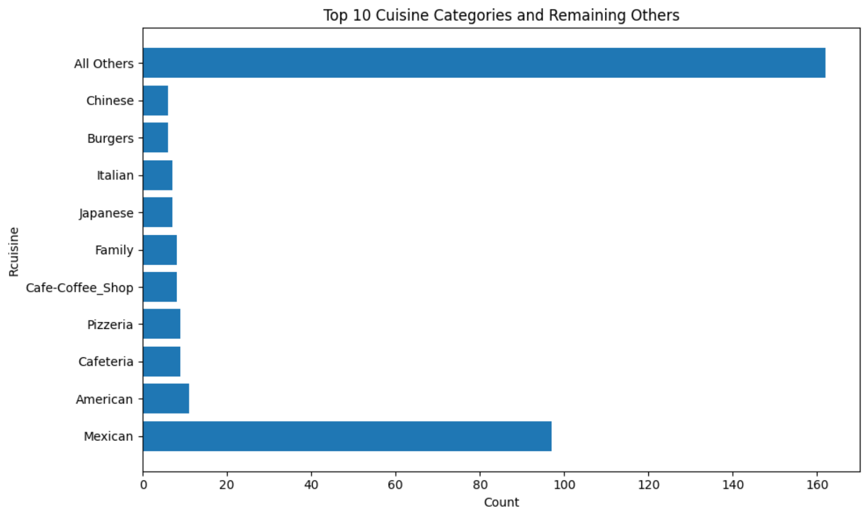
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")

pd.set_option("display.max_rows", None)
pd.set_option("display.max_columns", None)


usercuisine = pd.read_csv("usercuisine.csv")
userpayment = pd.read_csv("userpayment.csv")
userprofile = pd.read_csv("userprofile.csv")
geoplaces = pd.read_csv("geoplaces2.csv")
chefmozaccepts = pd.read_csv("chefmozaccepts.csv")
chefmozcuisine = pd.read_csv("chefmozcuisine.csv")
chefmozhours = pd.read_csv("chefmozhours4.csv")
chefmozparking = pd.read_csv("chefmozparking.csv")
ratings = pd.read_csv("ratings.csv")
```

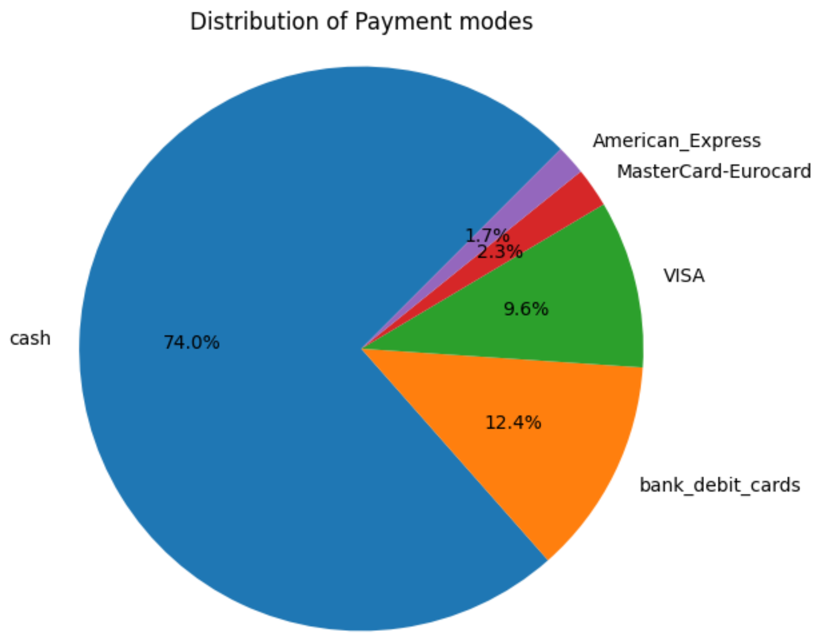
```
import pandas as pd

def dfdetails(dfs):
    for df_name, df in dfs.items():
        print(f"DataFrame: {df_name}")
        print(f"\n{df.head(3)}\n")

        print("\nData Summary:")
        print(df.describe())

        # Check for missing values
        print("\nMissing Values:")
        print(df.isnull().sum())

        # Check for duplicate rows
        print("\nDuplicate Rows:")
        print(df.duplicated().sum())

        categorical_cols = df.select_dtypes(include=["object"]).columns
        if len(categorical_cols) > 0:
            print("\nUnique Values in Categorical Columns:")
            for col in categorical_cols:
                print(f"\n{col}:")
                print(df[col].unique())

        numerical_cols = df.select_dtypes(include=["int64", "float64"]).columns
        if len(numerical_cols) > 0:
            print("\nNumerical Column Statistics:")
            for col in numerical_cols:
                print(f"\n{col}:")
                print(f"Minimum: {df[col].min()}")
                print(f"Maximum: {df[col].max()}")
                print(f"Mean: {df[col].mean()}")
                print(f"Median: {df[col].median()}")
                print(f"Standard Deviation: {df[col].std()}")



        print(f"Shape: {df.shape}")
        print(f"Columns: {', '.join(df.columns)}\n")
        print({df.info()})
        print("\n==========================================================================================================

data_frames = {
    "usercuisine": usercuisine,
    "userpayment": userpayment,
    "userprofile": userprofile,
    "geoplaces": geoplaces,
    "chefmozaccepts": chefmozaccepts,
    "chefmozcuisine": chefmozcuisine,
    "chefmozhours": chefmozhours,
    "chefmozparking": chefmozparking,
    "ratings": ratings
}

dfdetails(data_frames)
```

```
DataFrame: usercuisine

  userID  Rcuisine
0  U1001  American
1  U1002   Mexican
2  U1003   Mexican


Data Summary:
       userID Rcuisine
count     330      330
unique    138      103
top     U1135  Mexican
freq      103       97

Missing Values:
userID      0
Rcuisine    0
dtype: int64

Duplicate Rows:
0

Unique Values in Categorical Columns:

userID:
['U1001' 'U1002' 'U1003' 'U1004' 'U1005' 'U1006' 'U1007' 'U1008' 'U1009'
 'U1010' 'U1011' 'U1012' 'U1013' 'U1014' 'U1015' 'U1016' 'U1017' 'U1018'
 'U1019' 'U1020' 'U1021' 'U1022' 'U1023' 'U1024' 'U1025' 'U1026' 'U1027'
 'U1028' 'U1029' 'U1030' 'U1031' 'U1032' 'U1033' 'U1034' 'U1035' 'U1036'
 'U1037' 'U1038' 'U1039' 'U1040' 'U1041' 'U1042' 'U1043' 'U1044' 'U1045'
 'U1046' 'U1047' 'U1048' 'U1049' 'U1050' 'U1051' 'U1052' 'U1053' 'U1054'
 'U1055' 'U1056' 'U1057' 'U1058' 'U1059' 'U1060' 'U1061' 'U1062' 'U1063'
 'U1064' 'U1065' 'U1066' 'U1067' 'U1068' 'U1069' 'U1070' 'U1071' 'U1072'
 'U1073' 'U1074' 'U1075' 'U1076' 'U1077' 'U1078' 'U1079' 'U1080' 'U1081'
 'U1082' 'U1083' 'U1084' 'U1085' 'U1086' 'U1087' 'U1088' 'U1089' 'U1090'
 'U1091' 'U1092' 'U1093' 'U1094' 'U1095' 'U1096' 'U1097' 'U1098' 'U1099'
 'U1100' 'U1101' 'U1102' 'U1103' 'U1104' 'U1105' 'U1106' 'U1107' 'U1108'
 'U1109' 'U1110' 'U1111' 'U1112' 'U1113' 'U1114' 'U1115' 'U1116' 'U1117'
 'U1118' 'U1119' 'U1120' 'U1121' 'U1122' 'U1123' 'U1124' 'U1125' 'U1126'
 'U1127' 'U1128' 'U1129' 'U1130' 'U1131' 'U1132' 'U1133' 'U1134' 'U1135'
 'U1136' 'U1137' 'U1138']

Rcuisine:
['American' 'Mexican' 'Bakery' 'Breakfast-Brunch' 'Japanese'
 'Contemporary' 'Bagels' 'Cafe-Coffee_Shop' 'Continental-European'
 'Cafeteria' 'Family' 'Juice' 'Hawaiian' 'Hot_Dogs' 'Latin_American'
 'Korean' 'Italian' 'Diner' 'Fast_Food' 'Deli-Sandwiches' 'Regional'
 'Fusion' 'Portuguese' 'Indian-Pakistani' 'Eastern_European' 'Lebanese'
 'Moroccan' 'Barbecue' 'Polynesian' 'Polish' 'Chinese' 'Pizzeria'
 'Burgers' 'Afghan' 'Middle_Eastern' 'Mongolian' 'Bar' 'Cuban' 'Tex-Mex'
 'Spanish' 'Soup' 'Sushi' 'Game' 'Doughnuts' 'Australian' 'Asian'
 'Dessert-Ice_Cream' 'Seafood' 'Turkish' 'Organic-Healthy' 'Steaks'
 'Mediterranean' 'British' 'Austrian' 'Israeli' 'Russian-Ukrainian'
 'Malaysian' 'Vegetarian' 'Peruvian' 'Tapas' 'Eclectic' 'African' 'Basque'
 'Canadian' 'Irish' 'Southwestern' 'Tea_House' 'International'
 'Pacific_Northwest' 'German' 'Persian' 'Ethiopian' 'Romanian' 'Cambodian'
```

```python
cuisine_counts = usercuisine["Rcuisine"].value_counts()

top_n = 10

top_cuisine_counts = cuisine_counts.nlargest(top_n)
other_count = cuisine_counts.sum() - top_cuisine_counts.sum()

categories_to_plot = list(top_cuisine_counts.index)
categories_to_plot.append("All Others")

counts_to_plot = list(top_cuisine_counts.values)
counts_to_plot.append(other_count)

plt.figure(figsize=(10, 6))
plt.barh(categories_to_plot, counts_to_plot)
plt.xlabel("Count")
plt.ylabel("Rcuisine")
plt.title(f"Top {top_n} Cuisine Categories and Remaining Others")
plt.tight_layout()  # To prevent overlapping labels
plt.show()
```



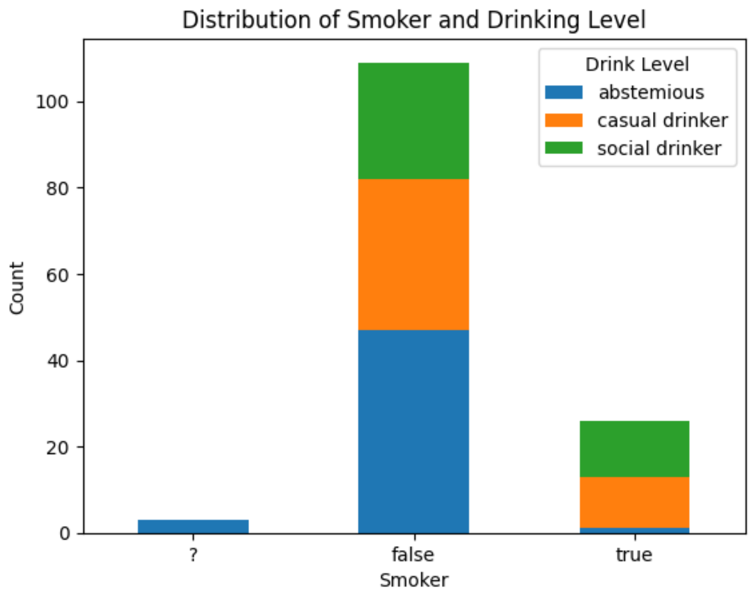
```
payment_counts = userpayment["Upayment"].value_counts()
plt.figure(figsize=(6,6))
plt.pie(payment_counts, labels=payment_counts.index, autopct='%1.1f%%',startangle = 45)
plt.title("Distribution of Payment modes")
plt.axis('equal')
plt.show()
```



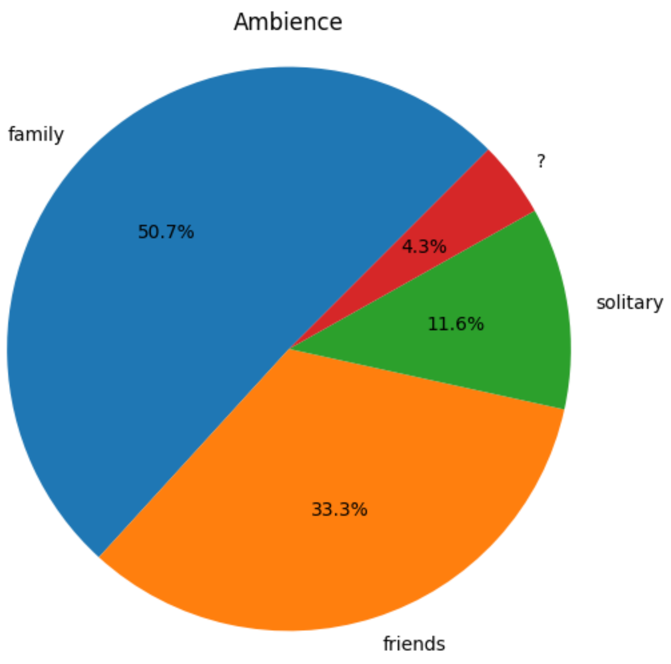
```
userprofile.columns

     Index(['userID', 'latitude', 'longitude', 'smoker', 'drink_level',
            'dress_preference', 'ambience', 'transport', 'marital_status', 'hijos',
            'birth_year', 'interest', 'personality', 'religion', 'activity',
            'color', 'weight', 'budget', 'height'],
           dtype='object')
```

```
import pandas as pd
import matplotlib.pyplot as plt

df = userprofile[['smoker', 'drink_level']]

# Group and count the occurrences of each combination
grouped_counts = df.groupby(['smoker', 'drink_level']).size().unstack(fill_value=0)

# Plot the grouped bar chart
plt.figure(figsize=(6, 6))
grouped_counts.plot(kind='bar', stacked=True)
plt.xlabel('Smoker')
plt.ylabel('Count')
plt.title('Distribution of Smoker and Drinking Level')
plt.legend(title='Drink Level', loc='upper right')
plt.xticks(rotation=0)  # Keep the x-axis labels horizontal for better readability
plt.show()
```

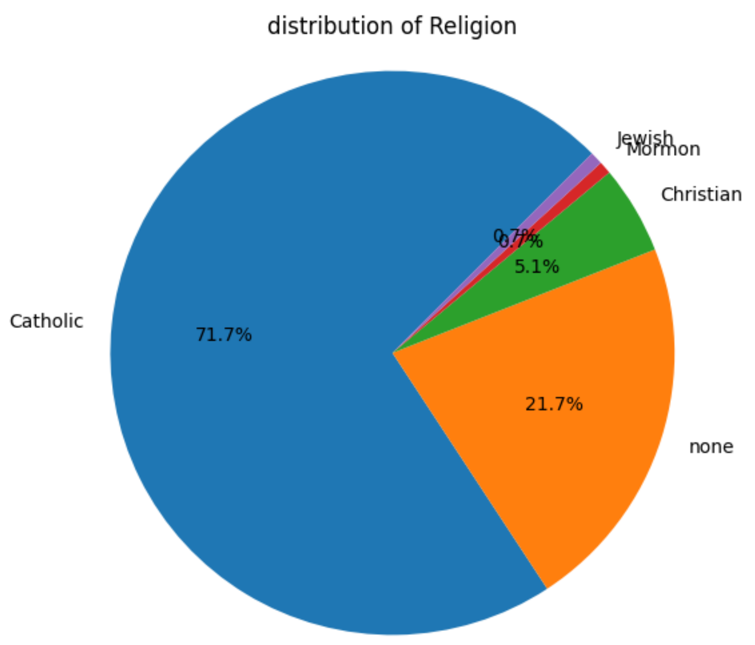
```
<Figure size 600x600 with 0 Axes>
```



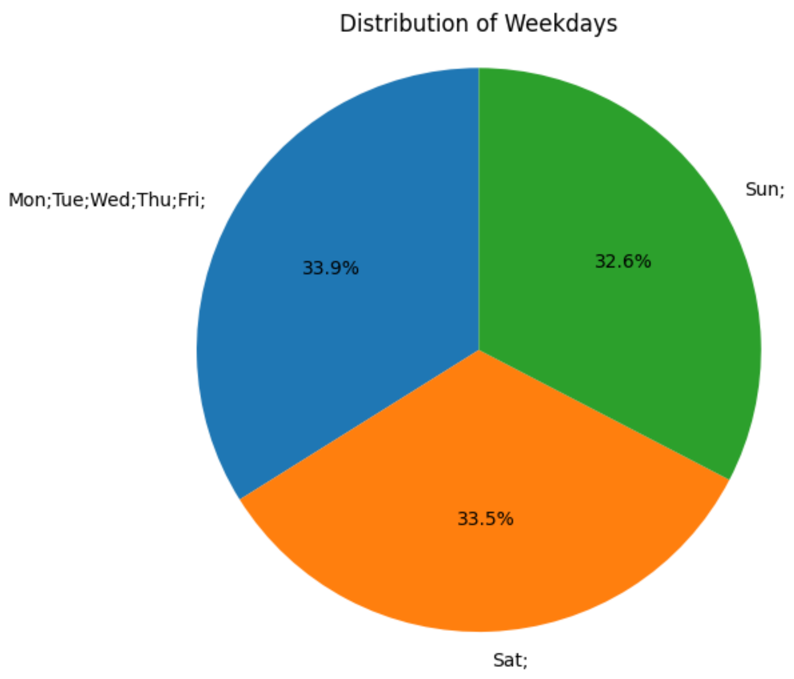
```
df = userprofile

ambience_counts = df['ambience'].value_counts()
plt.figure(figsize=(6,6))
plt.pie(ambience_counts, labels=ambience_counts.index, autopct='%1.1f%%',startangle=45)
plt.title("Ambience")
plt.axis('equal')
plt.show()
```



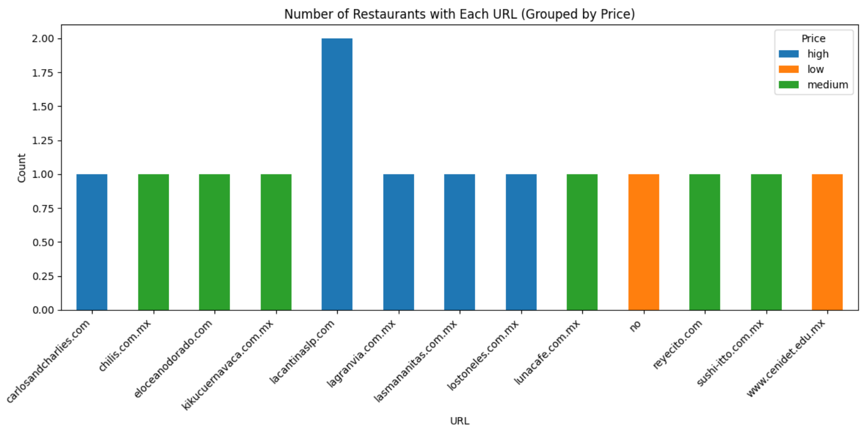
```
df = userprofile

ambience_counts = df['religion'].value_counts()
plt.figure(figsize=(6,6))
plt.pie(ambience_counts, labels=ambience_counts.index, autopct='%1.1f%%',startangle=45)
plt.title("distribution of Religion")
plt.axis('equal')
plt.show()
```



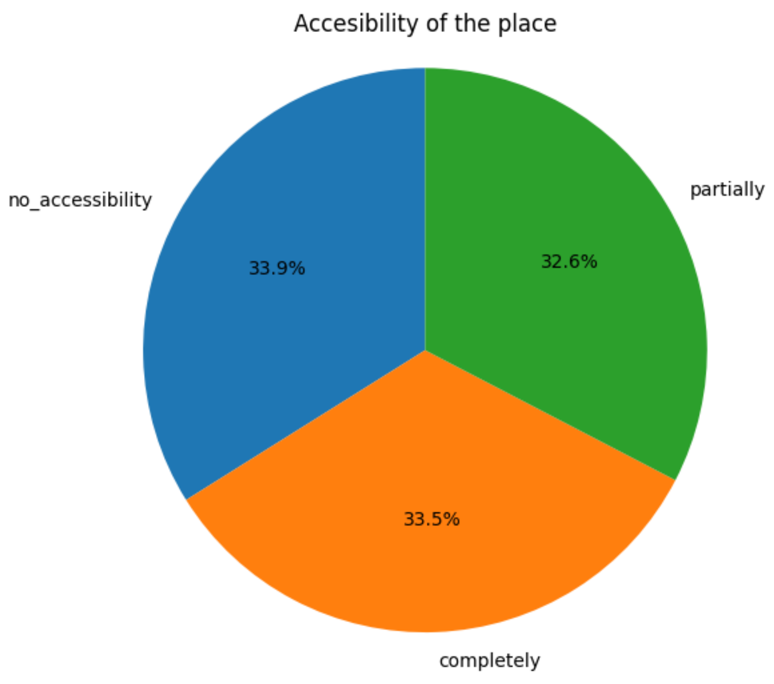
```
df = chefmozhours

ambience_counts = df['days'].value_counts()
plt.figure(figsize=(6,6))
plt.pie(ambience_counts, labels=ambience_counts.index, autopct='%1.1f%%',startangle=90)
plt.title("Distribution of Weekdays")
plt.axis('equal')
plt.show()
```



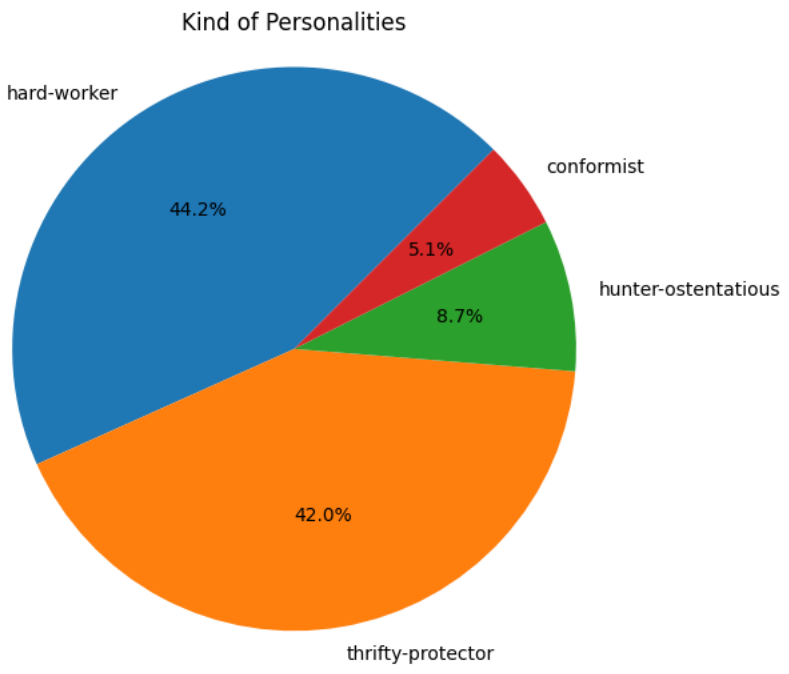
```
df = geoplaces

# Exclude rows with the value '?'
df_filtered = df[df['url'] != '?']

# Group the data by 'url' and 'price' and calculate the counts
url_price_counts = df_filtered.groupby(['url', 'price']).size().unstack(fill_value=0)

# Create a bar chart with hue="price"
plt.figure(figsize=(12, 6))
url_price_counts.plot(kind='bar', stacked=True, ax=plt.gca())
plt.xlabel("URL")
plt.ylabel("Count")
plt.title("Number of Restaurants with Each URL (Grouped by Price)")
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels and align them to the right
plt.legend(title="Price", loc="upper right")  # Add legend for price categories
plt.tight_layout()  # To prevent overlapping labels
plt.show()
```



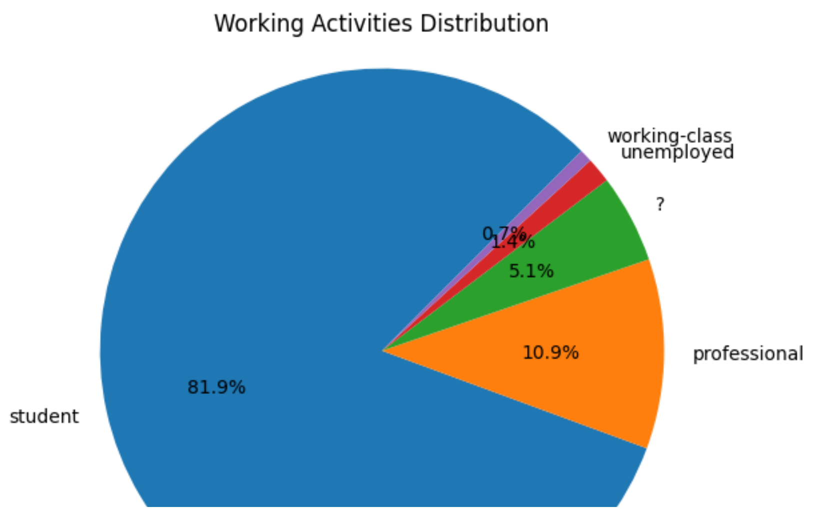
```
accessibility = geoplaces['accessibility'].value_counts()
plt.figure(figsize=(6,6))
plt.pie(ambience_counts, labels=accessibility.index, autopct='%1.1f%%',startangle=90)
plt.title("Accesibility of the place")
plt.axis('equal')
plt.show()
```



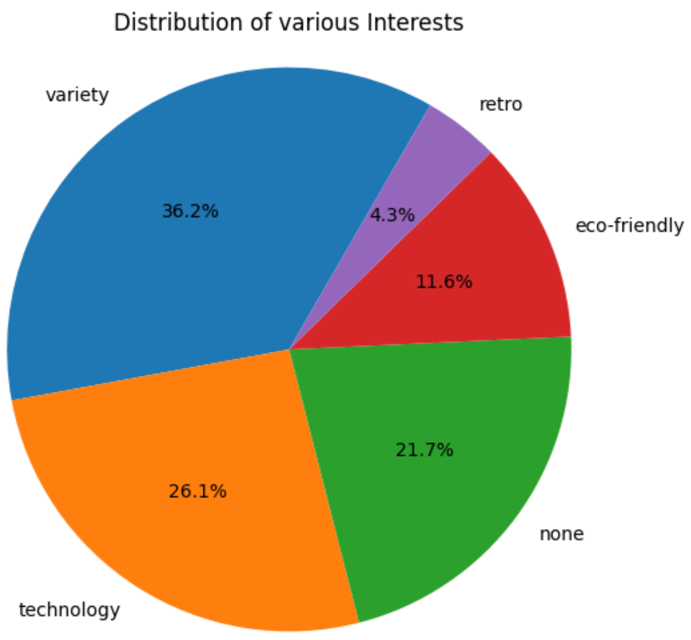
```
df = userprofile

ambience_counts = df['personality'].value_counts()
plt.figure(figsize=(6,6))
plt.pie(ambience_counts, labels=ambience_counts.index, autopct='%1.1f%%',startangle=45)
plt.title("Kind of Personalities")
plt.axis('equal')
plt.show()
```



```
df = userprofile

ambience_counts = df['activity'].value_counts()
plt.figure(figsize=(6,6))
plt.pie(ambience_counts, labels=ambience_counts.index, autopct='%1.1f%%',startangle=45)
plt.title("Working Activities Distribution")
plt.axis('equal')
plt.show()
```



```
df = userprofile

ambience_counts = df['interest'].value_counts()
plt.figure(figsize=(6,6))
plt.pie(ambience_counts, labels=ambience_counts.index, autopct='%1.1f%%',startangle=60)
plt.title("Distribution of various Interests")
plt.axis('equal')
plt.show()
```



```
df = userprofile

ambience_counts = df['transport'].value_counts()
plt.figure(figsize=(6,6))
plt.pie(ambience_counts, labels=ambience_counts.index, autopct='%1.1f%%',startangle=45)
plt.title("User Profile Distribution")
plt.axis('equal')
plt.show()
```