```
# Importing necessary libraries for data manipulation, visualization, and machine learning
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.impute import SimpleImputer

# Load the dataset
housing_survey_df = pd.read_csv('HousingSurvey1.csv')

# Inspect the first few rows to understand the structure and initial data
housing_survey_df.head()

# Get an overview of the dataset, including data types, null counts, and memory usage
# This step helps us identify missing values, types of data, and columns that might need cleaning
housing_survey_df.info()

# Summary statistics of the numerical columns in the dataset (e.g., mean, std, min, max)
# This gives an understanding of the numerical distributions and potential outliers
housing_survey_df.describe()

# Step 1: Data Cleaning

# Remove rows where all answers are missing
# This ensures that incomplete or irrelevant rows are excluded from the analysis
housing_survey_cleaned = housing_survey_df.dropna(how='all')

# Remove rows where students are not from Purdue ('Purdue student?' column has 'No')
# We only want to focus on Purdue students in this analysis
housing_survey_cleaned = housing_survey_cleaned[housing_survey_cleaned['Purdue student?'] != 'No']

# Drop irrelevant columns that won't contribute to clustering
# These columns are metadata (such as start/end date, IP address) that do not influence student preferences
housing_survey_cleaned = housing_survey_cleaned.drop(columns=[
    'StartDate', 'EndDate', 'Status', 'IPAddress', 'Progress',
    'Duration (in seconds)', 'Finished', 'RecordedDate', 'ResponseId',
    'RecipientLastName', 'RecipientFirstName', 'RecipientEmail',
    'ExternalReference', 'DistributionChannel', 'UserLanguage'
])
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 67 entries, 0 to 66
Data columns (total 43 columns):
 #   Column                                       Non-Null Count  Dtype
---  ------                                       --------------  -----
 0   StartDate                                    67 non-null     object
 1   EndDate                                      67 non-null     object
 2   Status                                       67 non-null     object
 3   IPAddress                                    51 non-null     object
 4   Progress                                     67 non-null     int64
 5   Duration (in seconds)                        67 non-null     int64
 6   Finished                                     67 non-null     bool
 7   RecordedDate                                 67 non-null     object
 8   ResponseId                                   67 non-null     object
 9   RecipientLastName                            0 non-null      float64
 10  RecipientFirstName                           0 non-null      float64
 11  RecipientEmail                               0 non-null      float64
 12  ExternalReference                            0 non-null      float64
 13  LocationLatitude                             66 non-null     float64
 14  LocationLongitude                            66 non-null     float64
 15  DistributionChannel                          67 non-null     object
 16  UserLanguage                                 67 non-null     object
 17  Q_RecaptchaScore                             67 non-null     float64
 18  Purdue student?                              66 non-null     object
 19  Program                                      59 non-null     object
 20  Other_Program                                1 non-null      object
 21  Age_Group                                    59 non-null     object
 22  Gender                                       59 non-null     object
```

```
 23  Current_Living                           59 non-null     object
 24  On_Campus_ReasonsToChoose                9 non-null      object
 25  On_Campus_ReasonsToChoose_Other          0 non-null      float64
 26  On_Campus_StrongestReasonToChoose        9 non-null      object
 27  On_Campus_StrongestReasonToChoose_ Other  1 non-null     object
 28  Current_Rent_Ex_Utility                  9 non-null      object
 29  Next_Sem_OffCampus?                      9 non-null      object
 30  Off_Campus_CampusProximity               55 non-null     object
 31  Off_Campus_UtilityProximity              55 non-null     object
 32  Campus_Commute                           55 non-null     object
 33  Campus_Commute_Other                     0 non-null      float64
 34  Amenities_Community_Events               54 non-null     object
 35  Amenities_Advanced_Study_Rooms           55 non-null     object
 36  Amenities_Sports_Facilities              55 non-null     object
 37  Amenities_ 24/7_Maintenance              54 non-null     object
 38  Amenities_Enhanced_Security              55 non-null     object
 39  Amenities_ Sustainable_Features          55 non-null     object
 40  Preference_Private_Bathroom              55 non-null     float64
 41  Preference_OffCampus_Rent                55 non-null     object
 42  Recommend_CurrentHousing                 54 non-null     object
dtypes: bool(1), float64(10), int64(2), object(30)
memory usage: 22.2+ KB
```

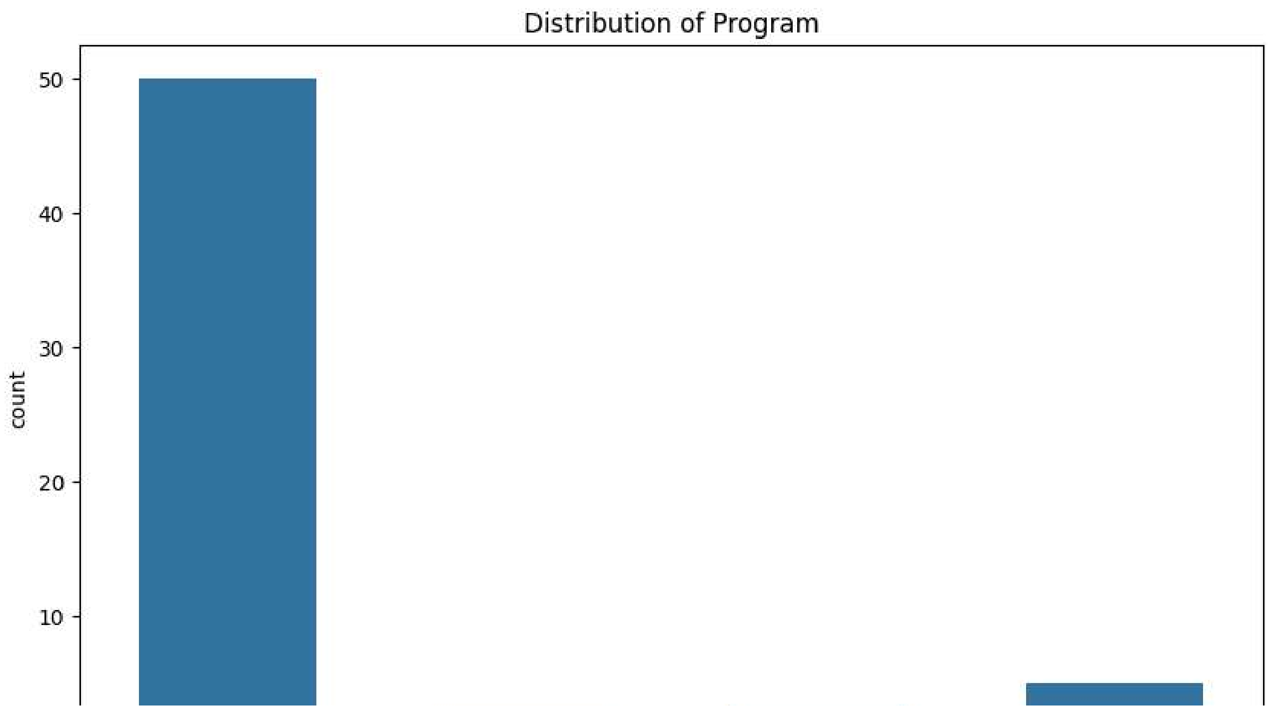
```
# Step 2: Handling Missing Values

# Identify which columns are numerical and which are categorical for imputation
# Numerical columns will use median imputation; categorical columns will use mode imputation
numeric_cols = housing_survey_cleaned.select_dtypes(include=[np.number]).columns
categorical_cols = housing_survey_cleaned.select_dtypes(include=['object']).columns

# Impute missing values in numerical columns using the median
# The median is a robust measure of central tendency that reduces the effect of outliers
imputer_num = SimpleImputer(strategy='median')

# Identify columns where all values are NaN and exclude them from further processing
nan_cols = housing_survey_cleaned[numeric_cols].columns[housing_survey_cleaned[numeric_cols].isna().all()]
numeric_cols = numeric_cols.drop(nan_cols)  # Dropping completely empty numerical columns

# Apply median imputation to the remaining numeric columns to fill in missing values
housing_survey_cleaned[numeric_cols] = imputer_num.fit_transform(housing_survey_cleaned[numeric_cols])

# Impute missing values in categorical columns using the mode (most frequent value)
# The mode is appropriate for categorical data since it represents the most common value
imputer_cat = SimpleImputer(strategy='most_frequent')
housing_survey_cleaned[categorical_cols] = imputer_cat.fit_transform(housing_survey_cleaned[categorical_cols])
```

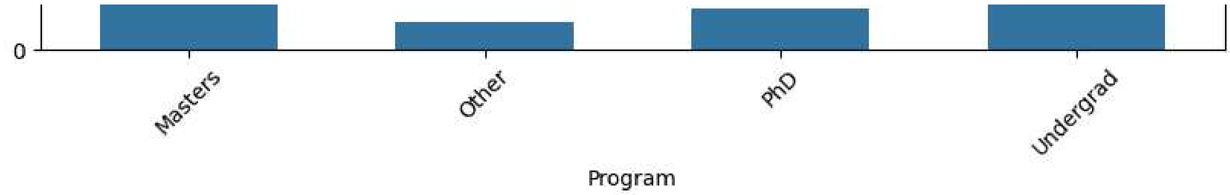
```
# Step 3: Data Preprocessing

# Convert categorical variables into numerical format using one-hot encoding
# This allows categorical variables (e.g., Gender, Program) to be used in machine learning algorithms like K-Means
# drop_first=True removes one dummy variable to avoid multicollinearity
housing_survey_encoded = pd.get_dummies(housing_survey_cleaned, drop_first=True)

# Standardize the data using StandardScaler to ensure all features contribute equally to the clustering process
# Standardization ensures that variables with different scales (e.g., rent vs age) are treated equally in the model
scaler = StandardScaler()
housing_survey_scaled = scaler.fit_transform(housing_survey_encoded)
```

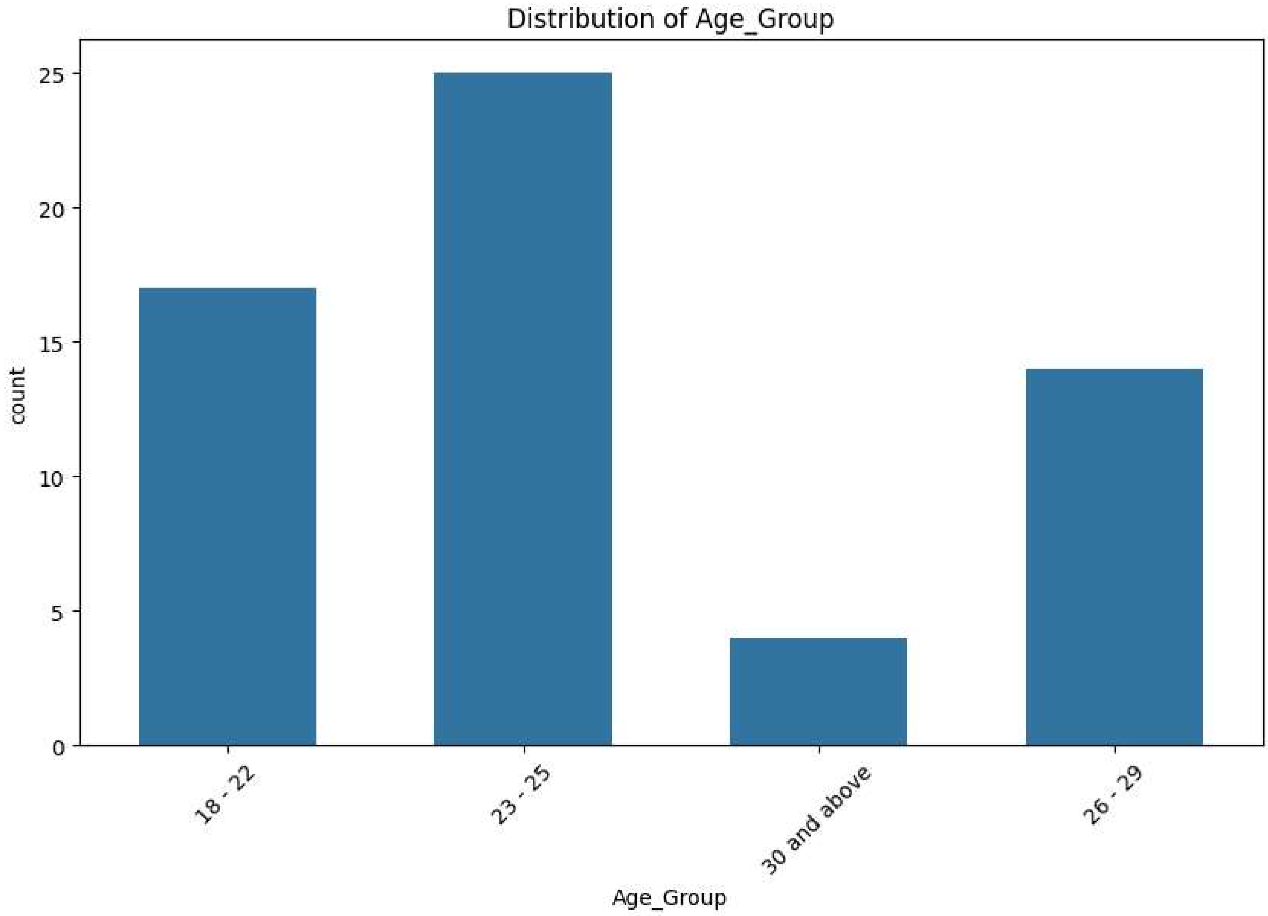
```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/extmath.py:1137: RuntimeWarning: invalid value encountered i
  updated_mean = (last_sum + new_sum) / updated_sample_count
/usr/local/lib/python3.10/dist-packages/sklearn/utils/extmath.py:1142: RuntimeWarning: invalid value encountered i
  T = new_sum / new_sample_count
/usr/local/lib/python3.10/dist-packages/sklearn/utils/extmath.py:1162: RuntimeWarning: invalid value encountered i
  new_unnormalized_variance -= correction**2 / new_sample_count
```

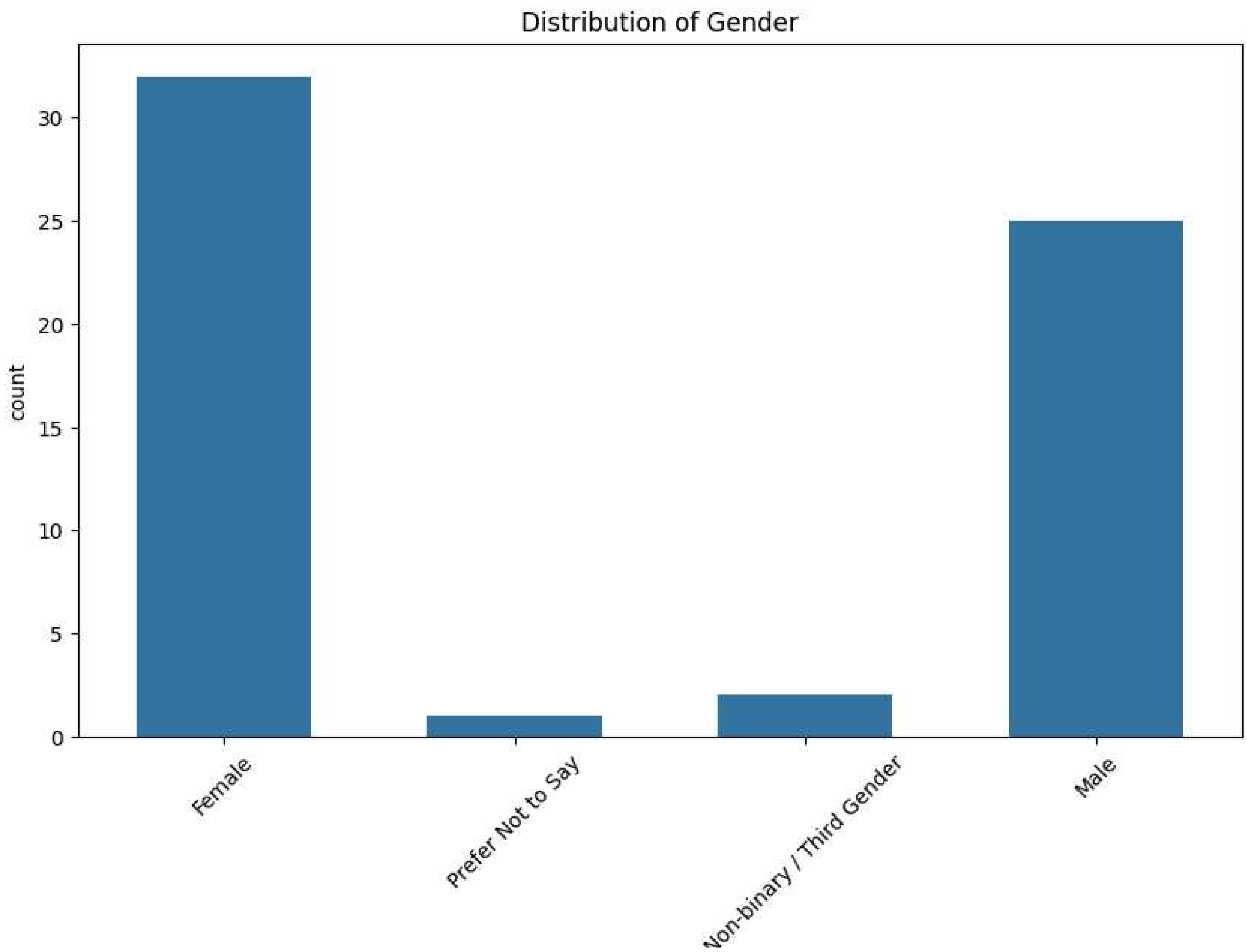
```
# Step 4: Descriptive Statistics and Visualizations

# Calculate and display summary statistics for numerical columns
# This helps us understand the range, central tendencies, and variability of the numerical features in the dataset
print("Descriptive statistics for numerical features:")
print(housing_survey_cleaned[numeric_cols].describe())

# List of categorical columns for further analysis and visualization
categorical_columns_to_analyze = ['Program', 'Age_Group', 'Gender', 'Current_Living']

# For each categorical column, print the frequency counts (how many times each category appears in the data)
# This provides insights into the distribution of categories in the dataset (e.g., how many students belong to each ag
print("\nDescriptive Statistics for Categorical Variables:")
for column in categorical_columns_to_analyze:
    print(f"\n{column} value counts:")
    print(housing_survey_cleaned[column].value_counts())

# Visualizing the distribution of each categorical variable using bar plots
# Bar plots help us see the distribution of categories (e.g., gender distribution, housing choices)
for column in categorical_columns_to_analyze:
    plt.figure(figsize=(10, 6))
    sns.countplot(x=housing_survey_cleaned[column], width=0.6)  # Bar width reduced for clarity
    plt.title(f"Distribution of {column}")
    plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
    plt.show()

# Visualize rent willingness (how much rent students are willing to pay for off-campus housing)
plt.figure(figsize=(15, 6))
sns.countplot(x=housing_survey_cleaned['Preference_OffCampus_Rent'])
plt.title('Rent Willingness Distribution')
plt.show()

# Correlation heatmap to show relationships between all numerical and encoded categorical variables
# This helps us understand how features are related to each other (e.g., whether rent willingness is correlated with p
plt.figure(figsize=(10, 6))
sns.heatmap(pd.DataFrame(housing_survey_scaled, columns=housing_survey_encoded.columns).corr(), cmap="coolwarm", annot
plt.title("Correlation Heatmap")
plt.show()
```

```
Descriptive statistics for numerical features:
       LocationLatitude  LocationLongitude  Q_RecaptchaScore  \
count         60.000000          60.000000         60.000000
mean          40.713645         -83.097943          0.968333
std            2.195369          21.203836          0.092958
min           33.447500        -121.919100          0.400000
25%           40.444400         -86.925600          1.000000
50%           40.444400         -86.925600          1.000000
75%           40.444400         -86.925600          1.000000
max           48.951200           2.338700          1.000000

       Preference_Private_Bathroom
count                    60.000000
mean                      4.350000
std                       0.898681
min                       2.000000
25%                       4.000000
50%                       5.000000
75%                       5.000000
max                       5.000000

Descriptive Statistics for Categorical Variables:

Program value counts:
Program
Masters      50
Undergrad     5
PhD           3
Other         2
Name: count, dtype: int64

Age_Group value counts:
Age_Group
23 - 25         25
18 - 22         17
26 - 29         14
30 and above     4
Name: count, dtype: int64

Gender value counts:
Gender
Female                        32
Male                          25
Non-binary / Third Gender      2
Prefer Not to Say              1
Name: count, dtype: int64

Current_Living value counts:
Current_Living
Off-campus    51
On-campus      9
Name: count, dtype: int64
```

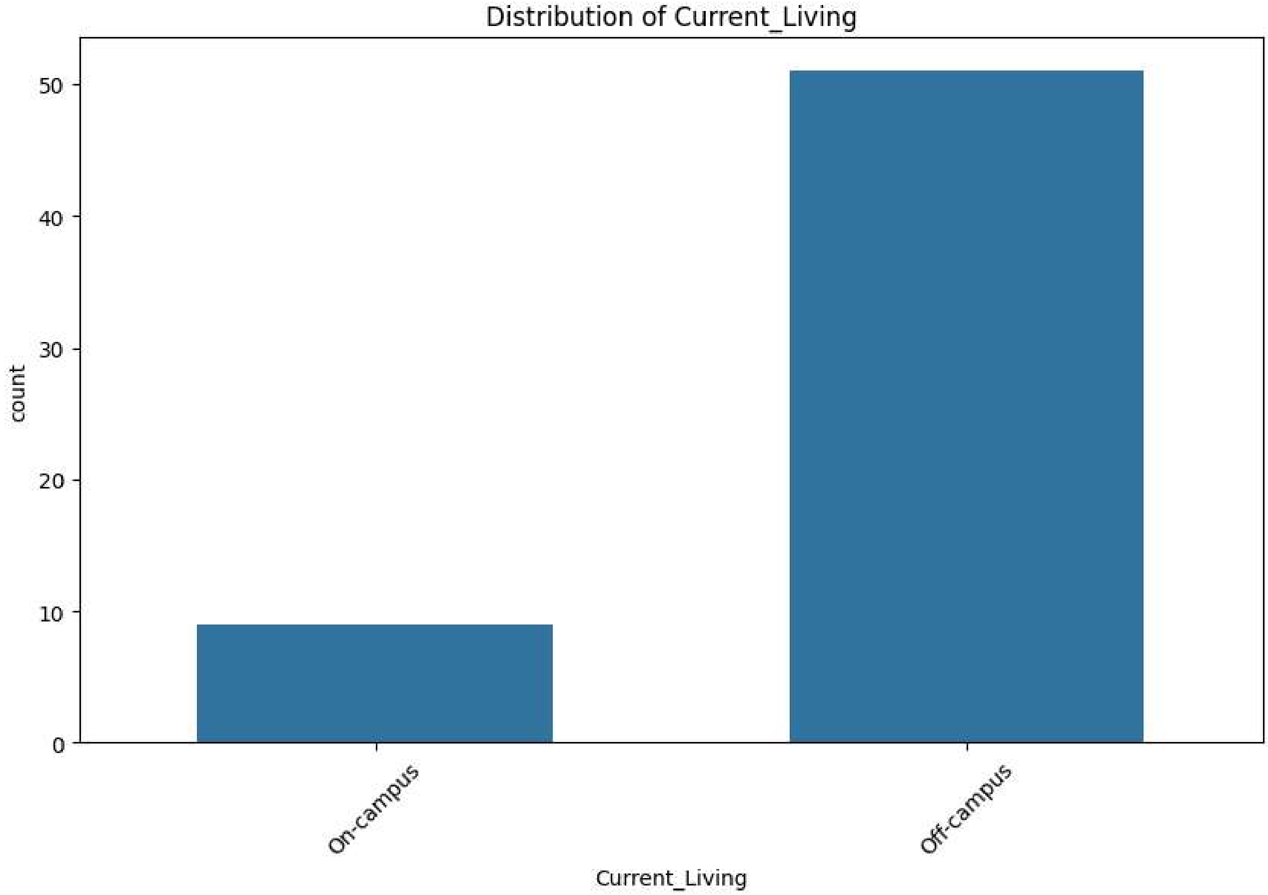


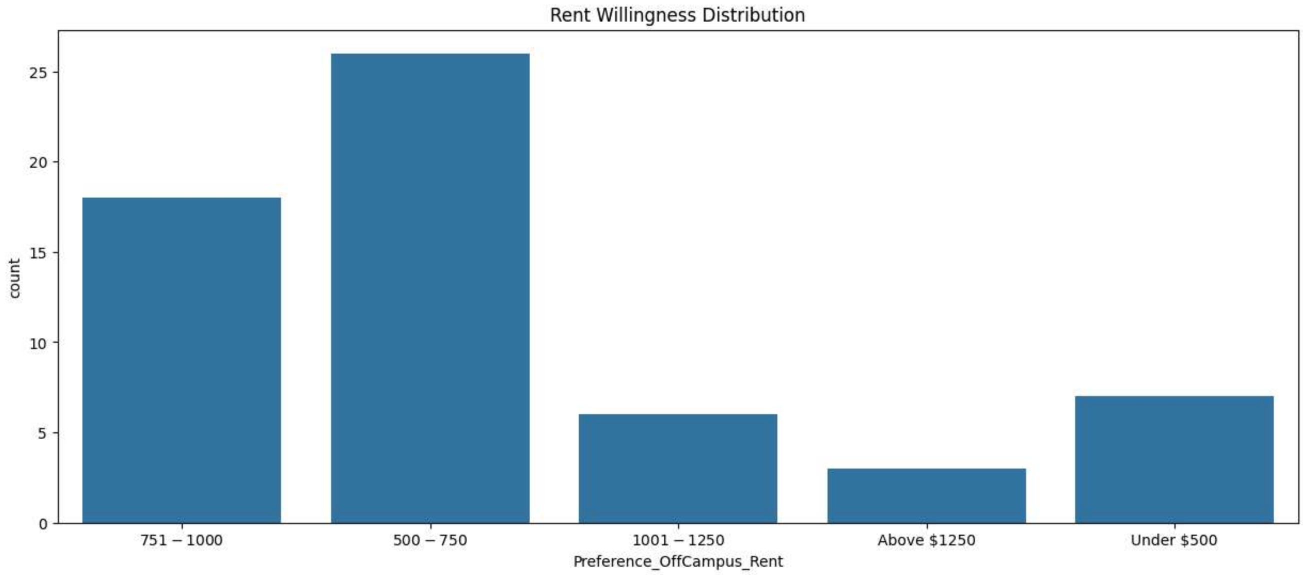

0
Masters
Other
PhD
Undergrad
Program

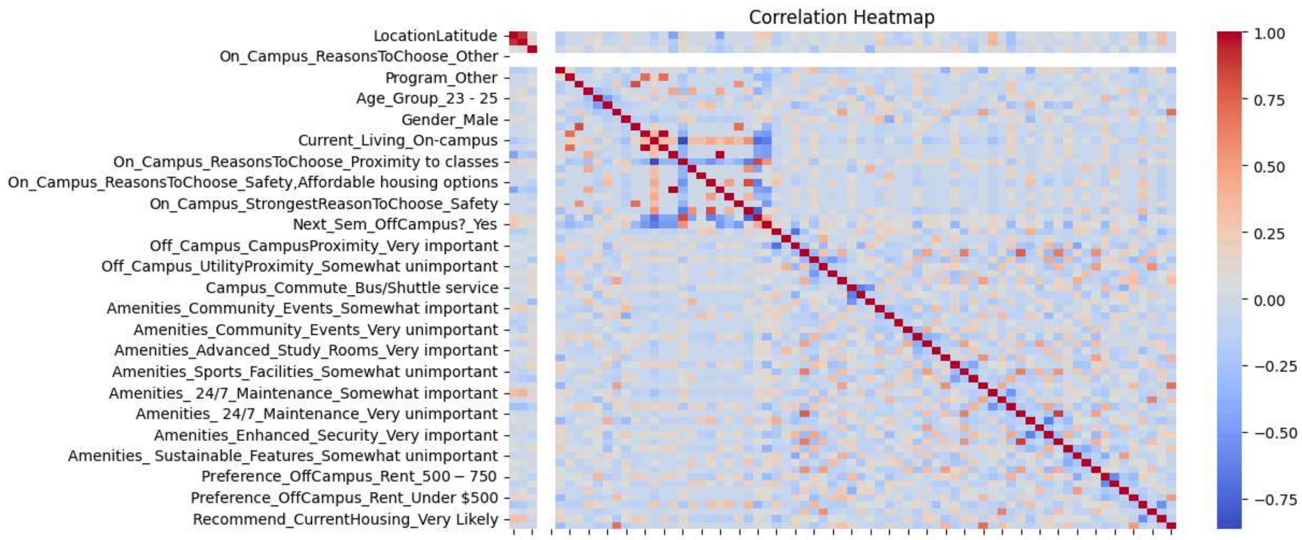

Distribution of Age_Group
count
0
5
10
15
20
25
18 - 22
23 - 25
30 and above
26 - 29
Age_Group

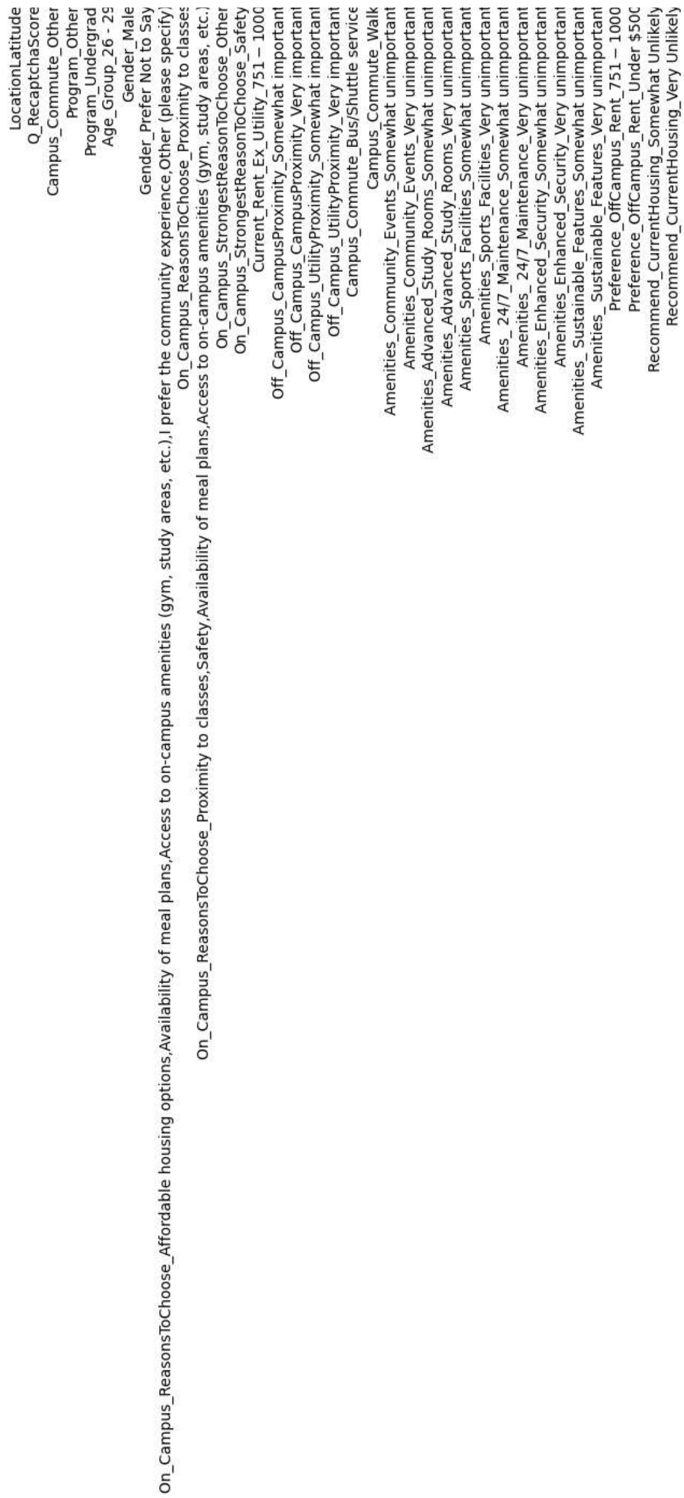

Distribution of Gender
count
0
5
10
15
20
25
30
Female
Prefer Not to Say
Non-binary / Third Gender
Male

Gender


Distribution of Current_Living
count
0
10
20
30
40
50
On-campus
Off-campus
Current_Living

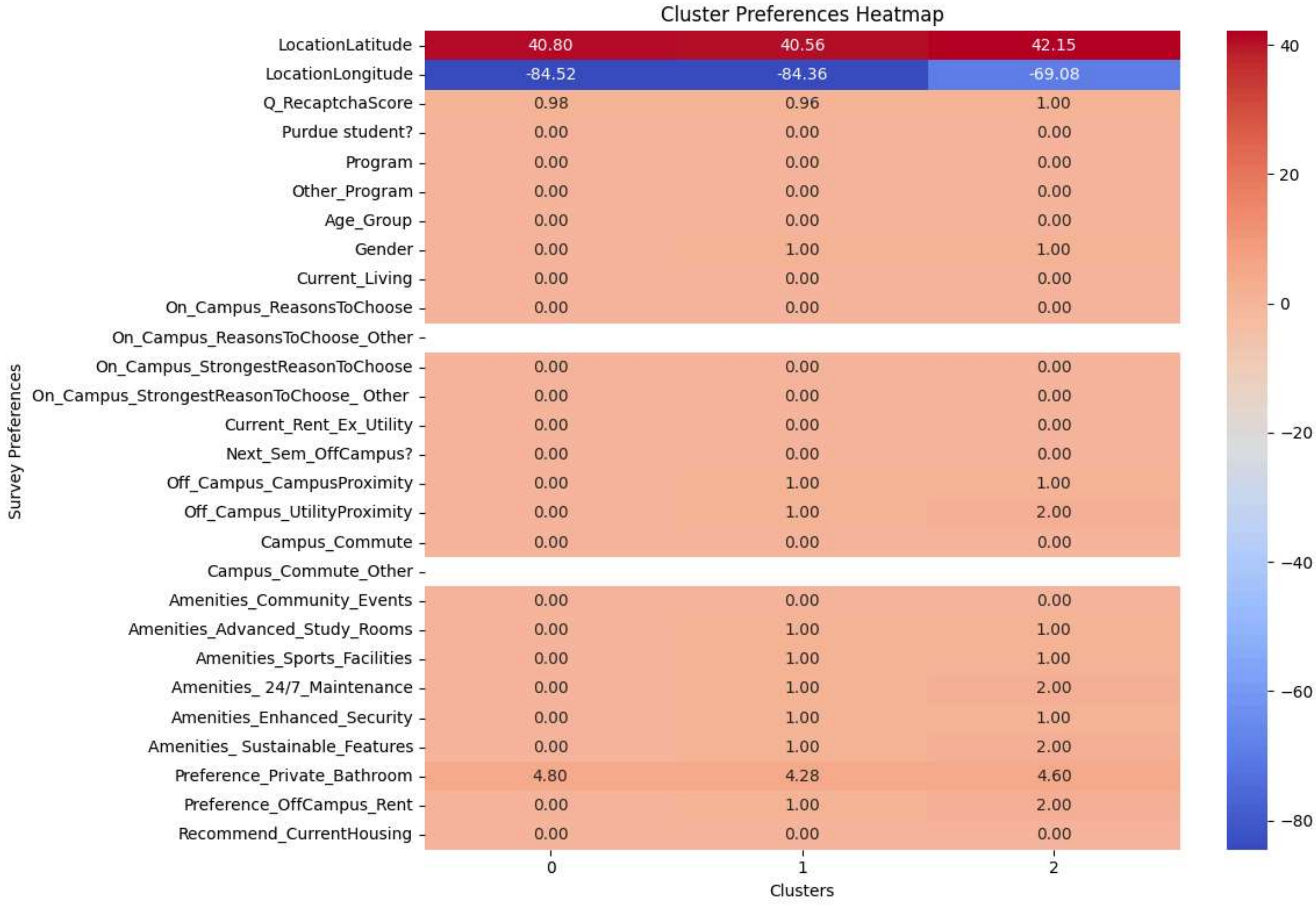

Rent Willingness Distribution
count
0
5
10
15
20
25
751 – 1000
500 – 750
1001 – 1250
Above $1250
Under $500
Preference_OffCampus_Rent


Correlation Heatmap
LocationLatitude
On_Campus_ReasonsToChoose_Other
Program_Other
Age_Group_23 - 25
Gender_Male
Current_Living_On-campus
On_Campus_ReasonsToChoose_Proximity to classes
On_Campus_ReasonsToChoose_Safety,Affordable housing options
On_Campus_StrongestReasonToChoose_Safety
Next_Sem_OffCampus?_Yes
Off_Campus_CampusProximity_Very important
Off_Campus_UtilityProximity_Somewhat unimportant
Campus_Commute_Bus/Shuttle service
Amenities_Community_Events_Somewhat important
Amenities_Community_Events_Very unimportant
Amenities_Advanced_Study_Rooms_Very important
Amenities_Sports_Facilities_Somewhat unimportant
Amenities_ 24/7_Maintenance_Somewhat important
Amenities_ 24/7_Maintenance_Very unimportant
Amenities_Enhanced_Security_Very important
Amenities_ Sustainable_Features_Somewhat unimportant
Preference_OffCampus_Rent_500 – 750
Preference_OffCampus_Rent_Under $500
Recommend_CurrentHousing_Very Likely
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75


LocationLatitude
Q_RecaptchaScore
Campus_Commute_Other
Program_Other
Program_Undergrad
Age_Group_26 - 29
Gender_Male
Gender_Prefer Not to Say
On_Campus_ReasonsToChoose_Affordable housing options,Availability of meal plans,Access to on-campus amenities (gym, study areas, etc.),I prefer the community experience,Other (please specify)
On_Campus_ReasonsToChoose_Proximity to classes
On_Campus_ReasonsToChoose_Proximity to classes,Safety,Availability of meal plans,Access to on-campus amenities (gym, study areas, etc.)
On_Campus_StrongestReasonToChoose_Other
On_Campus_StrongestReasonToChoose_Safety
Current_Rent_Ex_Utility_751 – 1000
Off_Campus_CampusProximity_Somewhat important
Off_Campus_CampusProximity_Very important
Off_Campus_UtilityProximity_Somewhat important
Off_Campus_UtilityProximity_Very important
Campus_Commute_Bus/Shuttle service
Campus_Commute_Walk
Amenities_Community_Events_Somewhat unimportant
Amenities_Community_Events_Very unimportant
Amenities_Advanced_Study_Rooms_Somewhat unimportant
Amenities_Advanced_Study_Rooms_Very unimportant
Amenities_Sports_Facilities_Somewhat unimportant
Amenities_Sports_Facilities_Very unimportant
Amenities_24/7_Maintenance_Somewhat unimportant
Amenities_24/7_Maintenance_Very unimportant
Amenities_Enhanced_Security_Somewhat unimportant
Amenities_Enhanced_Security_Very unimportant
Amenities_Sustainable_Features_Somewhat unimportant
Amenities_Sustainable_Features_Very unimportant
Preference_OffCampus_Rent_751 – 1000
Preference_OffCampus_Rent_Under $500
Recommend_CurrentHousing_Somewhat Unlikely
Recommend_CurrentHousing_Very Unlikely

```
# Step 5: Clustering (Model Building)

# Apply K-Means clustering to divide students into 3 clusters based on their preferences
# K-Means attempts to minimize the variance within each cluster by assigning students with similar preferences to the

# Check for any NaN values in the scaled data before applying K-Means clustering
print("NaN values in scaled data before clustering:", np.isnan(housing_survey_scaled).sum())

# Fill NaN or infinite values in the scaled data
# np.nan_to_num replaces NaNs with 0 and infinite values with the largest/smallest possible float values
housing_survey_scaled = np.nan_to_num(housing_survey_scaled, nan=0.0, posinf=np.finfo(np.float64).max, neginf=np.finfo

# Check again to confirm there are no NaN values after cleaning
print("NaN values in scaled data after cleaning:", np.isnan(housing_survey_scaled).sum())

# Applying K-Means clustering with 3 clusters
# We are using 3 clusters based on prior exploratory analysis (e.g., Elbow Method or Silhouette Score)
kmeans = KMeans(n_clusters=3, random_state=42)
housing_survey_cleaned['Cluster'] = kmeans.fit_predict(housing_survey_scaled)  # Assign each student to a cluster

# Analyzing the clusters
# Grouping the data by the 'Cluster' column to calculate the mean (for numeric variables) and mode (for categorical va
cluster_analysis = housing_survey_cleaned.groupby('Cluster').agg(
    lambda x: x.mean() if pd.api.types.is_numeric_dtype(x) else x.value_counts().index[0]  # Use mode for categorical
)

# Convert categorical columns in cluster_analysis to numerical for better visualization in heatmaps
for col in cluster_analysis.columns:
    if not pd.api.types.is_numeric_dtype(cluster_analysis[col]):
        cluster_analysis[col] = pd.factorize(cluster_analysis[col])[0]

# Visualizing the preferences of each cluster using a heatmap
# This heatmap will highlight differences in preferences between clusters (e.g., rent willingness, amenity preferences
plt.figure(figsize=(12, 8))
sns.heatmap(cluster_analysis.T, cmap="coolwarm", annot=True, fmt=".2f")  # Format to show two decimal places
plt.title("Cluster Preferences Heatmap")
plt.xlabel("Clusters")
plt.ylabel("Survey Preferences")
plt.tight_layout()
plt.show()
```

```
NaN values in scaled data before clustering: 120
NaN values in scaled data after cleaning: 0
```




```
# Step 6: Insights

# Summarize insights for each cluster by displaying the most important preferences and characteristics
# For example, which cluster values premium amenities or lower rent more
for cluster in range(3):
    print(f"Cluster {cluster}:")
    print(cluster_analysis.loc[cluster])
    print("\n")
```