Loading the dataset from UC Irvine Datasets

In [19]:
```
from sklearn.preprocessing import StandardScaler
import pandas as pd
from ucimlrepo import fetch_ucirepo
import matplotlib.pyplot as plt
```

In [20]:
```
travel_review_ratings = fetch_ucirepo(id=485)

X = travel_review_ratings.data.features
y = travel_review_ratings.data.targets

print(travel_review_ratings.metadata)
print(travel_review_ratings.variables)
```

```
{'uci_id': 485, 'name': 'Travel Review Ratings', 'repository_url': 'https://
archive.ics.uci.edu/dataset/485/tarvel+review+ratings', 'data_url': 'http
s://archive.ics.uci.edu/static/public/485/data.csv', 'abstract': 'Google rev
iews on attractions from 24 categories across Europe are considered. Google
user rating ranges from 1 to 5 and average user rating per category is calcu
lated.', 'area': 'Other', 'tasks': ['Classification', 'Clustering'], 'charac
teristics': ['Multivariate', 'Text'], 'num_instances': 5456, 'num_features':
24, 'feature_types': ['Real'], 'demographics': [], 'target_col': None, 'inde
x_col': ['userid'], 'has_missing_values': 'no', 'missing_values_symbol': Non
e, 'year_of_dataset_creation': 2018, 'last_updated': 'Tue Apr 09 2024', 'dat
aset_doi': '10.24432/C5C31Q', 'creators': ['Shini Renjith'], 'intro_paper':
{'ID': 466, 'type': 'NATIVE', 'title': 'Evaluation of Partitioning Clusterin
g Algorithms for Processing Social Media Data in Tourism Domain', 'authors':
'Dr. Shini Renjith, A. Sreekumar, M. Jathavedan', 'venue': 'IEEE Recent Adva
nces in Intelligent Computational Systems', 'year': 2018, 'journal': None,
'DOI': None, 'URL': 'https://www.semanticscholar.org/paper/Evaluation-of-Par
titioning-Clustering-Algorithms-in-Renjith-Sreekumar/0c667df7f0adb8b15ae1c39
e4f5cc2ebad0ce33f', 'sha': None, 'corpus': None, 'arxiv': None, 'mag': None,
'acl': None, 'pmid': None, 'pmcid': None}, 'additional_info': {'summary': 'T
his data set is populated by capturing user ratings from Google reviews. Rev
iews on attractions from 24 categories across Europe are considered. Google
user rating ranges from 1 to 5 and average user rating per category is calcu
lated. ', 'purpose': None, 'funded_by': None, 'instances_represent': None,
'recommended_data_splits': None, 'sensitive_data': None, 'preprocessing_desc
ription': None, 'variable_info': 'Attribute 1 : Unique user id\r\nAttribute
2 : Average ratings on churches\r\nAttribute 3 : Average ratings on resorts
\r\nAttribute 4 : Average ratings on beaches\r\nAttribute 5 : Average rating
s on parks\r\nAttribute 6 : Average ratings on theatres\r\nAttribute 7 : Ave
rage ratings on museums\r\nAttribute 8 : Average ratings on malls\r\nAttribu
te 9 : Average ratings on zoo\r\nAttribute 10 : Average ratings on restauran
ts\r\nAttribute 11 : Average ratings on pubs/bars\r\nAttribute 12 : Average
ratings on local services\r\nAttribute 13 : Average ratings on burger/pizza
shops\r\nAttribute 14 : Average ratings on hotels/other lodgings\r\nAttribut
e 15 : Average ratings on juice bars\r\nAttribute 16 : Average ratings on ar
t galleries\r\nAttribute 17 : Average ratings on dance clubs\r\nAttribute 18
: Average ratings on swimming pools\r\nAttribute 19 : Average ratings on gym
s\r\nAttribute 20 : Average ratings on bakeries\r\nAttribute 21 : Average ra
tings on beauty & spas\r\nAttribute 22 : Average ratings on cafes\r\nAttribu
te 23 : Average ratings on view points\r\nAttribute 24 : Average ratings on
monuments\r\nAttribute 25 : Average ratings on gardens', 'citation': None}}
```


```
                     name     role        type demographic description unit
s  \
0                  userid       ID  Categorical        None        None  Non
e
1                churches  Feature   Continuous        None        None  Non
e
2                 resorts  Feature   Continuous        None        None  Non
e
3                 beaches  Feature      Integer        None        None  Non
e
4                   parks  Feature   Continuous        None        None  Non
e
5                theatres  Feature   Continuous        None        None  Non
e
6                 museums  Feature   Continuous        None        None  Non
e
```

```
7                    malls  Feature   Continuous        None        None  Non
e
8                     zoos  Feature   Continuous        None        None  Non
e
9              restaurants  Feature      Integer        None        None  Non
e
10               pubs/bars  Feature   Continuous        None        None  Non
e
11          local services  Feature   Continuous        None        None  Non
e
12      burger/pizza shops  Feature   Continuous        None        None  Non
e
13  hotels/other lodgings  Feature   Continuous        None        None  Non
e
14              juice bars  Feature   Continuous        None        None  Non
e
15           art galleries  Feature      Integer        None        None  Non
e
16             dance clubs  Feature   Continuous        None        None  Non
e
17          swimming pools  Feature   Continuous        None        None  Non
e
18                    gyms  Feature   Continuous        None        None  Non
e
19               bakeries  Feature   Continuous        None        None  Non
e
20           beauty & spas  Feature   Continuous        None        None  Non
e
21                   cafes  Feature   Continuous        None        None  Non
e
22             view points  Feature   Continuous        None        None  Non
e
23               monuments  Feature   Continuous        None        None  Non
e
24                 gardens  Feature   Continuous        None        None  Non
e

   missing_values
0              no
1              no
2              no
3              no
4              no
5              no
6              no
7              no
8              no
9              no
10             no
11             no
12             no
13             no
14             no
15             no
16             no
17             no
```

```
18              no
19              no
20              no
21              no
22              no
23              no
24              no
```

Cleaning Data - Even though it has no missing values there are some features I need to eliminate

In [21]:
```python
if 'userid' in X.columns:
    X = X.drop(columns=['userid'])
print(X.head())
```

```
   churches  resorts  beaches  parks  theatres  museums  malls  zoos  \
0       0.0      0.0     3.63   3.65       5.0     2.92    5.0  2.35
1       0.0      0.0     3.63   3.65       5.0     2.92    5.0  2.64
2       0.0      0.0     3.63   3.63       5.0     2.92    5.0  2.64
3       0.0      0.5     3.63   3.63       5.0     2.92    5.0  2.35
4       0.0      0.0     3.63   3.63       5.0     2.92    5.0  2.64

   restaurants  pubs/bars  ... art galleries  dance clubs  swimming pools  \
0         2.33       2.64  ...          1.74         0.59             0.5
1         2.33       2.65  ...          1.74         0.59             0.5
2         2.33       2.64  ...          1.74         0.59             0.5
3         2.33       2.64  ...          1.74         0.59             0.5
4         2.33       2.64  ...          1.74         0.59             0.5

   gyms  bakeries  beauty & spas  cafes  view points  monuments  gardens
0   0.0       0.5            0.0    0.0          0.0        0.0      0.0
1   0.0       0.5            0.0    0.0          0.0        0.0      0.0
2   0.0       0.5            0.0    0.0          0.0        0.0      0.0
3   0.0       0.5            0.0    0.0          0.0        0.0      0.0
4   0.0       0.5            0.0    0.0          0.0        0.0      0.0

[5 rows x 24 columns]
```

Since these are all ratings they should all be numerical, if they aren't them something is wrong, so I forced non numerical inputs to na, and then we will count how many na are in the dataset now

In [22]:
```python
X = X.apply(pd.to_numeric, errors='coerce')
print(X.isna().sum())
```

```
churches                 0
resorts                  0
beaches                  0
parks                    0
theatres                 0
museums                  0
malls                    0
zoos                     0
restaurants              0
pubs/bars                0
local services           1
burger/pizza shops       1
hotels/other lodgings    0
juice bars               0
art galleries            0
dance clubs              0
swimming pools           0
gyms                     0
bakeries                 0
beauty & spas            0
cafes                    0
view points              0
monuments                0
gardens                  1
dtype: int64
```

Since there is very minimal, I chose to input the mean value of there columns to replace the na values, this will not chnage the dataset much

In [23]:

```
X = X.fillna(X.mean())
print(X.isna().sum())
```

```
churches                0
resorts                 0
beaches                 0
parks                   0
theatres                0
museums                 0
malls                   0
zoos                    0
restaurants             0
pubs/bars               0
local services          0
burger/pizza shops      0
hotels/other lodgings   0
juice bars              0
art galleries           0
dance clubs             0
swimming pools          0
gyms                    0
bakeries                0
beauty & spas           0
cafes                   0
view points             0
monuments               0
gardens                 0
dtype: int64
```

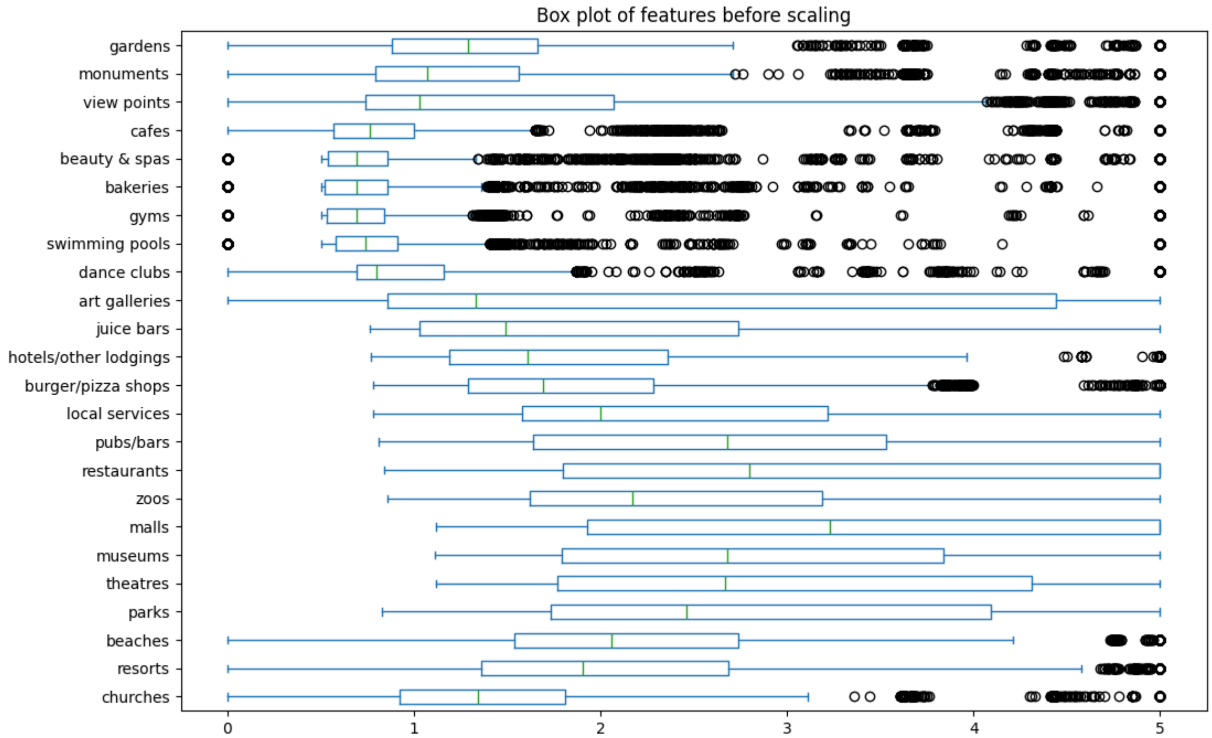
Scaling the features so they are all within the same range

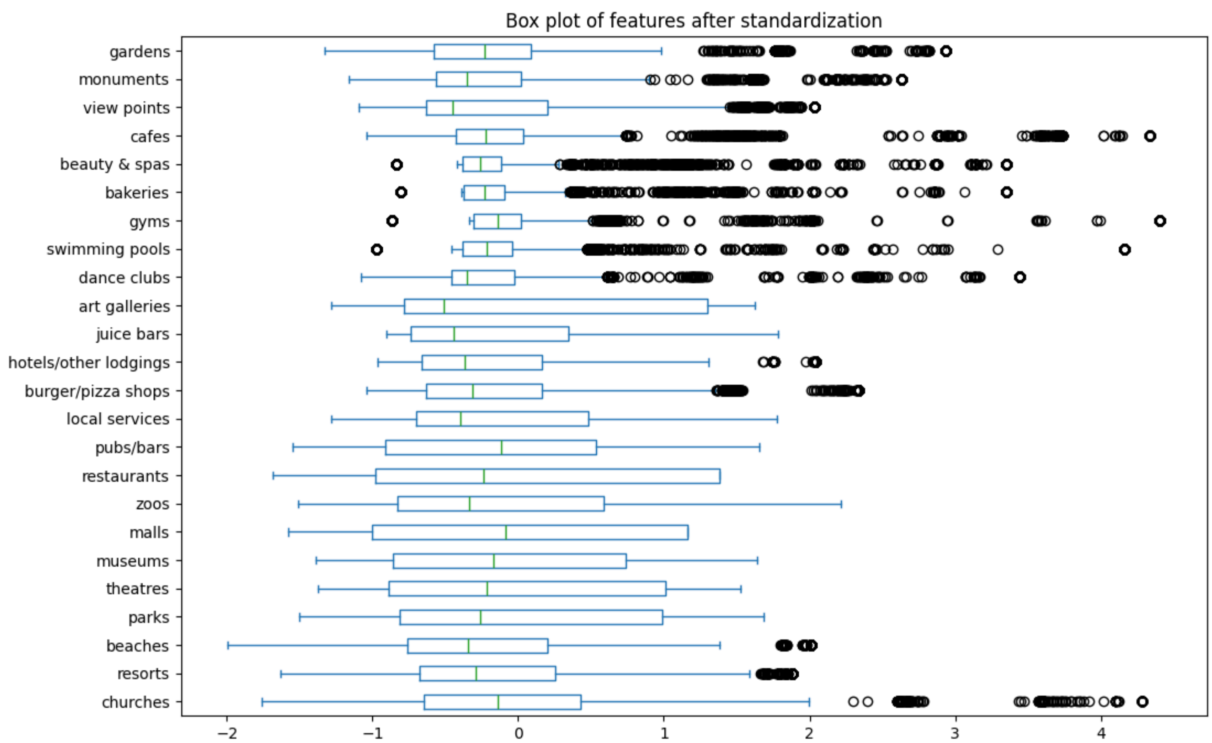
In [24]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_scaled_df = pd.DataFrame(X_scaled, columns=X.columns)

print(X_scaled_df.head())
```

```
   churches   resorts   beaches     parks  theatres   museums     malls  \
0 -1.759118 -1.632094  0.914217  0.651710  1.524392  0.020674  1.166442
1 -1.759118 -1.632094  0.914217  0.651710  1.524392  0.020674  1.166442
2 -1.759118 -1.632094  0.914217  0.636432  1.524392  0.020674  1.166442
3 -1.759118 -1.280305  0.914217  0.636432  1.524392  0.020674  1.166442
4 -1.759118 -1.632094  0.914217  0.636432  1.524392  0.020674  1.166442

       zoos  restaurants  pubs/bars  ...  art galleries  dance clubs  \
0 -0.171688    -0.586741  -0.147398  ...      -0.271927    -0.544583
1  0.089270    -0.586741  -0.139750  ...      -0.271927    -0.544583
2  0.089270    -0.586741  -0.147398  ...      -0.271927    -0.544583
3 -0.171688    -0.586741  -0.147398  ...      -0.271927    -0.544583
4  0.089270    -0.586741  -0.147398  ...      -0.271927    -0.544583

   swimming pools      gyms  bakeries  beauty & spas     cafes  view points
\
0       -0.461456 -0.867686 -0.390254      -0.837734 -1.038794    -1.095052
1       -0.461456 -0.867686 -0.390254      -0.837734 -1.038794    -1.095052
2       -0.461456 -0.867686 -0.390254      -0.837734 -1.038794    -1.095052
3       -0.461456 -0.867686 -0.390254      -0.837734 -1.038794    -1.095052
4       -0.461456 -0.867686 -0.390254      -0.837734 -1.038794    -1.095052

   monuments   gardens
0   -1.16304 -1.332223
1   -1.16304 -1.332223
2   -1.16304 -1.332223
3   -1.16304 -1.332223
4   -1.16304 -1.332223

[5 rows x 24 columns]
```

Let's take a look at the data and see what we've done so far and how the ratings look before and after scaling them

In [25]:
```
X.plot(kind='box', vert=False, figsize=(12, 8))
plt.title("Box plot of features before scaling")
plt.show()

X_scaled_df.plot(kind='box', vert=False, figsize=(12, 8))
plt.title("Box plot of features after standardization")
plt.show()
```





Below, we fit the data to K-Means clustering. The first step is to apply the K-Means algorithm to the dataset, where the goal is to assign each user to one of several clusters based on their ratings for different types of attractions (e.g., parks, beaches, museums, zoos, etc.).

Each user is assigned to a cluster based on their rating patterns. For example, users who consistently rate parks and beaches low (or similarly for other categories) will be grouped together in the same cluster. This means that users with similar preferences across the different attraction categories will be placed in the same cluster, reflecting shared tastes and behaviors.

The K-Means algorithm finds patterns in the data, and these patterns allow us to segment users into distinct groups, each with a "typical" user profile. These profiles are represented by the centroids of the clusters, which are the average ratings for each attraction category across all users in that cluster.

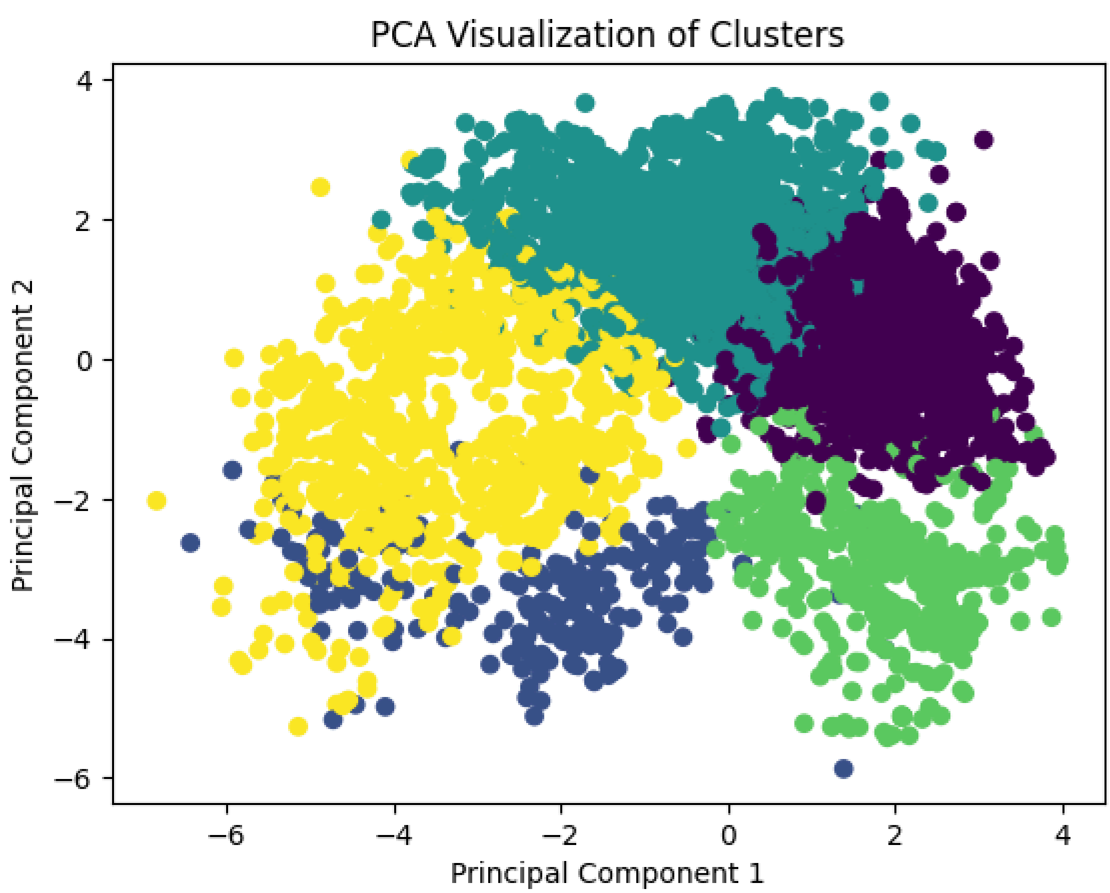
In [30]:

```
from sklearn.cluster import KMeans

# Apply K-Means clustering on the scaled data
kmeans = KMeans(n_clusters=5, random_state=42)
X_scaled_df['Cluster'] = kmeans.fit_predict(X_scaled_df)

# View the resulting clusters
print(X_scaled_df)
```

```
      churches   resorts   beaches     parks  theatres   museums     malls  \
0    -1.759118 -1.632094  0.914217  0.651710  1.524392  0.020674  1.166442
1    -1.759118 -1.632094  0.914217  0.651710  1.524392  0.020674  1.166442
2    -1.759118 -1.632094  0.914217  0.636432  1.524392  0.020674  1.166442
3    -1.759118 -1.280305  0.914217  0.636432  1.524392  0.020674  1.166442
4    -1.759118 -1.632094  0.914217  0.636432  1.524392  0.020674  1.166442
...        ...       ...       ...       ...       ...       ...       ...
5451 -0.659458  1.885794  1.210762 -0.005260 -0.141113 -0.252276 -0.651917
5452 -0.635290  1.885794  1.226791 -0.005260 -0.133644 -0.252276 -1.118888
5453 -0.623205  1.885794  1.234806  0.002379 -0.133644 -0.252276 -1.133039
5454 -0.611121  1.217396  1.250836  0.010018 -0.126175 -0.353658 -1.125963
5455 -0.611121  1.231467  2.012236  0.017657 -0.118707 -0.252276 -0.658992

          zoos  restaurants  pubs/bars  ...  dance clubs  swimming pools  \
0    -0.171688    -0.586741  -0.147398  ...    -0.544583       -0.461456
1     0.089270    -0.586741  -0.139750  ...    -0.544583       -0.461456
2     0.089270    -0.586741  -0.147398  ...    -0.544583       -0.461456
3    -0.171688    -0.586741  -0.147398  ...    -0.544583       -0.461456
4     0.089270    -0.586741  -0.147398  ...    -0.544583       -0.461456
...        ...          ...        ...  ...          ...             ...
5451 -1.305507    -0.999515  -1.371065  ...    -0.481343       -0.307364
5452 -1.323504    -1.006886  -1.386361  ...    -0.490378       -0.317637
5453 -1.341501    -1.014256  -1.401657  ...    -0.490378       -0.327910
5454 -1.359498    -1.021627  -1.416953  ...    -0.499412       -0.327910
5455 -1.368497    -1.021627  -1.432249  ...    -0.499412       -0.338183

          gyms  bakeries  beauty & spas     cafes  view points  monuments  \
0    -0.867686 -0.390254      -0.837734 -1.038794    -1.095052  -1.163040
1    -0.867686 -0.390254      -0.837734 -1.038794    -1.095052  -1.163040
2    -0.867686 -0.390254      -0.837734 -1.038794    -1.095052  -1.163040
3    -0.867686 -0.390254      -0.837734 -1.038794    -1.095052  -1.163040
4    -0.867686 -0.390254      -0.837734 -1.038794    -1.095052  -1.163040
...        ...       ...            ...       ...          ...        ...
5451 -0.171354 -0.232428       3.350637  0.090520     2.032709   2.634136
5452 -0.181905  0.515166       0.519298  0.101275     2.032709   2.634136
5453 -0.192455 -0.190895       3.350637  0.112031     2.032709   2.634136
5454 -0.192455 -0.182589       3.350637  0.122786     2.032709   2.634136
5455 -0.203006 -0.157669       3.350637  0.122786     2.032709   2.634136

       gardens  Cluster
0    -1.332223        1
1    -1.332223        1
2    -1.332223        1
3    -1.332223        1
4    -1.332223        1
...        ...      ...
5451 -0.000645        2
5452 -0.401825        2
5453 -0.384754        2
5454 -0.376218        2
5455 -0.333539        2

[5456 rows x 25 columns]
```

The K-Means clustering analysis groups users based on their rating patterns across 24 attraction categories. Each cluster represents a distinct user group with similar preferences—such as users who rate outdoor attractions like parks and beaches highly, or those who favor cultural experiences like museums and galleries. By visualizing the clusters, we can identify common trends in user behavior, such as whether they tend to rate attractions positively or negatively. These insights can be used for targeted recommendations and personalized strategies based on user preferences.

In [27]:
```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled_df.drop(columns=['Cluster']))

plt.scatter(X_pca[:, 0], X_pca[:, 1], c=X_scaled_df['Cluster'], cmap='viridi
plt.title("PCA Visualization of Clusters")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()
```



In [28]:
```
# Get the cluster centroids
centroids = kmeans.cluster_centers_

# Convert centroids into a DataFrame for easy interpretation
centroids_df = pd.DataFrame(centroids, columns=X.columns)
```

```python
# Display the centroids
print(centroids_df)
```

```
   churches   resorts   beaches     parks  theatres   museums     malls  \
0 -0.343016 -0.056747 -0.412691 -0.441040 -0.380687  0.125228  0.626680
1  0.375196 -0.251747 -0.360444 -0.587500 -0.708876 -0.810138 -0.500706
2  0.027303  0.188445  0.581238  0.893237  1.066304  0.633267  0.027782
3 -0.903898 -0.927080 -0.680689 -0.890922 -0.942703 -0.902997 -0.245893
4  1.074139  0.383181  0.183342 -0.181839 -0.574079 -0.708432 -0.988901

       zoos  restaurants  pubs/bars  ...  art galleries  dance clubs  \
0  0.865373     0.968179   0.745712  ...       0.141664    -0.210834
1 -0.527896    -0.634259  -0.561111  ...       0.603244     1.929578
2 -0.159559    -0.386447  -0.220776  ...      -0.487430    -0.100207
3 -0.497473    -0.207308   0.062835  ...       0.967215    -0.120362
4 -0.923618    -0.830304  -0.908581  ...      -0.081839     0.163168

   swimming pools      gyms  bakeries  beauty & spas     cafes  view points
\
0       -0.311912 -0.319090 -0.365085      -0.297690 -0.285168    -0.419118
1        3.539079  2.511428  0.947175       0.034341  0.529534     0.074434
2       -0.221374 -0.296633 -0.341616      -0.318648 -0.205291     0.244667
3       -0.150594  0.063905  0.229450       0.010228 -0.367534    -0.697059
4        0.183731  0.488684  0.984529       1.189393  1.043633     0.748997

   monuments   gardens
0  -0.407870 -0.385061
1   0.235552  0.305428
2   0.251148  0.173360
3  -0.712159 -0.729537
4   0.680402  0.780446

[5 rows x 24 columns]
```

Conclusion

- The K-Means clustering centroids provide a clear understanding of the typical preferences for each cluster. For example:
- **Cluster 0**: This cluster tends to rate attractions like zoos, restaurants, and pubs/bars highly, but rates outdoor attractions like parks and beaches more negatively. This could represent users who prefer indoor or cultural activities.
- **Cluster 1**: Users in this cluster have a preference for nature-related attractions like parks and beaches, with lower ratings for more commercialized spots like malls. This group likely values outdoor and scenic experiences.
- **Cluster 2**: This cluster shows high ratings for both nature (parks and beaches) and cultural attractions (museums, art galleries). These users seem to appreciate a balanced mix of outdoor and cultural experiences.
- **Cluster 3**: These users tend to rate most attractions lower, particularly cultural attractions like theatres and museums, but show slightly higher interest in activities

such as swimming pools and gyms. This might represent users who are more inclined toward physical activities.

- **Cluster 4**: This cluster stands out with higher ratings across the board, particularly for fitness-related activities like gyms and swimming pools, as well as leisure spots like spas and cafes. Users in this group likely enjoy a variety of activities, both outdoor and wellness-focused.

These centroids highlight the diversity in user preferences, suggesting that different groups prioritize different types of attractions. By identifying these clusters, we can offer more personalized recommendations based on the unique tendencies of each group.