[39]:
```
from sklearn.preprocessing import StandardScaler
import pandas as pd

# Load your data
data = pd.read_csv("C:/Users/MaRin/Desktop/nls_emp_survey.csv")
data.head()
```

[39]:

	Response	Employee_ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
0	1	810081	6	3	5	4	3	6	5	3	5	4	6	4
1	2	512221	2	5	3	3	6	2	3	5	3	2	2	3
2	3	177541	3	2	3	5	2	3	3	2	3	6	3	6
3	4	679938	3	5	3	3	6	2	2	5	2	2	3	2
4	5	777934	3	5	3	5	6	4	4	5	4	6	3	5

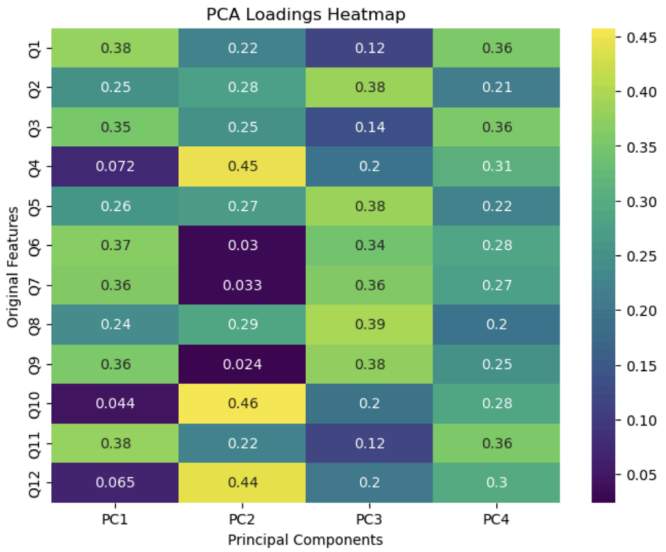
[69]:
```
loadings = pd.DataFrame(pca.components_.T, columns=[f'PC{i+1}' for i in range(pca.n_components_)], index=features.columns)
print(loadings)
```

```
         PC1       PC2       PC3       PC4       PC5       PC6       PC7  \
Q1   0.377841  0.224249 -0.115196 -0.357202 -0.027824 -0.377597  0.147696
Q2  -0.253190  0.276803  0.383411 -0.213789 -0.013124  0.003765  0.045597
Q3   0.351393  0.251104 -0.138598 -0.356234  0.049858  0.751025 -0.304280
Q4   0.072333 -0.445143  0.198855 -0.311536 -0.486822  0.089384 -0.006558
Q5  -0.260122  0.267763  0.382000 -0.221837 -0.002444 -0.076255 -0.127828
Q6   0.366604  0.029642  0.343320  0.282507  0.018812 -0.227827 -0.682999
Q7   0.361852  0.033165  0.360231  0.273266  0.029664 -0.019944  0.061169
Q8  -0.237301  0.288001  0.393521 -0.195376  0.011486  0.045639  0.046428
Q9   0.355177  0.024317  0.375288  0.250666 -0.055231  0.283121  0.602802
Q10  0.043768 -0.457583  0.195117 -0.284303 -0.310746 -0.047849 -0.086211
Q11  0.377841  0.224249 -0.115196 -0.357202 -0.027824 -0.377597  0.147696
Q12  0.065438 -0.441593  0.200932 -0.299368  0.811057 -0.012278  0.046220

         PC8       PC9      PC10      PC11          PC12
Q1   0.009394  0.007303  0.014827  0.023578 -7.071068e-01
Q2   0.127882  0.459144 -0.659940  0.040145  5.551115e-17
Q3  -0.067082 -0.008995 -0.034620 -0.049681  8.326673e-17
Q4   0.631054 -0.116429  0.051592 -0.023319 -1.249001e-16
Q5  -0.140555 -0.784702 -0.073497  0.049283 -3.677614e-16
Q6   0.080038  0.112877  0.046582  0.356342 -1.804112e-16
```

[69]:
```
loadings = pd.DataFrame(pca.components_.T, columns=[f'PC{i+1}' for i in range(pca.n_components_)], index=features.columns)
print(loadings)
```

```
         PC1       PC2       PC3       PC4       PC5       PC6       PC7  \
Q1   0.377841  0.224249 -0.115196 -0.357202 -0.027824 -0.377597  0.147696
Q2  -0.253190  0.276803  0.383411 -0.213789 -0.013124  0.003765  0.045597
Q3   0.351393  0.251104 -0.138598 -0.356234  0.049858  0.751025 -0.304280
Q4   0.072333 -0.445143  0.198855 -0.311536 -0.486822  0.089384 -0.006558
Q5  -0.260122  0.267763  0.382000 -0.221837 -0.002444 -0.076255 -0.127828
Q6   0.366604  0.029642  0.343320  0.282507  0.018812 -0.227827 -0.682999
Q7   0.361852  0.033165  0.360231  0.273266  0.029664 -0.019944  0.061169
Q8  -0.237301  0.288001  0.393521 -0.195376  0.011486  0.045639  0.046428
Q9   0.355177  0.024317  0.375288  0.250666 -0.055231  0.283121  0.602802
Q10  0.043768 -0.457583  0.195117 -0.284303 -0.310746 -0.047849 -0.086211
Q11  0.377841  0.224249 -0.115196 -0.357202 -0.027824 -0.377597  0.147696
Q12  0.065438 -0.441593  0.200932 -0.299368  0.811057 -0.012278  0.046220

         PC8       PC9      PC10      PC11          PC12
Q1   0.009394  0.007303  0.014827  0.023578 -7.071068e-01
Q2   0.127882  0.459144 -0.659940  0.040145  5.551115e-17
Q3  -0.067082 -0.008995 -0.034620 -0.049681  8.326673e-17
Q4   0.631054 -0.116429  0.051592 -0.023319 -1.249001e-16
Q5  -0.140555 -0.784702 -0.073497  0.049283 -3.677614e-16
Q6   0.080038  0.112877  0.046582  0.356342 -1.804112e-16
Q7  -0.003594 -0.059887 -0.091904 -0.804026  1.110223e-16
Q8   0.010984  0.335004  0.736643 -0.091389 -4.163336e-17
Q9  -0.093015 -0.078781  0.010038  0.456122 -0.000000e+00
Q10 -0.732470  0.157249 -0.037628 -0.029358  6.938894e-18
Q11  0.009394  0.007303  0.014827  0.023578  7.071068e-01
Q12  0.096341 -0.014054  0.001328  0.033722 -4.857226e-17
```

[71]:
```
import numpy as np
loading_df = pd.DataFrame(np.abs(pca.components_.T[:, :4]),
                          columns=[f'PC{i+1}' for i in range(4)],
                          index=features.columns)
plt.figure(figsize=(8, 6))
sns.heatmap(loading_df, annot=True, cmap='viridis')
plt.title("PCA Loadings Heatmap")
plt.xlabel("Principal Components")
plt.ylabel("Original Features")
plt.show()
```



[73]:
```
loading_threshold = 0.4
significant_loadings_df = loadings.applymap(lambda x: x if abs(x) >= loading_threshold else 0)
significant_loadings_df
```

[73]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12
Q1	0	0.000000	0	0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-0.707107
Q2	0	0.000000	0	0	0.000000	0.000000	0.000000	0.000000	0.459144	-0.659940	0.000000	0.000000
Q3	0	0.000000	0	0	0.000000	0.751025	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Q4	0	-0.445143	0	0	-0.486822	0.000000	0.000000	0.631054	0.000000	0.000000	0.000000	0.000000
Q5	0	0.000000	0	0	0.000000	0.000000	0.000000	0.000000	-0.784702	0.000000	0.000000	0.000000
Q6	0	0.000000	0	0	0.000000	0.000000	-0.682999	0.000000	0.000000	0.000000	0.000000	0.000000
Q7	0	0.000000	0	0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-0.804026	0.000000
Q8	0	0.000000	0	0	0.000000	0.000000	0.000000	0.000000	0.000000	0.736643	0.000000	0.000000
Q9	0	0.000000	0	0	0.000000	0.000000	0.602802	0.000000	0.000000	0.000000	0.456122	0.000000
Q10	0	-0.457583	0	0	0.000000	0.000000	0.000000	-0.732470	0.000000	0.000000	0.000000	0.000000
Q11	0	0.000000	0	0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.707107
Q12	0	-0.441593	0	0	0.811057	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

[81]:
```
pca_summary_df = pd.DataFrame({
    'Principal Component': [f'PC{i+1}' for i in range(len(pca.explained_variance_))],
    'Explained Variance Ratio': pca.explained_variance_ratio_,
    'Cumulative Variance Ratio': pca.explained_variance_ratio_.cumsum(),
    'Eigenvalue': pca.explained_variance_
})
pca_summary_df
```

[63]:

	Principal Component	Explained Variance Ratio	Cumulative Variance Ratio	Eigenvalue
0	PC1	3.722901e-01	0.372290	4.474939e+00
1	PC2	3.132278e-01	0.685518	3.765009e+00
2	PC3	2.041820e-01	0.889700	2.454275e+00
3	PC4	8.657420e-02	0.976274	1.040625e+00
4	PC5	7.022877e-03	0.983297	8.441522e-02
5	PC6	4.191542e-03	0.987489	5.038248e-02
6	PC7	3.282081e-03	0.990771	3.945073e-02
7	PC8	2.774222e-03	0.993545	3.334624e-02
8	PC9	2.405066e-03	0.995950	2.890898e-02
9	PC10	2.140136e-03	0.998090	2.572451e-02
10	PC11	1.909967e-03	1.000000	2.295787e-02
11	PC12	1.614290e-33	1.000000	1.940382e-32

[75]:

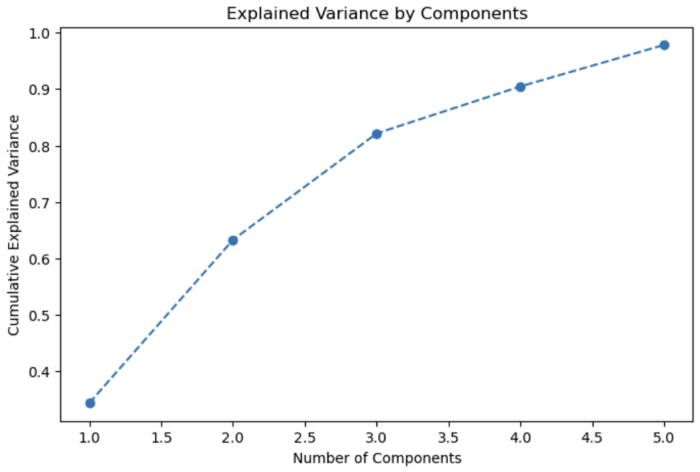
```
from sklearn.cluster import KMeans
n_clusters = 3
n_pcs = 3
kmeans = KMeans(n_clusters=n_clusters, max_iter=100, n_init=10)
kmeans.fit(pca_data[:, :n_pcs])
cluster_centers_pca = pd.DataFrame(kmeans.cluster_centers_, columns=[f'PC{i+1}' for i in range(n_pcs)])
cluster_sizes = pd.Series(kmeans.labels_).value_counts(normalize=True).sort_index()
cluster_centers_pca['size'] = cluster_sizes.values
cluster_centers_pca.rename(columns={'PC1': 'Dimension 1', 'PC2': 'Dimension 2', 'PC3': 'Dimension 3'}, inplace=True)
cluster_centers_pca
```

[75]:

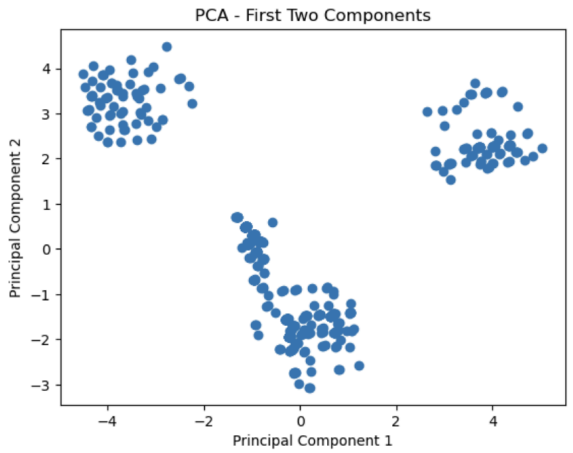
	Dimension 1	Dimension 2	Dimension 3	size
0	-3.699924	3.278670	-1.132007	0.116667
1	-0.289671	-1.083482	0.261092	0.716667
2	3.835531	2.363906	-0.330290	0.166667

[21]:

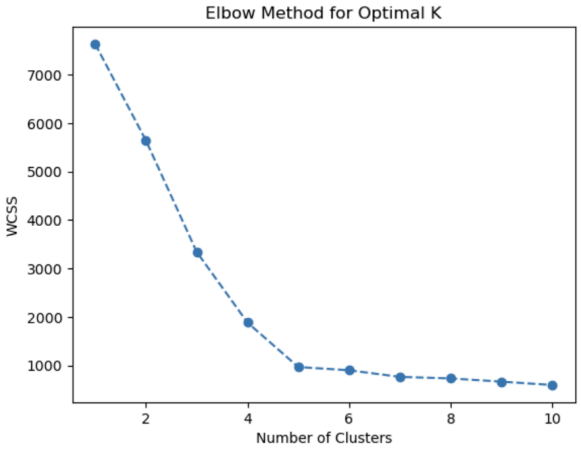
```
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5))
plt.plot(range(1, len(explained_variance) + 1), explained_variance.cumsum(), marker='o', linestyle='--')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('Explained Variance by Components')
plt.show()
```



[23]:
```
plt.scatter(pca_df['PC1'], pca_df['PC2'])
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA - First Two Components')
plt.show()
```



[25]:
```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=0)
    kmeans.fit(principal_components)
    wcss.append(kmeans.inertia_)

plt.plot(range(1, 11), wcss, marker='o', linestyle='--')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.title('Elbow Method for Optimal K')
plt.show()
```

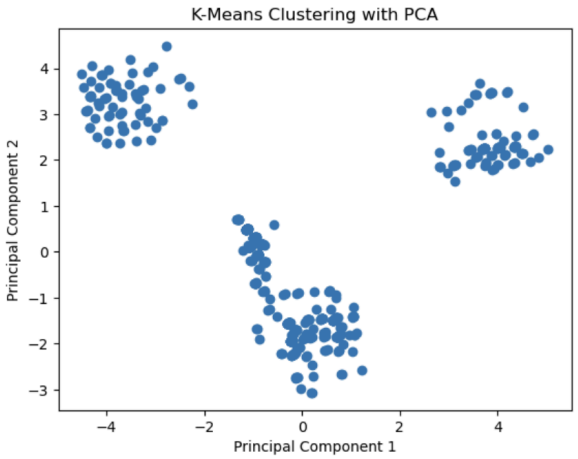


[53]:
```
optimal_clusters = 3
kmeans = KMeans(n_clusters=optimal_clusters, random_state=0)
clusters = kmeans.fit_predict(principal_components)
pca_df['Cluster'] = clusters
```

[29]:
```
cluster_centers = kmeans.cluster_centers_
print("Cluster Centers:\n", cluster_centers)
```

```
Cluster Centers:
 [[-1.84493146  1.13404917  0.88913879 -0.42447063 -0.26113065]
 [ 0.07918145 -1.73440767 -0.57999861  0.35899491  0.21237233]
 [ 3.83714116  2.3614333  -0.33210923 -0.07135037 -0.02015511]]
```

[51]:
```
plt.scatter(pca_df['PC1'], pca_df['PC2'])
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('K-Means Clustering with PCA')

plt.show()
```



•[85]:
```
from sklearn.cluster import KMeans
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import numpy as np
kmeans = KMeans(n_clusters=2, random_state=42)  # Adjust n_clusters as needed
pca_df['cluster'] = kmeans.fit_predict(pca_df[['PC1', 'PC2']])
y_pred = pca_df['cluster']
cf = confusion_matrix(y_true, y_pred)
ConfusionMatrixDisplay(cf).plot(cmap='Blues')
plt.xlabel("Predicted Cluster")
plt.ylabel("True Cluster")
plt.title("Confusion Matrix for K-Means Clustering")
plt.show()
```

```
C:\Users\MaRin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
```