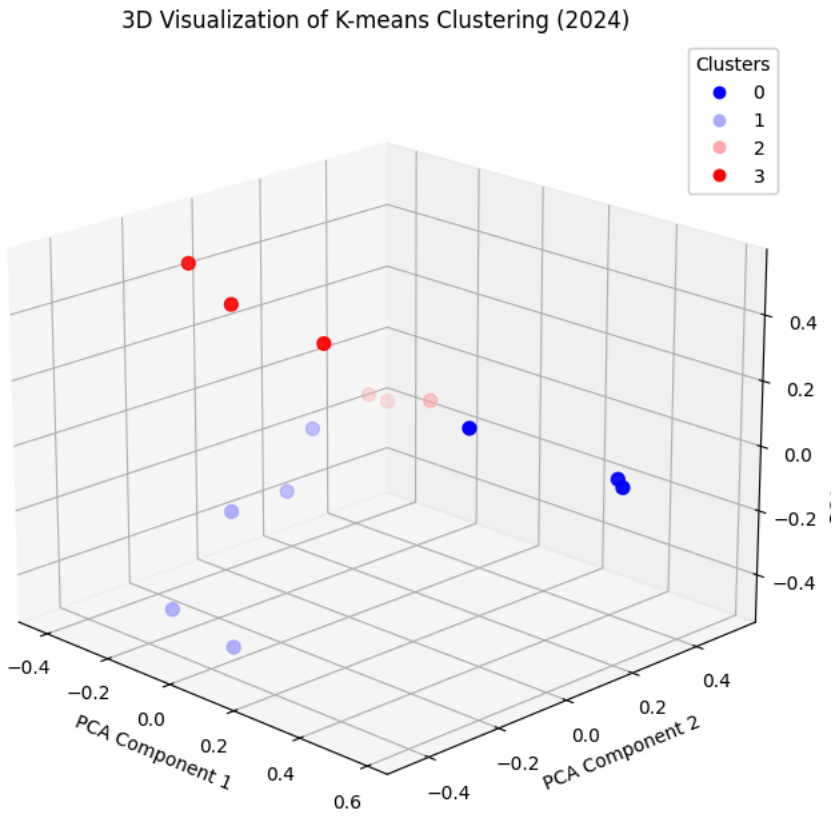
```
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\golde\AppData\Local\Temp\ipykernel_41288\3060935115.py:18: UserWarning: marker is redundantly defined by the 'marker' keyword
  plt.plot(k_values, inertia, 'bo-', color='blue', marker='o')
C:\Users\golde\AppData\Local\Temp\ipykernel_41288\3060935115.py:18: UserWarning: color is redundantly defined by the 'color' keyword a
  plt.plot(k_values, inertia, 'bo-', color='blue', marker='o')
```




```
from mpl_toolkits.mplot3d import Axes3D

# Sample list of questions as they appear in your data (e.g., Q1, Q2, ..., Q14)
questions = d2024_cols[7:]

# Step 1: Vectorize the text using TF-IDF
vectorizer = TfidfVectorizer(stop_words='english')
tfidf_matrix = vectorizer.fit_transform(questions)

# Visualize the TF-IDF vectors in 2D using PCA for better understanding
pca = PCA(n_components=3)
reduced_tfidf = pca.fit_transform(tfidf_matrix.toarray())

# Step 2: Apply K-Means Clustering
num_clusters = 4  # Adjust the number of clusters based on your analysis
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
kmeans.fit(reduced_tfidf)

# Step 3: Analyze the clusters
clusters = kmeans.labels_
```

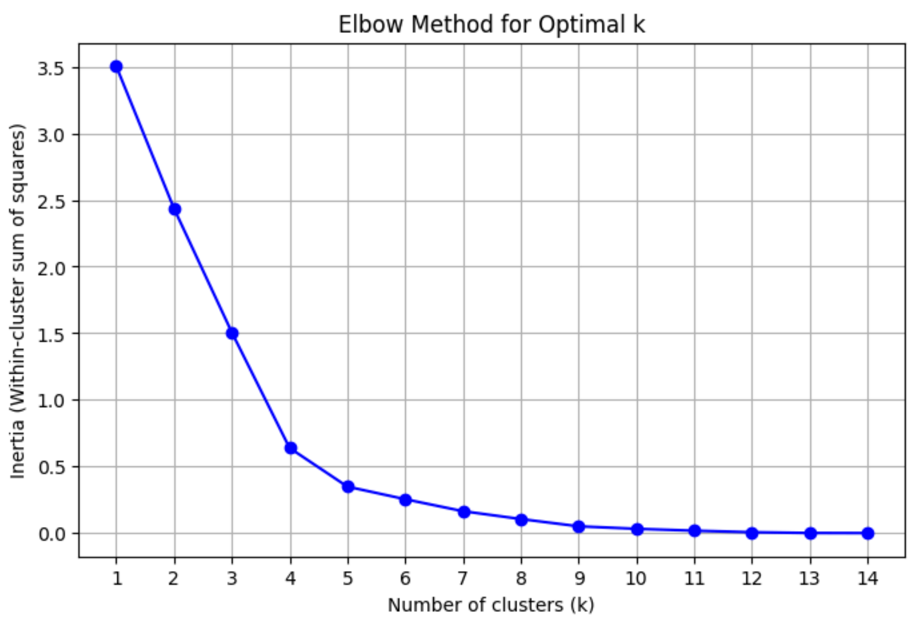
```
# Create a DataFrame with the questions and their assigned clusters
df_clusters2 = pd.DataFrame({'Question': questions,
                             'Cluster': clusters,
                             'PCA1': reduced_tfidf[:, 0],
                             'PCA2': reduced_tfidf[:, 1],
                             'PCA3': reduced_tfidf[:, 2]})

df_clusters2["Mapped Question"] = df_clusters2["Question"].apply(func=lambda s: d2024_col_mapping[s])

d2024_cluster_lst = []
# Display the clustered questions
for cluster in range(num_clusters):
    print(f"\nCluster {cluster}:")
    print(df_clusters2[df_clusters2['Cluster'] == cluster]['Mapped Question'].tolist())
    d2024_cluster_lst.append(df_clusters2[df_clusters2['Cluster'] == cluster]['Mapped Question'].tolist())


# Step 4: Visualize the 3D K-means clustering
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')

# Create scatter plot
scatter = ax.scatter(df_clusters2['PCA1'], df_clusters2['PCA2'], df_clusters2['PCA3'],
                     c=df_clusters2['Cluster'], cmap='bwr', s=50)

# Set labels and title
ax.set_title('3D Visualization of K-means Clustering (2024)')
ax.set_xlabel('PCA Component 1')
ax.set_ylabel('PCA Component 2')
ax.set_zlabel('PCA Component 3')

# Create a legend with cluster labels
legend1 = ax.legend(*scatter.legend_elements(), title="Clusters")
ax.add_artist(legend1)

ax.view_init(elev=20
             , azim=-45)

plt.show()
```

```
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default valu
  super()._check_params_vs_input(X, default_n_init=10)

Cluster 0:
['Q8: Single Parent, Four Kids', 'Q9: Split Spouse Child Support', 'Q11: Ex Wedding Invitation Revoked']

Cluster 1:
['Q1: Ignored Knee Pain', 'Q3: Trust Fund Split 50/50', 'Q5: Lost Cat, No Reward', 'Q7: Business Class Over Kids', 'Q12: Relatives Unapp

Cluster 2:
['Q4: Schooling Seperate Finances', 'Q6: Surprisig Child Drop-Off', 'Q10: Expensive School, Cheap Career']

Cluster 3:
['Q2: Rejected Aisle Walk', 'Q13: Pregnant Wife, No Drinks', 'Q14: Rejected Dyed Hair']
```




```
X = reduced_tfidf

k_values = range(1, 15)  # Testing k from 1 to 14

inertia = []

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X)
    inertia.append(kmeans.inertia_)

# Plot the elbow graph
plt.figure(figsize=(8, 5))
plt.plot(k_values, inertia, 'bo-', color='blue', marker='o')
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia (Within-cluster sum of squares)')
plt.xticks(k_values)
plt.grid(True)
plt.show()
```

```
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\golde\AppData\Local\Programs\Python\Python38\lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default va
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\golde\AppData\Local\Temp\ipykernel_41288\2351289573.py:14: UserWarning: marker is redundantly defined by the 'marker' keyword
  plt.plot(k_values, inertia, 'bo-', color='blue', marker='o')
C:\Users\golde\AppData\Local\Temp\ipykernel_41288\2351289573.py:14: UserWarning: color is redundantly defined by the 'color' keyword a
  plt.plot(k_values, inertia, 'bo-', color='blue', marker='o')
```




```
d2024_cluster_lst
```

```
[['Q8: Single Parent, Four Kids',
  'Q9: Split Spouse Child Support',
  'Q11: Ex Wedding Invitation Revoked'],
 ['Q1: Ignored Knee Pain',
  'Q3: Trust Fund Split 50/50',
  'Q5: Lost Cat, No Reward',
  'Q7: Business Class Over Kids',
  'Q12: Relatives Unapprove Wedding'],
 ['Q4: Schooling Seperate Finances',
  'Q6: Surprisig Child Drop-Off',
  'Q10: Expensive School, Cheap Career'],
 ['Q2: Rejected Aisle Walk',
  'Q13: Pregnant Wife, No Drinks',
  'Q14: Rejected Dyed Hair']]
```

```
d2023_cluster_lst.reverse()
```

```
d2023_cluster_lst
```

```
[['Q8: Single Parent, Four Kids',
  'Q9: Split Spouse Child Support',
  'Q11: Ex Wedding Invitation Revoked'],
 ['Q3: Trust Fund Split 50/50',
  'Q5: Lost Cat, No Reward',
  'Q12: Relatives Unapprove Wedding'],
 ['Q4: Schooling Seperate Finances',
  'Q10: Expensive School, Cheap Career',
  'Q13: Pregnant Wife, No Drinks'],
 ['Q1: Ignored Knee Pain',
  'Q2: Rejected Aisle Walk',
  'Q6: Surprisig Child Drop-Off',
  'Q7: Business Class Over Kids',
  'Q14: Rejected Dyed Hair']]
```

```
res = []
for i in range(0,4):
    set1 = set(d2024_cluster_lst[i])
    set2 = set(d2023_cluster_lst[i])
    res.append(set1.intersection(set2))

res
```

```
[{'Q11: Ex Wedding Invitation Revoked',
  'Q8: Single Parent, Four Kids',
  'Q9: Split Spouse Child Support'},
 {'Q12: Relatives Unapprove Wedding',
  'Q3: Trust Fund Split 50/50',
  'Q5: Lost Cat, No Reward'},
 {'Q10: Expensive School, Cheap Career', 'Q4: Schooling Seperate Finances'},
 {'Q14: Rejected Dyed Hair', 'Q2: Rejected Aisle Walk'}]
```

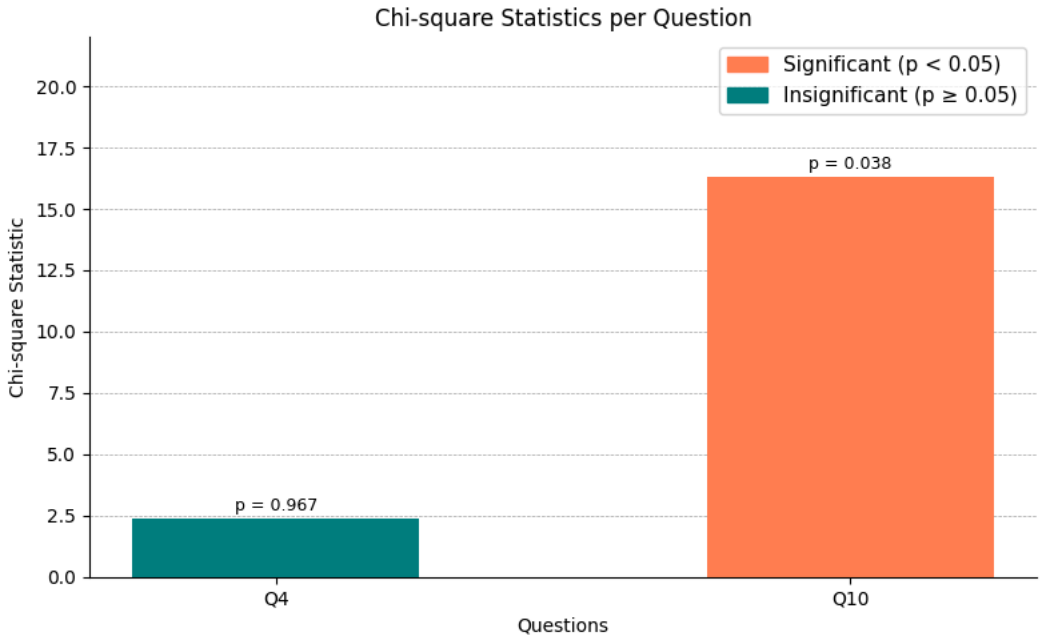
```
max_cols[7+11:8+11]
```

```
["\nSome of my relatives refuse to come to my wedding, since they don't approve of our 'lifestyle'. I would like to donate the money I will save to an LBGTQ organization in their name. I’m hoping that I or the organization will be able to send them a receipt/thank you for the donation. Would I be a jerk?"]
```

```
import matplotlib.patches as mpatches

ct = []

ct.append(pd.crosstab(df_all['Year'], df_all[short_cols[7+3]]))
ct.append(pd.crosstab(df_all['Year'], df_all[short_cols[7+9]]))


chi_square_stats_priming = []
p_values_priming = []
questions_priming = []

i = 4
for ct in ct:
    chi2, p, dof, expected = chi2_contingency(ct)
    chi_square_stats_priming.append(chi2)
    p_values_priming.append(p)
    questions_priming.append(f"Q{i}")
    print(f"Q{i}\t|\t{'Significant' if p < float(0.05) else 'Insignificant'}\t|\tChi-square statistic: {chi2}, p-value: {p}, Degrees of fre
    i += 6

bar_width = 0.5
positions = np.arange(len(questions_priming))

fig, ax = plt.subplots(figsize=(8, 5))

colors = ['coral' if p < 0.05 else 'teal' for p in p_values_priming]
bars = ax.bar(positions, chi_square_stats_priming, color=colors, width=bar_width)

ax.set_xticks(positions)
ax.set_xticklabels(questions_priming)

ax.set_xlabel('Questions')
ax.set_ylabel('Chi-square Statistic')
ax.set_title('Chi-square Statistics per Question')

red_patch = mpatches.Patch(color='coral', label='Significant (p < 0.05)')
blue_patch = mpatches.Patch(color='teal', label='Insignificant (p ≥ 0.05)')
```

```
ax.legend(handles=[red_patch, blue_patch], fontsize=11)  # Removed frameon=False
ax.set_ybound(upper=22)

for bar, p_value in zip(bars, p_values_priming):
    y = bar.get_height()
    x = bar.get_x() + bar.get_width() / 2
    ax.text(x, y + max(chi_square_stats_priming)*0.01, f'p = {p_value:.3f}', ha='center', va='bottom', fontsize=9)

ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

ax.yaxis.grid(True, which='major', linestyle='--', linewidth=0.5, color='gray', alpha=0.7)
ax.set_axisbelow(True)

plt.tight_layout()
plt.show()

significant_questions = [short_cols[7+i] for i, p in enumerate(p_values_priming) if p < 0.05]
print("Significant Questions (p < 0.05)")
print("-"*32)
for i in range(len(significant_questions)):
    print(f"{significant_questions[i]}")
```

```
Q4      |       Insignificant   |       Chi-square statistic: 2.3743770351362605, p-value: 0.967331677275035, Degrees of freedom: 8
Q10     |       Significant     |       Chi-square statistic: 16.29803120339662, p-value: 0.03830767213280616, Degrees of freedom: 8
```




```
Significant Questions (p < 0.05)
--------------------------------
Q2: Rejected Aisle Walk
```

```
import matplotlib.patches as mpatches

ct = []
```